Package

com.sparsity.sparksee.algorithms
Class ConnectedComponents

This class contains the results processed on a Connectivity algorithm. These results contain information related to the connected components found. We must consider that each connected component has a number in order to identify it. These number identifiers are values from 0 to N-1, where N is the number of different connected components found.

When executing any implementation of the Connectivity, it is possible to indicate whether the results of the execution must be stored persistently using the class Connectivity setMaterializedAttribute method. In case the results are set to be materialized, users can retrieve this data whenever they want, even if the graph has been closed and opened again, just by creating a new instance of this class.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author: Sparsity Technologies http://www.sparsity-technologies.com

Constructor Summary

<table>
<thead>
<tr>
<th>public</th>
<th>ConnectedComponents(Session s, String materializedattribute)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creates a new instance of ConnectedComponents.</td>
<td></td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>void</th>
<th>close()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Closes the ConnectedComponents instance.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>long</th>
<th>getConnectedComponent(long idNode)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns the connected component where the given node belongs to.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>long</th>
<th>getCount()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns the number of connected components found in the graph.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Objects</th>
<th>getNodes(long idConnectedComponent)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns the collection of nodes contained in the given connected component.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>long</th>
<th>getSize(long idConnectedComponent)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns the number of nodes contained in the given connected component.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>isConnected()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gets if ConnectedComponents instance has been closed or not.</td>
<td></td>
</tr>
</tbody>
</table>
### Methods inherited from class java.lang.Object

- clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

### Methods inherited from interface java.io.Closeable

- close

### Constructors

**ConnectedComponents**

```java
public ConnectedComponents(Session s,
                          String materializedattribute)
```

Creates a new instance of ConnectedComponents.

This constructor method can only be called when a previous execution of any implementation of the Connectivity class has materialized the results in a common attribute type for all the nodes in the graph. For further information about materializing the results processed on any Connectivity execution see the documentation of the Connectivity#SetMaterializedAttribute method.

**Parameters:**
- `s` - [in] Session to get the graph Graph on which the information will be retrieved just by getting the values contained in the given common attribute type for all the nodes in the graph and processing them.
- `materializedattribute` - [in] The common attribute type for all the nodes in the graph where data will be retrieved in order to process the results related to the connected components found in the graph.

### Methods

#### getSize

```java
public long getSize(long idConnectedComponent)
```

Returns the number of nodes contained in the given connected component.

**Parameters:**
- `idConnectedComponent` - The connected component for which the number of nodes contained in it will be returned.

**Results:**
- The number of nodes contained in the given connected component.

#### getCount

```java
public long getCount()
```

Returns the number of connected components found in the graph.

**Results:**
- The number of connected components found in the graph.
getConnectedComponent
public long getConnectedComponent(long idNode)

Returns the connected component where the given node belongs to.

Parameters:
   idNode - [in] The node identifier for which the connected component identifier where it belongs will be returned.

Returns:
The connected component identifier where the given node identifier belongs to.

getNodes
public Objects getNodes(long idConnectedComponent)

Returns the collection of nodes contained in the given connected component.

Parameters:
   idConnectedComponent - The connected component for which the collection of nodes contained in it will be returned.

Returns:
The collection of node identifiers contained in the given connected component.

isClosed
public boolean isClosed()

Gets if ConnectedComponents instance has been closed or not.

Returns:
   TRUE if the ConnectedComponents instance has been closed, FALSE otherwise.

See Also:
close()

close
public void close()

Closes the ConnectedComponents instance.

It must be called to ensure the integrity of all data.
public class Connectivity extends Object implements Closeable

Connectivity class.

Any class implementing this abstract class can be used to solve a problem related to graph connectivity as finding the strongly connected components or finding the weakly connected components.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author:
Sparsity Technologies http://www.sparsity-technologies.com

---

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void addAllNodeTypes()</code></td>
<td>Allows connectivity through all node types of the graph.</td>
</tr>
<tr>
<td><code>void addNodeType(int t)</code></td>
<td>Allows connectivity through nodes of the given type.</td>
</tr>
<tr>
<td><code>void close()</code></td>
<td>Closes the Connectivity instance.</td>
</tr>
<tr>
<td><code>void excludeEdges(Objects edges)</code></td>
<td>Set which edges can't be used.</td>
</tr>
<tr>
<td><code>void excludeNodes(Objects nodes)</code></td>
<td>Set which nodes can't be used.</td>
</tr>
<tr>
<td><code>ConnectedComponents getConnectedComponents()</code></td>
<td>Returns the results generated by the execution of the algorithm.</td>
</tr>
<tr>
<td><code>boolean isClosed()</code></td>
<td>Gets if Connectivity instance has been closed or not.</td>
</tr>
<tr>
<td><code>void run()</code></td>
<td>Runs the algorithm in order to find the connected components.</td>
</tr>
<tr>
<td><code>void setMaterializedAttribute(String attributeName)</code></td>
<td>Creates a new common attribute type for all node types in the graph in order to store, persistently, the results related to the connected components found while executing this algorithm.</td>
</tr>
</tbody>
</table>
Methods inherited from class java.lang.Object

| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

Methods inherited from interface java.io.Closeable

| close |

## Methods

**addAllNodeTypes**

```java
public void addAllNodeTypes()
```

Allows connectivity through all node types of the graph.

**run**

```java
public void run()
```

Runs the algorithm in order to find the connected components.

This method can be called only once.

**excludeNodes**

```java
public void excludeNodes(Objects nodes)
```

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

### Parameters:

- **nodes** - [in] A set of node identifiers that must be kept intact until the destruction of the class.

**addNodeType**

```java
public void addNodeType(int t)
```

Allows connectivity through nodes of the given type.

### Parameters:

- **t** - null

**setMaterializedAttribute**

```java
public void setMaterializedAttribute(String attributeName)
```
Creates a new common attribute type for all node types in the graph in order to store, persistently, the results related to the connected components found while executing this algorithm.

Whenever the user wants to retrieve the results, even when the graph has been closed and opened again, it is only necessary to create a new instance of the class ConnectedComponents indicating the graph and the name of the common attribute type which stores the results. This instance will have all the information related to the connected components found in the moment of the execution of the algorithm that stored this data.

It is possible to run the algorithm without specifying this parameter in order to avoid materializing the results of the execution.

**Parameters:**
- `attributeName` - [in] The name of the common attribute type for all node types in the graph which will store persistently the results generated by the execution of the algorithm.

**excludeEdges**

```java
public void excludeEdges(Objects edges)
```

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it’s less efficient than not allowing an edge type.

**Parameters:**
- `edges` - [in] A set of edge identifiers that must be kept intact until the destruction of the class.

**isClosed**

```java
public boolean isClosed()
```

Gets if Connectivity instance has been closed or not.

**Returns:**
- TRUE if the Connectivity instance has been closed, FALSE otherwise.

**See Also:**
- `close()`

**close**

```java
public void close()
```

Closes the Connectivity instance.

It must be called to ensure the integrity of all data.

**getConnectedComponents**

```java
public ConnectedComponents getConnectedComponents()
```

Returns the results generated by the execution of the algorithm.

These results contain information related to the connected components found as the number of different components, the set of nodes contained in each component or many other data.

**Returns:**
- Returns an instance of the class ConnectedComponents which contain information related to the connected components found.
## com.sparsity.sparksee.algorithms

### Context

**Class Context**

```java
java.lang.Object
   +---com.sparsity.sparksee.algorithms.Context
```

All Implemented Interfaces:

- Closeable

public class Context
   extends Object
   implements Closeable

Context class.

It provides a very similar functionality than the Traversal classes. The main difference is Context returns a resulting collection whereas Traversal provides an iterator behaviour.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

**Author:**
Sparsity Technologies http://www.sparsity-technologies.com

### Constructor Summary

| public | Context(Session s, long node) |
|        | Creates a new instance. |

### Method Summary

| void | addAllEdgeTypes(EdgesDirection d) |
|      | Allows for traversing all edge types of the graph. |

| void | addAllNodeTypes() |
|      | Allows for traversing all node types of the graph. |

| void | addEdgeType(int t, EdgesDirection d) |
|      | Allows for traversing edges of the given type. |

| void | addNodeType(int t) |
|      | Allows for traversing nodes of the given type. |

| void | close() |
|      | Closes the Context instance. |

| Objects | compute() |
|         | Gets the resulting collection of nodes. |

| static Objects | compute(Session s, long node, TypeList nodeTypes, TypeList edgeTypes, EdgesDirection dir, int maxhops, boolean include) |
|               | Helper method to easily compute a context from a node. |

| void | excludeEdges(Objects edges) |
|      | Set which edges can't be used. |
**void**  
**excludeNodes**(*Objects* nodes)  
Set which nodes can't be used.

**boolean**  
**isClosed**()  
Gets if Context instance has been closed or not.

**void**  
**setMaximumHops**(int maxhops, boolean include)  
Sets the maximum hops restriction.

**Methods inherited from class** java.lang.Object

close, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait,
Parameters:
   nodes - [in] A set of node identifiers that must be kept intact until the destruction of the class.

compute
public Objects compute()

    Gets the resulting collection of nodes.

    Returns:
    The resulting collection of nodes.

addAllEdgeTypes
public void addAllEdgeTypes(EdgesDirection d)

    Allows for traversing all edge types of the graph.

    Parameters:
    d - [in] Edge direction.

addNodeType
public void addNodeType(int t)

    Allows for traversing nodes of the given type.

    Parameters:
    t - null

compute
public static Objects compute(Session s,
                           long node,
                           TypeList nodeTypes,
                           TypeList edgeTypes,
                           EdgesDirection dir,
                           int maxhops,
                           boolean include)

    Helper method to easily compute a context from a node.

    Parameters:
    s - [in] Session to get the graph from and perform operation.
    node - [in] Node to start traversal from.
    nodeTypes - [in] Allowed node type list. NULL means all node types are allowed.
    edgeTypes - [in] Allowed edge type list. NULL means all edge types are allowed.
    dir - [in] Allowed direction for the allowed edge types.
    maxhops - [in] The maximum hops restriction. It must be positive or zero. Zero, the default value, means unlimited.
    include - [in] If TRUE, the resulting collection will include those nodes at distance less or equal than the given one, otherwise it will just include those nodes at distance equal than the given one. This parameter just makes sense if maxhops is different from 0; in that case it includes all nodes no matters the distance.
**close**

`public void close()`

Closes the Context instance.

It must be called to ensure the integrity of all data.

**addAllNodeTypes**

`public void addAllNodeTypes()`

Allows for traversing all node types of the graph.

**setMaximumHops**

`public void setMaximumHops(int maxhops, boolean include)`

Sets the maximum hops restriction.

All paths longer than the maximum hops restriction will be ignored.

**Parameters:**

- `maxhops` - [in] The maximum hops restriction. It must be positive or zero. Zero, the default value, means unlimited.
- `include` - [in] If TRUE, the resulting collection will include those nodes at distance less or equal than the given one, otherwise it will just include those nodes at distance equal than the given one. This parameter just makes sense if maxhops is different from 0; in that case it includes all nodes no matters the distance.

**excludeEdges**

`public void excludeEdges(Objects edges)`

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

**Parameters:**

- `edges` - [in] A set of edge identifiers that must be kept intact until the destruction of the class.

**isClosed**

`public boolean isClosed()`

Gets if Context instance has been closed or not.

**Returns:**

- TRUE if the Context instance has been closed, FALSE otherwise.

**See Also:**

`close()`
com.sparsity.sparksee.algorithms
Class ShortestPath

java.lang.Object
   +--com.sparsity.sparksee.algorithms.ShortestPath

All Implemented Interfaces:
   Closeable

Direct Known Subclasses:
   SinglePairShortestPath

public class ShortestPath
extends Object
implements Closeable

ShortestPath class.

Classes implementing this abstract class solve the shortest path problem in a graph.
The user must set which node and edge types can be used for the traversal.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author:
Sparsity Technologies http://www.sparsity-technologies.com

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>void addAllEdgeTypes(EdgesDirection d)</td>
</tr>
<tr>
<td>Allows for traversing all edge types of the graph.</td>
</tr>
<tr>
<td>void addAllNodeTypes()</td>
</tr>
<tr>
<td>Allows for traversing all node types of the graph.</td>
</tr>
<tr>
<td>void addEdgeType(int t, EdgesDirection d)</td>
</tr>
<tr>
<td>Allows for traversing edges of the given type.</td>
</tr>
<tr>
<td>void addNodeType(int t)</td>
</tr>
<tr>
<td>Allows for traversing nodes of the given type.</td>
</tr>
<tr>
<td>void close()</td>
</tr>
<tr>
<td>Closes the ShortestPath instance.</td>
</tr>
<tr>
<td>void excludeEdges(Objects edges)</td>
</tr>
<tr>
<td>Set which edges can't be used.</td>
</tr>
<tr>
<td>void excludeNodes(Objects nodes)</td>
</tr>
<tr>
<td>Set which nodes can't be used.</td>
</tr>
<tr>
<td>boolean isClosed()</td>
</tr>
<tr>
<td>Gets if ShortestPath instance has been closed or not.</td>
</tr>
<tr>
<td>void run()</td>
</tr>
<tr>
<td>Runs the algorithm.</td>
</tr>
</tbody>
</table>
### Methods

#### addEdgeType

```java
public void addEdgeType(int t, EdgesDirection d)
```

Allows for traversing edges of the given type.

**Parameters:**
- `t` - [in] Edge type.
- `d` - [in] Edge direction.

#### addAllNodeTypes

```java
public void addAllNodeTypes()
```

Allows for traversing all node types of the graph.

#### run

```java
public void run()
```

Runs the algorithm.

This method can only be called once.

#### excludeNodes

```java
public void excludeNodes(Objects nodes)
```

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

**Parameters:**
- `nodes` - [in] A set of node identifiers that must be kept intact until the destruction of the class.
addNodeType

public void addNodeType(int t)

  Allows for traversing nodes of the given type.

  Parameters:
  t - null

addAllEdgeTypes

public void addAllEdgeTypes(EdgesDirection d)

  Allows for traversing all edge types of the graph.

  Parameters:
  d - [in] Edge direction.

excludeEdges

public void excludeEdges(Objects edges)

  Set which edges can't be used.
  
  This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

  Parameters:
  edges - [in] A set of edge identifiers that must be kept intact until the destruction of the class.

setMaximumHops

public void setMaximumHops(int maxhops)

  Sets the maximum hops restriction.
  
  All paths longer than the maximum hops restriction will be ignored. m[in] The maximum hops restriction. It must be positive or zero. Zero, the default value, means unlimited.

  Parameters:
  maxhops - null

isClosed

public boolean isClosed()

  Gets if ShortestPath instance has been closed or not.

  Returns:
  TRUE if the ShortestPath instance has been closed, FALSE otherwise.

  See Also:
  close()
close

public void close()

Closes the ShortestPath instance.

It must be called to ensure the integrity of all data.
public class SinglePairShortestPath extends ShortestPath

SinglePairShortestPath class.

Classes implementing this abstract class solve the shortest path problem in a graph from a given source node and to a given destination node.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author:
Sparsity Technologies http://www.sparsity-technologies.com

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>void addAllEdgeTypes(EdgesDirection d)</td>
</tr>
<tr>
<td>Allows for traversing all edge types of the graph.</td>
</tr>
<tr>
<td>void addAllNodeTypes()</td>
</tr>
<tr>
<td>Allows for traversing all node types of the graph.</td>
</tr>
<tr>
<td>void addEdgeType(int t, EdgesDirection d)</td>
</tr>
<tr>
<td>Allows for traversing edges of the given type.</td>
</tr>
<tr>
<td>void addNodeType(int t)</td>
</tr>
<tr>
<td>Allows for traversing nodes of the given type.</td>
</tr>
<tr>
<td>void excludeEdges(Objects edges)</td>
</tr>
<tr>
<td>Set which edges can't be used.</td>
</tr>
<tr>
<td>void excludeNodes(Objects nodes)</td>
</tr>
<tr>
<td>Set which nodes can't be used.</td>
</tr>
<tr>
<td>boolean exists()</td>
</tr>
<tr>
<td>Returns TRUE If a path exists or FALSE otherwise.</td>
</tr>
<tr>
<td>double getCost()</td>
</tr>
<tr>
<td>Gets the cost of the shortest path.</td>
</tr>
<tr>
<td>OIDList getPathAsEdges()</td>
</tr>
<tr>
<td>Gets the shortest path between the source node and the destination node as an ordered set of edges.</td>
</tr>
</tbody>
</table>
OIDList getPathAsNodes()

Gets the shortest path between the source node and the destination node as an ordered set of nodes.

void run()

Runs the algorithm.

void setMaximumHops(int maxhops)

Sets the maximum hops restriction.

Methods inherited from class com.sparsity.sparksee.algorithms.ShortestPath

addAllEdgeTypes, addAllNodeTypes, addEdgeType, addNodeType, close, excludeEdges, excludeNodes, isClosed, run, setMaximumHops

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

Methods inherited from interface java.io.Closeable

close

Methods

exists

class com.sparsity.sparksee.algorithms.SinglePairShortestPath

public boolean exists()

Returns TRUE If a path exists or FALSE otherwise.

addEdgeType

class com.sparsity.sparksee.algorithms.SinglePairShortestPath

public void addEdgeType(int t, EdgesDirection d)

Allows for traversing edges of the given type.

Parameters:

t - [in] Edge type.
d - [in] Edge direction.

excludeNodes

class com.sparsity.sparksee.algorithms.SinglePairShortestPath

public void excludeNodes(Objects nodes)

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

Parameters:
run

```java
public void run()
{
    Runs the algorithm.
    This method can only be called once.
}
```

addAllEdgeTypes

```java
public void addAllEdgeTypes(EdgesDirection d)

    Allows for traversing all edge types of the graph.

Parameters:
    d - [in] Edge direction.
```

addNodeType

```java
public void addNodeType(int t)

    Allows for traversing nodes of the given type.

Parameters:
    t - null
```

getPathAsEdges

```java
public OIDList getPathAsEdges()

    Gets the shortest path between the source node and the destination node as an ordered set of edges.

Returns:
    Ordered set of edge identifiers.
```

setMaximumHops

```java
public void setMaximumHops(int maxhops)

    Sets the maximum hops restriction.
    All paths longer than the maximum hops restriction will be ignored.
    m[in] The maximum hops restriction. It must be positive or zero.
    Zero, the default value, means unlimited.

Parameters:
    maxhops - null
```

addAllNodeTypes

```java
public void addAllNodeTypes()
```
Allows for traversing all node types of the graph.

**getCost**

```java
public double getCost ()
```

Gets the cost of the shortest path.

The cost for unweighted algorithms is the number of hops of the shortest path. For weighted algorithms, the cost is the sum of the costs of the edges of the shortest path.

**Returns:**
The cost of the shortest path.

**getPathAsNodes**

```java
public OIDList getPathAsNodes ()
```

Gets the shortest path between the source node and the destination node as an ordered set of nodes.

**Returns:**
Ordered set of node identifiers.

**excludeEdges**

```java
public void excludeEdges (Objects edges)
```

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

**Parameters:**
edges - [in] A set of edge identifiers that must be kept intact until the destruction of the class.
**com.sparsity.sparksee.algorithms**

**Class SinglePairShortestPathBFS**

```java
public class SinglePairShortestPathBFS
extends SinglePairShortestPath
```

SinglePairShortestPathBFS class.

It solves the single-pair shortest path problem using a BFS-based implementation.

It is a unweighted algorithm, that is it assumes all edges have the same cost.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

**Author:**
Sparsity Technologies http://www.sparsity-technologies.com

---

**Constructor Summary**

<table>
<thead>
<tr>
<th>public</th>
<th>SinglePairShortestPathBFS(Session s, long src, long dst)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Creates a new instance.</td>
</tr>
</tbody>
</table>

**Method Summary**

<table>
<thead>
<tr>
<th>void</th>
<th>addAllEdgeTypes(EdgesDirection d)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Allows for traversing all edge types of the graph.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>addAllNodeTypes()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Allows for traversing all node types of the graph.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>addEdgeType(int t, EdgesDirection d)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Allows for traversing edges of the given type.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>addNodeType(int t)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Allows for traversing nodes of the given type.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>checkOnlyExistence()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set that only the path existence must be calculated and not the path itself.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>excludeEdges(Objects edges)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set which edges can’t be used.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>excludeNodes(Objects nodes)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set which nodes can’t be used.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>exists()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Returns TRUE if a path exists or FALSE otherwise.</td>
</tr>
</tbody>
</table>
### double `getCost()`  
Gets the cost of the shortest path.

### OIDList `getPathAsEdges()`  
Gets the shortest path between the source node and the destination node as an ordered set of edges.

### OIDList `getPathAsNodes()`  
Gets the shortest path between the source node and the destination node as an ordered set of nodes.

### void `run()`  
Executes the algorithm.

### void `setMaximumHops(int maxhops)`  
Sets the maximum hops restriction.

### Methods inherited from class `com.sparsity.sparksee.algorithms.SinglePairShortestPath`
- addAllEdgeTypes, addAllNodeTypes, addEdgeType, addNodeType, excludeEdges, excludeNodes, exists, getCost, getPathAsEdges, getPathAsNodes, run, setMaximumHops

### Methods inherited from class `com.sparsity.sparksee.algorithms.ShortestPath`
- addAllEdgeTypes, addAllNodeTypes, addEdgeType, addNodeType, close, excludeEdges, excludeNodes, isClosed, run, setMaximumHops

### Methods inherited from class `java.lang.Object`
- clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### Methods inherited from interface `java.io.Closeable`
- close

### Constructors

**SinglePairShortestPathBFS**

```java
public SinglePairShortestPathBFS(Session s, long src, long dst)
```

Creates a new instance.

**Parameters:**
- `s` - [in] Session to get the graph from and perform traversal.
- `src` - [in] Source node.
- `dst` - [in] Destination node.

### Methods
exists

public boolean exists()

    Returns TRUE If a path exists or FALSE otherwise.

addEdgeType

public void addEdgeType(int t,
    EdgesDirection d)

    Allows for traversing edges of the given type.

    Parameters:
        t - [in] Edge type.
        d - [in] Edge direction.

excludeNodes

public void excludeNodes(Objects nodes)

    Set which nodes can't be used.

    This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific
    nodes from allowed node types because it's less efficient than not allowing a node type.

    Parameters:
        nodes - [in] A set of node identifiers that must be kept intact until the destruction of the class.

getPathAsEdges

public OIDList getPathAsEdges()

    Gets the shortest path between the source node and the destination node as an ordered set of edges.

    Returns:
        Ordered set of edge identifiers.

getPathAsNodes

public OIDList getPathAsNodes()

    Gets the shortest path between the source node and the destination node as an ordered set of nodes.

    Returns:
        Ordered set of node identifiers.

addAllEdgeTypes

public void addAllEdgeTypes(EdgesDirection d)
Allows for traversing all edge types of the graph.

**Parameters:**
- d - [in] Edge direction.

---

**addNodeType**

public void **addNodeType** (int t)

Allows for traversing nodes of the given type.

**Parameters:**
- t - null

---

**setMaximumHops**

public void **setMaximumHops** (int maxhops)

Sets the maximum hops restriction.

All paths longer than the maximum hops restriction will be ignored. m[in] The maximum hops restriction. It must be positive or zero. Zero, the default value, means unlimited.

**Parameters:**
- maxhops - null

---

**getCost**

public double **getCost** ()

Gets the cost of the shortest path.

The cost is the number of hops of the shortest path.

**Returns:**
- The cost of the shortest path.

---

**addAllNodeTypes**

public void **addAllNodeTypes** ()

Allows for traversing all node types of the graph.

---

**run**

public void **run** ()

Executes the algorithm.

---

**checkOnlyExistence**

public void **checkOnlyExistence** ()
Set that only the path existence must be calculated and not the path itself.

That method should improve the performance of the algorithm, but a call to GetPathAsNodes or GetPathAsEdges will generate an exception even if the path exists.

**excludeEdges**

```java
public void excludeEdges(Objects edges)
```

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

**Parameters:**

- **edges** - [in] A set of edge identifiers that must be kept intact until the destruction of the class.
public class SinglePairShortestPathDijkstra extends SinglePairShortestPath

SinglePairShortestPathDijkstra class.

It solves the single-pair shortest path problem using a Dijkstra-based implementation.

It is a weighted algorithm, so it takes into account the cost of the edges to compute a minimum-cost shortest path. That is, the user may set for each edge type which attribute should be used to retrieve the cost of the edge. If no attribute is given for an edge type, this will assume the edge has a fixed cost (the default is 1). Only numerical attribute can be set as weight attributes (that is Long, Integer or Double attributes are allowed).

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author: Sparsity Technologies http://www.sparsity-technologies.com

---

### Constructor Summary

| public | SinglePairShortestPathDijkstra(Session s, long src, long dst) |

Create a new instance.

---

### Method Summary

| public | addAllEdgeTypes(EdgesDirection d) |

Allows for traversing all edge types of the graph.

| void   | addAllNodeTypes() |

Allows for traversing all node types of the graph.

| void   | addEdgeType(int t, EdgesDirection d) |

Allows for traversing edges of the given type.

| void   | addNodeType(int t) |

Allows for traversing nodes of the given type.

| void   | addWeightedEdgeType(int t, EdgesDirection d, int attr) |

Allows for traversing edges of the given type using the given attribute as the weight.

| void   | excludeEdges(Objects edges) |

Set which edges can't be used.

| void   | excludeNodes(Objects nodes) |

Set which nodes can't be used.
boolean exists()
    Returns TRUE if a path exists or FALSE otherwise.

double getCost()
    Gets the cost of the shortest path.

OIDList getPathAsEdges()
    Gets the shortest path between the source node and the destination node as an ordered set of edges.

OIDList getPathAsNodes()
    Gets the shortest path between the source node and the destination node as an ordered set of nodes.

void run()
    Executes the algorithm.

void setMaximumHops(int maxhops)
    Sets the maximum hops restriction.

void setUnweightedEdgeCost(double weight)
    Sets the weight assigned to the unweighted edges.

Methods inherited from class com.sparsity.sparksee.algorithms.SinglePairShortestPath
addAllEdgeTypes, addAllNodeTypes, addEdgeType, addNodeType, excludeEdges, excludeNodes, exists, getCost, getPathAsEdges, getPathAsNodes, run, setMaximumHops

Methods inherited from class com.sparsity.sparksee.algorithms.ShortestPath
addAllEdgeTypes, addAllNodeTypes, addEdgeType, addNodeType, close, excludeEdges, excludeNodes, isClosed, run, setMaximumHops

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

Methods inherited from interface java.io.Closeable
close

Constructors

SinglePairShortestPathDijkstra

public SinglePairShortestPathDijkstra(Session s,
    Long src,
    long dst)

Creates a new instance.

Parameters:
    s - [in] Session to get the graph from and perform traversal.
    src - [in] Source node.
dst - [dst] Destination node.

Methods

exists

public boolean exists()

Returns TRUE If a path exists or FALSE otherwise.

addEdgeType

public void addEdgeType(int t, EdgesDirection d)

Allows for traversing edges of the given type.

Parameters:
  t - [in] Edge type.
  d - [in] Edge direction.

excludeNodes

public void excludeNodes(Objects nodes)

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

Parameters:
  nodes - [in] A set of node identifiers that must be kept intact until the destruction of the class.

getPathAsEdges

public OIDList getPathAsEdges()

Gets the shortest path between the source node and the destination node as an ordered set of edges.

Returns:
  Ordered set of edge identifiers.

getPathAsNodes

public OIDList getPathAsNodes()

Gets the shortest path between the source node and the destination node as an ordered set of nodes.

Returns:
  Ordered set of node identifiers.
addWeightedEdgeType

public void addWeightedEdgeType(int t,
        EdgesDirection d,
        int attr)

Allows for traversing edges of the given type using the given attribute as the weight.

Parameters:
    t - [in] Edge type.
    d - [in] Edge direction.
    attr - [in] Attribute to be used as the weight. It must be a global attribute or an attribute of the given edge type.

setUnweightedEdgeCost

public void setUnweightedEdgeCost(double weight)

Sets the weight assigned to the unweighted edges.

All the edges from the types added without an explicit weight attribute will get this weight. The default weight for this edges is 1.

Parameters:
    weight - null

addAllEdgeTypes

public void addAllEdgeTypes(EdgesDirection d)

Allows for traversing all edge types of the graph.

Parameters:
    d - [in] Edge direction.

addNodeType

public void addNodeType(int t)

Allows for traversing nodes of the given type.

Parameters:
    t - null

setMaximumHops

public void setMaximumHops(int maxhops)

Sets the maximum hops restriction.

All paths longer than the maximum hops restriction will be ignored. m[in] The maximum hops restriction. It must be positive or zero. Zero, the default value, means unlimited.

Parameters:
    maxhops - null
**getCost**

```java
public double getCost()
```

Gets the cost of the shortest path.

The cost is the sum of the weights of the edges in the shortest path.

**Returns:**

The cost of the shortest path.

---

**addAllNodeTypes**

```java
public void addAllNodeTypes()
```

Allows for traversing all node types of the graph.

---

**run**

```java
public void run()
```

Executes the algorithm.

---

**excludeEdges**

```java
public void excludeEdges(Objects edges)
```

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

**Parameters:**

edges - [in] A set of edge identifiers that must be kept intact until the destruction of the class.
public class StrongConnectivity extends Connectivity

StrongConnectivity class.

Any class implementing this abstract class can be used to solve the problem of finding strongly connected components in a directed graph.

It consists in finding components where every pair (u, v) of nodes contained in it has a path from u to v using the specified direction for each edge type.

It is possible to set some restrictions after constructing a new instance of this class and before running it in order to limit the results.

After the execution, we can retrieve the results stored in an instance of the ConnectedComponents class using the GetConnectedComponents method.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author: Sparsity Technologies http://www.sparsity-technologies.com

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void addAllEdgeTypes(EdgesDirection d)</td>
<td>Allows connectivity through all edge types of the graph.</td>
</tr>
<tr>
<td>void addAllNodeTypes()</td>
<td>Allows connectivity through all node types of the graph.</td>
</tr>
<tr>
<td>void addEdgeType(int t, EdgesDirection d)</td>
<td>Allows connectivity through edges of the given type.</td>
</tr>
<tr>
<td>void addNodeType(int t)</td>
<td>Allows connectivity through nodes of the given type.</td>
</tr>
<tr>
<td>void excludeEdges(Objects edges)</td>
<td>Set which edges can't be used.</td>
</tr>
<tr>
<td>void excludeNodes(Objects nodes)</td>
<td>Set which nodes can't be used.</td>
</tr>
<tr>
<td>ConnectedComponents getConnectedComponents()</td>
<td>Returns the results generated by the execution of the algorithm.</td>
</tr>
</tbody>
</table>
void run()

Runs the algorithm in order to find the connected components.

void setMaterializedAttribute(String attributeName)

Creates a new common attribute type for all node types in the graph in order to store, persistently, the results related to the connected components found while executing this algorithm.

Methods inherited from class com.sparsity.sparksee.algorithms.Connectivity

addAllNodeTypes, addNodeType, close, excludeEdges, excludeNodes, getConnectedComponents, isClosed, run, setMaterializedAttribute

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface java.io.Closeable

close

Methods

addEdgeType

public void addEdgeType(int t,
    EdgesDirection d)

Allows connectivity through edges of the given type.

Parameters:
  t - [in] Edge type.
  d - [in] Edge direction.

addAllNodeTypes

public void addAllNodeTypes()

Allows connectivity through all node types of the graph.

run

public void run()

Runs the algorithm in order to find the connected components.

This method can be called only once.

excludeNodes

public void excludeNodes(Objects nodes)
Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

**Parameters:**
- `nodes` - [in] A set of node identifiers that must be kept intact until the destruction of the class.

---

### addNodeType

```java
public void addNodeType(int t)
```

Allows connectivity through nodes of the given type.

**Parameters:**
- `t` - null

---

### addAllEdgeTypes

```java
public void addAllEdgeTypes(EdgesDirection d)
```

Allows connectivity through all edge types of the graph.

**Parameters:**
- `d` - [in] Edge direction.

---

### setMaterializedAttribute

```java
public void setMaterializedAttribute(String attributeName)
```

Creates a new common attribute type for all node types in the graph in order to store, persistently, the results related to the connected components found while executing this algorithm.

Whenever the user wants to retrieve the results, even when the graph has been closed and opened again, it is only necessary to create a new instance of the class ConnectedComponents indicating the graph and the name of the common attribute type which stores the results. This instance will have all the information related to the connected components found in the moment of the execution of the algorithm that stored this data.

It is possible to run the algorithm without specifying this parameter in order to avoid materializing the results of the execution.

**Parameters:**
- `attributeName` - [in] The name of the common attribute type for all node types in the graph which will store persistently the results generated by the execution of the algorithm.

---

### excludeEdges

```java
public void excludeEdges(Object edges)
```

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

**Parameters:**
- `edges` - [in] A set of edge identifiers that must be kept intact until the destruction of the class.
getConnectedComponents

public ConnectedComponents getConnectedComponents()

    Returns the results generated by the execution of the algorithm.

    These results contain information related to the connected components found as the number of different components, the
    set of nodes contained in each component or many other data.

    Returns:
    Returns an instance of the class ConnectedComponents which contain information related to the connected
    components found.
public class StrongConnectivityGabow
extends StrongConnectivity

This class can be used to solve the problem of finding strongly connected components in a directed graph.

It consists in finding components where every pair (u,v) of nodes contained in it has a path from u to v using the specified direction for each edge type. This implementation is based on the Gabow algorithm.

It is possible to set some restrictions after constructing a new instance of this class and before running it in order to limit the results.

After the execution, we can retrieve the results stored in an instance of the ConnectedComponents class using the 
GetConnectedComponents method.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author:
Sparsity Technologies http://www.sparsity-technologies.com

Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public StrongConnectivityGabow(Session s)</td>
<td>Creates a new instance of StrongConnectivityGabow.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void addAllEdgeTypes(EdgesDirection d)</td>
<td>Allows connectivity through all edge types of the graph.</td>
</tr>
<tr>
<td>void addAllNodeTypes()</td>
<td>Allows connectivity through all node types of the graph.</td>
</tr>
<tr>
<td>void addEdgeType(int t, EdgesDirection d)</td>
<td>Allows connectivity through edges of the given type.</td>
</tr>
<tr>
<td>void addNodeType(int t)</td>
<td>Allows connectivity through nodes of the given type.</td>
</tr>
<tr>
<td>void excludeEdges(Objects edges)</td>
<td>Set which edges can't be used.</td>
</tr>
<tr>
<td>void excludeNodes(Objects nodes)</td>
<td>Set which nodes can't be used.</td>
</tr>
<tr>
<td>Method Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ConnectedComponents</td>
<td>getConnectedComponents()</td>
</tr>
<tr>
<td></td>
<td>Returns the results generated by the execution of the algorithm.</td>
</tr>
<tr>
<td>void</td>
<td>run()</td>
</tr>
<tr>
<td></td>
<td>Executes the algorithm.</td>
</tr>
<tr>
<td>void</td>
<td>setMaterializedAttribute(String attributeName)</td>
</tr>
<tr>
<td></td>
<td>Creates a new common attribute type for all node types in the graph in order</td>
</tr>
<tr>
<td></td>
<td>to store, persistently, the results related to the connected components</td>
</tr>
<tr>
<td></td>
<td>found while executing this algorithm.</td>
</tr>
</tbody>
</table>

Methods inherited from class `com.sparsity.sparksee.algorithms.StrongConnectivity`

- `addAllEdgeTypes`, `addAllNodeTypes`, `addEdgeType`, `addNodeType`, `excludeEdges`, `excludeNodes`, `getConnectedComponents`, `run`, `setMaterializedAttribute`  

Methods inherited from class `com.sparsity.sparksee.algorithms.Connectivity`

- `addAllNodeTypes`, `addNodeType`, `close`, `excludeEdges`, `excludeNodes`, `getConnectedComponents`, `isClosed`, `run`, `setMaterializedAttribute`  

Methods inherited from class `java.lang.Object`  

- `clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`  

Methods inherited from interface `java.io.Closeable`

- `close`  

Constructors

**StrongConnectivityGabow**

```java
public StrongConnectivityGabow(Session s)
```

Creates a new instance of StrongConnectivityGabow.

After creating this instance is required to indicate the set of edge types and the set of node types which will be navigated through while traversing the graph in order to find the strong connected components.

**Parameters:**

- `s` - [in] Session to get the graph from and calculate the connectivity

Methods

**addEdgeType**

```java
public void addEdgeType(int t, EdgesDirection d)
```

Allows connectivity through edges of the given type.

**Parameters:**

- `t` - [in] Edge type.
d - [in] Edge direction.

### addAllNodeTypes

```java
public void addAllNodeTypes()
```

Allows connectivity through all node types of the graph.

### excludeNodes

```java
public void excludeNodes(Objects nodes)
```

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

**Parameters:**

- `nodes` - [in] A set of node identifiers that must be kept intact until the destruction of the class.

### run

```java
public void run()
```

Executes the algorithm.

### addNodeType

```java
public void addNodeType(int t)
```

Allows connectivity through nodes of the given type.

**Parameters:**

- `t` - null

### addAllEdgeTypes

```java
public void addAllEdgeTypes(EdgesDirection d)
```

Allows connectivity through all edge types of the graph.

**Parameters:**

- `d` - [in] Edge direction.

### setMaterializedAttribute

```java
public void setMaterializedAttribute(String attributeName)
```
Creates a new common attribute type for all node types in the graph in order to store, persistently, the results related to the connected components found while executing this algorithm.

Whenever the user wants to retrieve the results, even when the graph has been closed and opened again, it is only necessary to create a new instance of the class ConnectedComponents indicating the graph and the name of the common attribute type which stores the results. This instance will have all the information related to the connected components found in the moment of the execution of the algorithm that stored this data.

It is possible to run the algorithm without specifying this parameter in order to avoid materializing the results of the execution.

**Parameters:**

- `attributeName` - [in] The name of the common attribute type for all node types in the graph which will store persistently the results generated by the execution of the algorithm.

### excludeEdges

**public void excludeEdges(Objects edges)**

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

**Parameters:**

- `edges` - [in] A set of edge identifiers that must be kept intact until the destruction of the class.

### getConnectedComponents

**public ConnectedComponents getConnectedComponents()**

Returns the results generated by the execution of the algorithm.

These results contain information related to the connected components found as the number of different components, the set of nodes contained in each component or many other data.

**Returns:**

Returns an instance of the class ConnectedComponents which contain information related to the connected components found.
com.sparsity.sparksee.algorithms

Class Traversal

generic

+com.sparsity.sparksee.algorithms.Traversal

All Implemented Interfaces:
  Closeable

Direct Known Subclasses:
  TraversalDFS, TraversalBFS

public class Traversal
extends Object
implements Closeable

Traversal class.

Any class implementing this abstract class can be used to traverse a graph.

Once the instance has been created and the allowed node and edge types has been set, it can be used as an iterator, retrieving the
next object identifier of the traversal until there are no more.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author: Sparsity Technologies http://www.sparsity-technologies.com

Method Summary

<table>
<thead>
<tr>
<th>void</th>
<th>addAllEdgeTypes(EdgesDirection d)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Allows for traversing all edge types of the graph.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>addAllNodeTypes()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Allows for traversing all node types of the graph.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>addEdgeType(int t, EdgesDirection d)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Allows for traversing edges of the given type.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>addNodeType(int t)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Allows for traversing nodes of the given type.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>close()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Closes the Traversal instance.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>excludeEdges(Objects edges)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set which edges can't be used.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>excludeNodes(Objects nodes)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set which nodes can't be used.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>getCurrentDepth()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Returns the depth of the current node.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>hasNext()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets if there are more objects to be traversed.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>boolean isClosed()</td>
<td>Gets if Traversal instance has been closed or not.</td>
</tr>
<tr>
<td>long next()</td>
<td>Gets the next object of the traversal.</td>
</tr>
<tr>
<td>void setMaximumHops(int maxhops)</td>
<td>Sets the maximum hops restriction.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

- clone
- equals
- finalize
- getClass
- hashCode
- notify
- notifyAll
- toString
- wait

Methods inherited from interface java.io.Closeable

- close

Methods

**addEdgeType**

public void **addEdgeType** (int t, EdgesDirection d)

Allows for traversing edges of the given type.

**Parameters:**

- t - [in] Edge type.
- d - [in] Edge direction.

**hasNext**

public boolean **hasNext** ()

Gets if there are more objects to be traversed.

**Returns:**

- TRUE if there are more objects, FALSE otherwise.

**excludeNodes**

public void **excludeNodes** (Objects nodes)

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific
nodes from allowed node types because it's less efficient than not allowing a node type.

**Parameters:**

- nodes - [in] A set of node identifiers that must be kept intact until the destruction of the class.
addAllEdgeTypes
public void addAllEdgeTypes(EdgesDirection d)

Allows for traversing all edge types of the graph.

Parameters:
  d - [in] Edge direction.

addNodeType
public void addNodeType(int t)

Allows for traversing nodes of the given type.

Parameters:
  t - null

setMaximumHops
public void setMaximumHops(int maxhops)

Sets the maximum hops restriction.
All paths longer than the maximum hops restriction will be ignored.

Parameters:
  maxhops - [in] The maximum hops restriction. It must be positive or zero. Zero, the default value, means unlimited.

currentDepth
public int getCurrentDepth()

Returns the depth of the current node.
That is, it returns the depth of the node returned in the last call to Next().

Returns:
  The depth of the current node.

close
public void close()

Closes the Traversal instance.
It must be called to ensure the integrity of all data.

addAllNodeTypes
public void addAllNodeTypes()

Allows for traversing all node types of the graph.
**next**

public long **next**()

Gets the next object of the traversal.

**Returns:**
A node or edge identifier.

**excludeEdges**

public void **excludeEdges**(Objects edges)

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

**Parameters:**
edges - [in] A set of edge identifiers that must be kept intact until the destruction of the class.

**isClosed**

public boolean **isClosed**()

Gets if Traversal instance has been closed or not.

**Returns:**
TRUE if the Traversal instance has been closed, FALSE otherwise.

**See Also:**
**close()**
com.sparsity.sparksee.algorithms
Class TraversalBFS

public class TraversalBFS extends Traversal

Breadth-First Search implementation of Traversal.
Starting from a source node, it visits all its neighbors at distance 1, then all its neighbors at distance 2, and so on.
Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author:
Sparsity Technologies http://www.sparsity-technologies.com

Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Method Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>public TraversalBFS(Session s, long node)</td>
<td>Creates a new instance.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Method Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>void addAllEdgeTypes(EdgesDirection d)</td>
<td>Allows for traversing all edge types of the graph.</td>
</tr>
<tr>
<td>void addAllNodeTypes()</td>
<td>Allows for traversing all node types of the graph.</td>
</tr>
<tr>
<td>void addEdgeType(int t, EdgesDirection d)</td>
<td>Allows for traversing edges of the given type.</td>
</tr>
<tr>
<td>void addNodeType(int t)</td>
<td>Allows for traversing nodes of the given type.</td>
</tr>
<tr>
<td>void excludeEdges(Objects edges)</td>
<td>Set which edges can't be used.</td>
</tr>
<tr>
<td>void excludeNodes(Objects nodes)</td>
<td>Set which nodes can't be used.</td>
</tr>
<tr>
<td>int getCurrentDepth()</td>
<td>Returns the depth of the current node.</td>
</tr>
<tr>
<td>boolean hasNext()</td>
<td>Gets if there are more objects to be traversed.</td>
</tr>
<tr>
<td>long next()</td>
<td>Gets the next object of the traversal.</td>
</tr>
</tbody>
</table>
void setMaximumHops(int maxhops)
Sets the maximum hops restriction.

Methods inherited from class com.sparsity.sparksee.algorithms.Traversal
addAllEdgeTypes, addAllNodeTypes, addEdgeType, addNodeType, close, excludeEdges,
excludeNodes, getCurrentDepth, hasNext, isClosed, next, setMaximumHops

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface java.io.Closeable
close

Constructors

TraversalBFS
public TraversalBFS(Session s,
long node)

Creates a new instance.

Parameters:
    s - [in] Session to get the graph from and perform traversal.
    node - [in] Node to start traversal from.

Methods

addEdgeType
public void addEdgeType(int t,
EdgesDirection d)

Allows for traversing edges of the given type.

Parameters:
    t - [in] Edge type.
    d - [in] Edge direction.

addAllNodeTypes
public void addAllNodeTypes()

Allows for traversing all node types of the graph.
hasNext

public boolean hasNext()

Gets if there are more objects to be traversed.

Returns:
TRUE if there are more objects, FALSE otherwise.

excludeNodes

public void excludeNodes(Objects nodes)

Set which nodes can't be used.
This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

Parameters:
    nodes - [in] A set of node identifiers that must be kept intact until the destruction of the class.

next

public long next()

Gets the next object of the traversal.

Returns:
A node or edge identifier.

addNodeType

public void addNodeType(int t)

Allows for traversing nodes of the given type.

Parameters:
    t - null

addAllEdgeTypes

public void addAllEdgeTypes(EdgesDirection d)

Allows for traversing all edge types of the graph.

Parameters:
    d - [in] Edge direction.

gGetCurrentDepth

public int getGetCurrentDepth()
Returns the depth of the current node.
That is, it returns the depth of the node returned in the last call to Next().

**Returns:**
The depth of the current node.

---

**setMaximumHops**

```java
public void setMaximumHops(int maxhops)
```

Sets the maximum hops restriction.
All paths longer than the maximum hops restriction will be ignored.

**Parameters:**
- `maxhops` - [in] The maximum hops restriction. It must be positive or zero. Zero, the default value, means unlimited.

---

**excludeEdges**

```java
public void excludeEdges(Objects edges)
```

Set which edges can't be used.
This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

**Parameters:**
- `edges` - [in] A set of edge identifiers that must be kept intact until the destruction of the class.
com.sparsity.sparksee.algorithms
Class TraversalDFS

java.lang.Object
   +-com.sparsity.sparksee.algorithms.Traversal
   |+
|   +-com.sparsity.sparksee.algorithms.TraversalDFS

All Implemented Interfaces:
   Closeable

public class TraversalDFS
extends Traversal

Depth-First Search (DFS) implementation of Traversal.

Starting from a source or root node, it visits as far as possible along each branch before backtracking.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author:
   Sparsity Technologies http://www.sparsity-technologies.com

Constructor Summary

| public TraversalDFS(Session s, long node) |
| Creates a new instance. |

Method Summary

| void addAllEdgeTypes(EdgesDirection d) |
| Allows for traversing all edge types of the graph. |
| void addAllNodeTypes() |
| Allows for traversing all node types of the graph. |
| void addEdgeType(int t, EdgesDirection d) |
| Allows for traversing edges of the given type. |
| void addNodeType(int t) |
| Allows for traversing nodes of the given type. |
| void excludeEdges(Objects edges) |
| Set which edges can't be used. |
| void excludeNodes(Objects nodes) |
| Set which nodes can't be used. |
| int getCurrentDepth() |
| Returns the depth of the current node. |
| boolean hasNext() |
| Gets if there are more objects to be traversed. |
| long next() |
| Gets the next object of the traversal. |
void **setMaximumHops**(int maxhops)

Sets the maximum hops restriction.

Methods inherited from class `com.sparsity.sparksee.algorithms.Traversal`

`addAllEdgeTypes`, `addAllNodeTypes`, `addEdgeType`, `addNodeType`, `close`, `excludeEdges`, `excludeNodes`, `getCurrentDepth`, `hasNext`, `isClosed`, `next`, `setMaximumHops`

Methods inherited from class `java.lang.Object`

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Methods inherited from interface `java.io.Closeable`

`close`

### Constructors

**TraversalDFS**

```java
public TraversalDFS(Session s,
long node)
```

Creates a new instance.

**Parameters:**

- `s`: [in] Session to get the graph from and perform traversal.
- `node`: [in] Node to start traversal from.

### Methods

#### addEdgeType

```java
public void addEdgeType(int t,
EdgesDirection d)
```

Allows for traversing edges of the given type.

**Parameters:**

- `t`: [in] Edge type.
- `d`: [in] Edge direction.

#### addAllNodeTypes

```java
public void addAllNodeTypes()
```

Allows for traversing all node types of the graph.
hasNext

public boolean hasNext()

  Gets if there are more objects to be traversed.

  Returns:
  
  TRUE if there are more objects, FALSE otherwise.

excludeNodes

public void excludeNodes(Objects nodes)

  Set which nodes can't be used.

  This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

  Parameters:
  
  nodes - [in] A set of node identifiers that must be kept intact until the destruction of the class.

next

public long next()

  Gets the next object of the traversal.

  Returns:
  A node or edge identifier.

addNodeType

public void addNodeType(int t)

  Allows for traversing nodes of the given type.

  Parameters:
  
  t - null

addAllEdgeTypes

public void addAllEdgeTypes(EdgesDirection d)

  Allows for traversing all edge types of the graph.

  Parameters:
  
  d - [in] Edge direction.

gGetCurrentDepth

public int getGetCurrentDepth()
Returns the depth of the current node.
That is, it returns the depth of the node returned in the last call to Next().

**Returns:**
The depth of the current node.

### setMaximumHops

```java
public void setMaximumHops(int maxhops)
```

Sets the maximum hops restriction.
All paths longer than the maximum hops restriction will be ignored.

**Parameters:**

- `maxhops` - [in] The maximum hops restriction. It must be positive or zero. Zero, the default value, means unlimited.

### excludeEdges

```java
public void excludeEdges(Objects edges)
```

Set which edges can't be used.
This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

**Parameters:**

- `edges` - [in] A set of edge identifiers that must be kept intact until the destruction of the class.
public class WeakConnectivity extends Connectivity

WeakConnectivity class.

Any class implementing this abstract class can be used to solve the problem of finding weakly connected components in an undirected graph or in a directed graph which will be considered as an undirected one.

It consists in finding components where every pair (u,v) of nodes contained in it has a path from u to v and from v to u.

It is possible to set some restrictions after constructing a new instance of this class and before running it in order to limit the results.

After the execution, we can retrieve the results stored in an instance of the ConnectedComponents class using the getConnectedComponents method.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author:
Sparsity Technologies http://www.sparsity-technologies.com

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>void</strong> addAllEdgeTypes()</td>
</tr>
<tr>
<td>Allows connectivity through all edge types of the graph.</td>
</tr>
<tr>
<td><strong>void</strong> addAllNodeTypes()</td>
</tr>
<tr>
<td>Allows connectivity through all node types of the graph.</td>
</tr>
<tr>
<td><strong>void</strong> addEdgeType(int t)</td>
</tr>
<tr>
<td>Allows connectivity through edges of the given type.</td>
</tr>
<tr>
<td><strong>void</strong> addNodeType(int t)</td>
</tr>
<tr>
<td>Allows connectivity through nodes of the given type.</td>
</tr>
<tr>
<td><strong>void</strong> excludeEdges(Objects edges)</td>
</tr>
<tr>
<td>Set which edges can't be used.</td>
</tr>
<tr>
<td><strong>void</strong> excludeNodes(Objects nodes)</td>
</tr>
<tr>
<td>Set which nodes can't be used.</td>
</tr>
<tr>
<td><strong>ConnectedComponents</strong> getConnectedComponents()</td>
</tr>
<tr>
<td>Returns the results generated by the execution of the algorithm.</td>
</tr>
</tbody>
</table>
### void run()
Runs the algorithm in order to find the connected components.

### void setMaterializedAttribute(String attributeName)
Creates a new common attribute type for all node types in the graph in order to store, persistently, the results related to the connected components found while executing this algorithm.

**Methods inherited from class** [com.sparsity.sparksee.algorithms.Connectivity](#)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>addAllNodeTypes, addNodeType, close, excludeEdges, excludeNodes, getConnectedComponents, isClosed, run, setMaterializedAttribute</td>
<td></td>
</tr>
</tbody>
</table>

**Methods inherited from class** [java.lang.Object](#)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait</td>
<td></td>
</tr>
</tbody>
</table>

**Methods inherited from interface** [java.io.Closeable](#)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>close</td>
<td></td>
</tr>
</tbody>
</table>

## Methods

### addEdgeType
**public void** addEdgeType(int t)

Allows connectivity through edges of the given type.

In a weak connectivity the edges can be used in Any direction. d[|n| Edge direction.

**Parameters:**
- t - [in] Edge type.

### addAllNodeTypes
**public void** addAllNodeTypes()

Allows connectivity through all node types of the graph.

### addAllEdgeTypes
**public void** addAllEdgeTypes()

Allows connectivity through all edge types of the graph.

In a weak connectivity the edges can be used in Any direction. d[|n| Edge direction.

### run
**public void** run()
Runs the algorithm in order to find the connected components.
This method can be called only once.

**excludeNodes**

```java
public void excludeNodes(Objects nodes)
```

Set which nodes can't be used.
This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

**Parameters:**
- `nodes` - [in] A set of node identifiers that must be kept intact until the destruction of the class.

**addNodeType**

```java
public void addNodeType(int t)
```

Allows connectivity through nodes of the given type.

**Parameters:**
- `t` - null

**setMaterializedAttribute**

```java
public void setMaterializedAttribute(String attributeName)
```

Creates a new common attribute type for all node types in the graph in order to store, persistently, the results related to the connected components found while executing this algorithm.

Whenever the user wants to retrieve the results, even when the graph has been closed and opened again, it is only necessary to create a new instance of the class ConnectedComponents indicating the graph and the name of the common attribute type which stores the results. This instance will have all the information related to the connected components found in the moment of the execution of the algorithm that stored this data.

It is possible to run the algorithm without specifying this parameter in order to avoid materializing the results of the execution.

**Parameters:**
- `attributeName` - [in] The name of the common attribute type for all node types in the graph which will store persistently the results generated by the execution of the algorithm.

**excludeEdges**

```java
public void excludeEdges(Objects edges)
```

Set which edges can't be used.
This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

**Parameters:**
- `edges` - [in] A set of edge identifiers that must be kept intact until the destruction of the class.

**getConnectedComponents**

```java
public ConnectedComponents getConnectedComponents()
```

Page 53 of 288
Returns the results generated by the execution of the algorithm.

These results contain information related to the connected components found as the number of different components, the set of nodes contained in each component or many other data.

**Returns:**

Returns an instance of the class ConnectedComponents which contain information related to the connected components found.
public class WeakConnectivityDFS
extends WeakConnectivity

WeakConnectivityDFS class.

This class can be used to solve the problem of finding weakly connected components in an undirected graph or in a directed graph which will be considered as an undirected one.

It consists in finding components where every pair (u,v) of nodes contained in it has a path from u to v and from v to u. This implementation is based on the Depth-First Search (DFS) algorithm.

It is possible to set some restrictions after constructing a new instance of this class and before running it in order to limit the results.

After the execution, we can retrieve the results stored in an instance of the ConnectedComponents class using the getConnectedComponents method.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author:
Sparsity Technologies http://www.sparsity-technologies.com

<table>
<thead>
<tr>
<th>Constructor Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>public WeakConnectivityDFS(Session s)</td>
</tr>
<tr>
<td>Creates a new instance of WeakConnectivityDFS.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>void addAllEdgeTypes()</td>
</tr>
<tr>
<td>Allows connectivity through all edge types of the graph.</td>
</tr>
<tr>
<td>void addAllNodeTypes()</td>
</tr>
<tr>
<td>Allows connectivity through all node types of the graph.</td>
</tr>
<tr>
<td>void addEdgeType(int t)</td>
</tr>
<tr>
<td>Allows connectivity through edges of the given type.</td>
</tr>
<tr>
<td>void addNodeType(int t)</td>
</tr>
<tr>
<td>Allows connectivity through nodes of the given type.</td>
</tr>
<tr>
<td>void excludeEdges(Objects edges)</td>
</tr>
<tr>
<td>Set which edges can't be used.</td>
</tr>
</tbody>
</table>
### void `excludeNodes(Objects nodes)`
Set which nodes can't be used.

### ConnectedComponents `getConnectedComponents()`
Returns the results generated by the execution of the algorithm.

### void `run()`
Executes the algorithm.

### void `setMaterializedAttribute(String attributeName)`
Creates a new common attribute type for all node types in the graph in order to store, persistently, the results related to the connected components found while executing this algorithm.

---

### Methods inherited from class `com.sparsity.sparksee.algorithms.WeakConnectivity`

- `addAllEdgeTypes`, `addAllNodeTypes`, `addEdgeType`, `addNodeType`, `excludeEdges`, `excludeNodes`, `getConnectedComponents`, `run`, `setMaterializedAttribute`

### Methods inherited from class `com.sparsity.sparksee.algorithms.Connectivity`

- `addAllNodeTypes`, `addNodeType`, `close`, `excludeEdges`, `excludeNodes`, `getConnectedComponents`, `isClosed`, `run`, `setMaterializedAttribute`

### Methods inherited from class `java.lang.Object`

- `clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

### Methods inherited from interface `java.io.Closeable`

- `close`

---

### Constructors

#### WeakConnectivityDFS

```java
public WeakConnectivityDFS(Session s)
```

Creates a new instance of WeakConnectivityDFS.

After creating this instance is required to indicate the set of edge types and the set of node types which will be navigated through while traversing the graph in order to find the weak connected components.

**Parameters:**

- `s` - [in] Session to get the graph from and calculate the connectivity

---

### Methods

#### addEdgeType

```java
public void addEdgeType(int t)
```

Allows connectivity through edges of the given type.

In a weak connectivity the edges can be used in Any direction. `d[in]` Edge direction.
Parameters:
  t - [in] Edge type.

**addAllNodeTypes**

```java
public void addAllNodeTypes()
```

Allows connectivity through all node types of the graph.

**addAllEdgeTypes**

```java
public void addAllEdgeTypes()
```

Allows connectivity through all edge types of the graph.

In a weak connectivity the edges can be used in Any direction. d[in] Edge direction.

**excludeNodes**

```java
public void excludeNodes(Objects nodes)
```

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

**Parameters:**
  nodes - [in] A set of node identifiers that must be kept intact until the destruction of the class.

**run**

```java
public void run()
```

Executes the algorithm.

**addNodeType**

```java
public void addNodeType(int t)
```

Allows connectivity through nodes of the given type.

**Parameters:**
  t - null

**setMaterializedAttribute**

```java
public void setMaterializedAttribute(String attributeName)
```
Creates a new common attribute type for all node types in the graph in order to store, persistently, the results related to the connected components found while executing this algorithm.

Whenever the user wants to retrieve the results, even when the graph has been closed and opened again, it is only necessary to create a new instance of the class ConnectedComponents indicating the graph and the name of the common attribute type which stores the results. This instance will have all the information related to the connected components found in the moment of the execution of the algorithm that stored this data.

It is possible to run the algorithm without specifying this parameter in order to avoid materializing the results of the execution.

**Parameters:**

- **attributeName** - [in] The name of the common attribute type for all node types in the graph which will store persistently the results generated by the execution of the algorithm.

---

**excludeEdges**

```java
public void excludeEdges(Objects edges)
```

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

**Parameters:**

- **edges** - [in] A set of edge identifiers that must be kept intact until the destruction of the class.

---

**getConnectedComponents**

```java
public ConnectedComponents getConnectedComponents()
```

Returns the results generated by the execution of the algorithm.

These results contain information related to the connected components found as the number of different components, the set of nodes contained in each component or many other data.

**Returns:**

- Returns an instance of the class ConnectedComponents which contain information related to the connected components found.
Package

com.sparsity.sparksee.gdb
public class Attribute
extends Object

Attribute data class.

It contains information about an attribute.

Author:
Sparsity Technologies http://www.sparsity-technologies.com

Field Summary

<table>
<thead>
<tr>
<th>public static</th>
<th>InvalidAttribute</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Invalid attribute identifier constant.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>long</th>
<th>getCount()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the number of non-NULL values.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DataType</th>
<th>getDataType()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the data type.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>getId()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the Sparksee attribute identifier.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AttributeKind</th>
<th>getKind()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the attribute kind.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>getName()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the unique attribute name.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>long</th>
<th>getSize()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the number of different values.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>getType()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the Sparksee type identifier.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>isSessionAttribute()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Check if it's a session attribute or a persistent one.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields
InvalidAttribute
public static int InvalidAttribute

Invalid attribute identifier constant.

Methods

getKind
public AttributeKind getKind()

Gets the attribute kind.

Returns:
The AttributeKind.

g getCount
public long getCount()

Gets the number of non-NULL values.

Returns:
The number of non-NULL values.

isSessionAttribute
public boolean isSessionAttribute()

Check if it's a session attribute or a persistent one.

Returns:
True if it's a session attribute, or false otherwise.

getSize
public long getSize()

Gets the number of different values.

Returns:
The number of different values.

getTypeId
public int getTypeId()
Gets the Sparksee type identifier.

**Returns:**
The Sparksee type identifier.

---

**getDataType**

```java
public DataType getDataType()
```

Gets the data type.

**Returns:**
The `DataType`.

---

**getId**

```java
public int getId()
```

Gets the Sparksee attribute identifier.

**Returns:**
The Sparksee attribute identifier.

---

**getName**

```java
public String getName()
```

Gets the unique attribute name.

**Returns:**
The unique attribute name.
public final class AttributeKind extends Enum

Attribute kind enumeration.

It determines the indexing-capabilities of an attribute.

**Author:**
Sparsity Technologies http://www.sparsity-technologies.com

<table>
<thead>
<tr>
<th>Field Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static final</td>
</tr>
<tr>
<td>Basic attribute (non indexed attribute).</td>
</tr>
<tr>
<td>public static final</td>
</tr>
<tr>
<td>Indexed attribute.</td>
</tr>
<tr>
<td>public static final</td>
</tr>
<tr>
<td>Unique attribute (indexed + unique restriction).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>static AttributeKind.valueOf(String name)</td>
</tr>
<tr>
<td>static AttributeKind[] values()</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Enum

clone, compareTo, equals, finalize, getDeclaringClass, hashCode, name, ordinal, toString, valueOf

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

Methods inherited from interface java.lang.Comparable

compareTo
Fields

Basic

public static final com.sparsity.sparksee.gdb.AttributeKind Basic

Basic attribute (non indexed attribute).

Indexed

public static final com.sparsity.sparksee.gdb.AttributeKind Indexed

Indexed attribute.

Unique

public static final com.sparsity.sparksee.gdb.AttributeKind Unique

Unique attribute (indexed + unique restriction).

Unique restriction sets two objects cannot have the same value for an attribute but NULL.

Methods

values

public static AttributeKind[] values()

valueOf

public static AttributeKind valueOf(String name)
public class AttributeList
extends Object
implements Iterable

Sparksee attribute identifier list.

It stores a Sparksee attribute identifier list.

Use AttributeListIterator to access all elements into this collection.

Author:
Sparsity Technologies http://www.sparsity-technologies.com

Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public AttributeList(Collection col)</td>
<td>Creates a new instance from an integer collection.</td>
</tr>
<tr>
<td>public AttributeList()</td>
<td>Constructor.</td>
</tr>
<tr>
<td>public AttributeList(int[] list)</td>
<td>Creates a new instance from an integer array.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void add(int attr)</td>
<td>Adds a Sparksee attribute identifier at the end of the list.</td>
</tr>
<tr>
<td>void clear()</td>
<td>Clears the list.</td>
</tr>
<tr>
<td>int count()</td>
<td>Number of elements in the list.</td>
</tr>
<tr>
<td>AttributeListIterator iterator()</td>
<td>Gets a new AttributeListIterator.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface java.lang.Iterable

iterator
Constructors

AttributeList
public AttributeList(Collection col)

    Creates a new instance from an integer collection.

    Parameters:
    col - Collection to initialize the instance.

AttributeList
public AttributeList()

    Constructor.
    This creates an empty list.

AttributeList
public AttributeList(int[] list)

    Creates a new instance from an integer array.

    Parameters:
    list - Integer array to initialize the instance.

Methods

clear
public void clear()

    Clears the list.

iterator
public AttributeListIterator iterator()

    Gets a new AttributeListIterator.

    Returns:
    AttributeListIterator instance.

count
public int count()
Number of elements in the list.

**Returns:**
Number of elements in the list.

### add

```java
public void add(int attr)
```

Adds a Sparksee attribute identifier at the end of the list.

**Parameters:**

- `attr` - [in] Sparksee attribute identifier.
public class AttributeListIterator extends Object
implements Iterator

AttributeList iterator class.

Iterator to traverse all the Sparksee attribute identifier into a AttributeList instance.

Author:
Sparsity Technologies http://www.sparsity-technologies.com

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hasNext</td>
<td>boolean</td>
<td>Gets if there are more elements.</td>
</tr>
<tr>
<td>next</td>
<td>Integer</td>
<td>See nextAttribute().</td>
</tr>
<tr>
<td>nextAttribute</td>
<td>int</td>
<td>Gets the next element.</td>
</tr>
<tr>
<td>remove</td>
<td>void</td>
<td>Operation not supported.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

Methods inherited from interface java.util.Iterator

hasNext, next, remove

Methods

hasNext

public boolean hasNext()

Gets if there are more elements.
Returns:
TRUE if there are more elements, FALSE otherwise.

remove
public void remove()

  Operation not supported.

next
public Integer next()

  See nextAttribute().

nextAttribute
public int nextAttribute()

  Gets the next element.
public class AttributeStatistics
extends Object

Attribute statistics class.

It contains statistic data about an attribute.

Some fields are valid just for numerical attributes and others just for string attributes. Also, some statistics are considered BASIC because computing them do not require to traverse all the different values of the attribute. For each getter method the documentation tells if the statistic is BASIC or not. See the Graph class method getAttributeStatistics or check out the SPARKSEE User Manual for more details on this.

Author:
Sparsity Technologies http://www.sparsity-technologies.com

#### Method Summary

<table>
<thead>
<tr>
<th>return type</th>
<th>method name</th>
<th>documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>double</td>
<td>getAvgLengthString()</td>
<td>Gets the average length.</td>
</tr>
<tr>
<td>long</td>
<td>getDistinct()</td>
<td>Gets the number of distinct values (BASIC statistics).</td>
</tr>
<tr>
<td>Value</td>
<td>getMax()</td>
<td>Gets the maximum existing value (BASIC statistics).</td>
</tr>
<tr>
<td>int</td>
<td>getMaxLengthString()</td>
<td>Gets the maximum length.</td>
</tr>
<tr>
<td>double</td>
<td>getMean()</td>
<td>Gets the mean or average.</td>
</tr>
<tr>
<td>double</td>
<td>getMedian()</td>
<td>Gets the median.</td>
</tr>
<tr>
<td>Value</td>
<td>getMin()</td>
<td>Gets the minimum existing value (BASIC statistics).</td>
</tr>
<tr>
<td>int</td>
<td>getMinLengthString()</td>
<td>Gets the minimum length.</td>
</tr>
<tr>
<td>Value</td>
<td>getMode()</td>
<td>Gets the mode.</td>
</tr>
<tr>
<td>long</td>
<td>getModeCount()</td>
<td>Gets the number of objects with a Value equal to the mode.</td>
</tr>
<tr>
<td>long</td>
<td>getNull()</td>
<td>Gets the number of objects NULL a Value (BASIC statistics).</td>
</tr>
<tr>
<td>Type</td>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>long</td>
<td>getTotal()</td>
<td>Gets the number of objects with a non-NULL Value (BASIC statistic).</td>
</tr>
<tr>
<td>double</td>
<td>getVariance()</td>
<td>Gets the variance.</td>
</tr>
</tbody>
</table>

**Methods inherited from class** java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

### Methods

**getMin**

```java
public Value getMin()
```

Gets the minimum existing value (BASIC statistics).

**Returns:**

The minimum existing value.

**getMinLengthString**

```java
public int getMinLengthString()
```

Gets the minimum length.

If the attribute is not an string attribute, it just returns 0.

**Returns:**

The minimum length.

**getVariance**

```java
public double getVariance()
```

Gets the variance.

It is computed just for numerical attributes.

**Returns:**

The variance.

**getMode**

```java
public Value getMode()
```

Gets the mode.

Mode: Most frequent Value.

**Returns:**

The mode.
**getNull**

```java
public long getNull()
```

Gets the number of objects NULL a Value (BASIC statistics).

**Returns:**
The number of objects NULL a Value.

---

**getDistinct**

```java
public long getDistinct()
```

Gets the number of distinct values (BASIC statistics).

**Returns:**
The number of distinct values.

---

**getMean**

```java
public double getMean()
```

Gets the mean or average.

Mean or average: Sum of all Values divided by the number of observations.

It is computed just for numerical attributes.

**Returns:**
The mean.

---

**getMax**

```java
public Value getMax()
```

Gets the maximum existing value (BASIC statistics).

**Returns:**
The maximum existing value.

---

**getMedian**

```java
public double getMedian()
```

Gets the median.

Median: Middle value that separates the higher half from the lower.

If a < b < c, then the median of the list \{a, b, c\} is b, and if a < b < c < d, then the median of the list \{a, b, c, d\} is the mean of b and c, i.e. it is \((b + c)/2\)

It is computed just for numerical attributes.

**Returns:**
The median.

getTotal
public long getTotal()

Gets the number of objects with a non-NULL Value (BASIC statistic).

Returns:
The number of objects with a non-NULL Value.

getMaxLengthString
public int getMaxLengthString()

Gets the maximum length.
If the attribute is not an string attribute, it just returns 0.

Returns:
The maximum length.

getAvgLengthString
public double getAvgLengthString()

Gets the average length.
If the attribute is not an string attribute, it just returns 0.

Returns:
The average length.

getModeCount
public long getModeCount()

Gets the number of objects with a Value equal to the mode.

Returns:
The number of objects with a Value equal to the mode.
com.sparsity.sparksee.gdb
Class BooleanList

java.lang.Object
   +- com.sparsity.sparksee.gdb.BooleanList

All Implemented Interfaces:
   Iterable

public class BooleanList
extends Object
implements Iterable

Boolean list.
It stores a Boolean list.

Use BooleanListIterator to access all elements into this collection.

Author:
  Sparsity Technologies http://www.sparsity-technologies.com

Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
<td>BooleanList(Collection col)</td>
</tr>
<tr>
<td></td>
<td>Creates a new instance from a boolean collection.</td>
</tr>
<tr>
<td>public</td>
<td>BooleanList(boolean[] list)</td>
</tr>
<tr>
<td></td>
<td>Creates a new instance from a boolean array.</td>
</tr>
<tr>
<td>public</td>
<td>BooleanList()</td>
</tr>
<tr>
<td></td>
<td>Constructor.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>add(boolean b)</td>
</tr>
<tr>
<td></td>
<td>Adds a Boolean at the end of the list.</td>
</tr>
<tr>
<td>void</td>
<td>clear()</td>
</tr>
<tr>
<td></td>
<td>Clears the list.</td>
</tr>
<tr>
<td>int</td>
<td>count()</td>
</tr>
<tr>
<td></td>
<td>Number of elements in the list.</td>
</tr>
<tr>
<td>BooleanListIterator</td>
<td>iterator()</td>
</tr>
<tr>
<td></td>
<td>Gets a new BooleanListIterator.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface java.lang.Iterable

iterator
Constructors

**BooleanList**

class BooleanList {

  **public BooleanList** (Collection col)

  Creates a new instance from a boolean collection.

  Parameters:
  - col - Collection to initialize the instance.

  **public BooleanList** (boolean[] list)

  Creates a new instance from a boolean array.

  Parameters:
  - list - Boolean array to initialize the instance.

  **public BooleanList** ()

  Constructor.
  This creates an empty list.

Methods

**clear**

  public void clear()

  Clears the list.

**add**

  public void add(boolean b)

  Adds a Boolean at the end of the list.

  Parameters:
  - b - [in] Boolean.

**iterator**

  public BooleanListIterator iterator()
Gets a new BooleanListIterator.

**Returns:**
BooleanListIterator instance.

count

```java
class BooleanList
{
    public int count()
    {
        Number of elements in the list.
        
        **Returns:**
        Number of elements in the list.
    }
}
```
com.sparsity.sparksee.gdb
Class BooleanListIterator

public class BooleanListIterator
extends Object
implements Iterator

BooleanList iterator class.
Iterator to traverse all the strings into a BooleanList instance.

Author:
Sparsity Technologies http://www.sparsity-technologies.com

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>hasNext()</td>
</tr>
<tr>
<td>Boolean</td>
<td>next()</td>
</tr>
<tr>
<td>boolean</td>
<td>nextBoolean()</td>
</tr>
<tr>
<td>void</td>
<td>remove()</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

Methods inherited from interface java.util.Iterator
hasNext, next, remove

Methods

hasNext

public boolean hasNext() | Gets if there are more elements.
Returns:
TRUE if there are more elements, FALSE otherwise.

**remove**
public void remove()

Operation not supported.

**next**
public Boolean next()

See nextBoolean().

**nextBoolean**
public boolean nextBoolean()

Gets the next element.
com.sparsity.sparksee.gdb
Class Condition

public final class Condition
extends Enum

Condition operators enumeration.

It is mainly used in the attribute-based graph select operations.

Author:
Sparsity Technologies http://www.sparsity-technologies.com

Field Summary

| public static final | Between | In range operator condition ([x,y]). |
| public static final | Equal   | Equal condition (==).               |
| public static final | GreaterEqual | Greater or equal condition (>=).   |
| public static final | GreaterThan | Greater than condition (>).        |
| public static final | LessEqual  | Less or equal condition (<=).       |
| public static final | LessThan   | Less than condition (<).            |
| public static final | Like       | Substring condition.                |
| public static final | LikeNoCase | Substring (no case sensitive) condition. |
| public static final | NotEqual   | Not equal condition (!=).           |
| public static final | RegExp     | Regular expression condition.       |

Method Summary

static Condition | String name | valueOf
static Condition[] values()

Methods inherited from class java.lang.Enum

clone, compareTo, equals, finalize, getDeclaringClass, hashCode, name, ordinal, toString, valueOf

Methods inherited from class java.lang.Object

close, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

Methods inherited from interface java.lang.Comparable

clazoTo

Fields

Equal

public static final com.sparsity.sparksee.gdb.Condition Equal

Equal condition (==).

Null values can be used together with this condition to retrieve all objects having a null value for an attribute.

GreaterEqual

public static final com.sparsity.sparksee.gdb.Condition GreaterEqual

Greater or equal condition (>=).

Null values cannot be used together with this condition.

GreaterThan

public static final com.sparsity.sparksee.gdb.Condition GreaterThan

Greater than condition (>).

Null values cannot be used together with this condition.

LessEqual

public static final com.sparsity.sparksee.gdb.Condition LessEqual

Less or equal condition (<=).

Null values cannot be used together with this condition.

LessThan

public static final com.sparsity.sparksee.gdb.Condition LessThan

Less than condition (<).

Null values cannot be used together with this condition.
**NotEqual**

**public static final com.sparsity.sparksee.gdb.Condition NotEqual**

Not equal condition (!=).

Null values can be used together with this condition to retrieve all objects having a non-null value for an attribute.

**Like**

**public static final com.sparsity.sparksee.gdb.Condition Like**

Substring condition.

Null values cannot be used together with this condition.

This condition can just be used together with String values. It allows for searching substrings (case sensitive). Ex:

'AAABBCCCD' Like 'BBB' returns TRUE

'AAABBCCCD' Like 'bbb' returns FALSE

'AAABBCCCD' Like 'E' returns FALSE

**LikeNoCase**

**public static final com.sparsity.sparksee.gdb.Condition LikeNoCase**

Substring (no case sensitive) condition.

Null values cannot be used together with this condition.

This condition can just be used together with String values. It allows for searching substrings (no case sensitive). Ex:

'AAABBCCCD' LikeNoCase 'BBB' returns TRUE

'AAABBCCCD' LikeNoCase 'bbb' returns TRUE

'AAABBCCCD' LikeNoCase 'E' returns FALSE

**Between**

**public static final com.sparsity.sparksee.gdb.Condition Between**

In range operator condition ([x,y]).

Null values cannot be used together with this condition.

**RegExp**

**public static final com.sparsity.sparksee.gdb.Condition RegExp**

Regular expression condition.

Null values cannot be used together with this condition.

This condition can just be used together with String values.

Regular expression format conforms most of the POSIX Extended Regular Expressions so it is case sensitive.

See the 'Regular expressions' section in the 'SPARKSEE User Manual' for details.
Methods

values
public static Condition[] values()

valueOf
public static Condition.valueOf(String name)
public class Database
extends Object
implements Closeable

Database class.

All the data of the Database is stored into a persistent file which just can be created or open through a Sparksee instance.

Also, all the manipulation of a Database must be done by means of a Session which can be initiated from a Database instance.

Multiple Databases do not share the memory, that is there is no negotiation among them. In those cases, memory must be prefixed for each Database. To do that, use the SPARKSEEConfig.

Author:
Sparsity Technologies http://www.sparity-technologies.com

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>void</strong> close()</td>
</tr>
<tr>
<td>Closes the Database instance.</td>
</tr>
<tr>
<td><strong>String</strong> getAlias()</td>
</tr>
<tr>
<td>Gets the alias of the Database.</td>
</tr>
<tr>
<td><strong>String</strong> getPath()</td>
</tr>
<tr>
<td>Gets the path of the Database.</td>
</tr>
<tr>
<td><strong>void</strong> getStatistics(DatabaseStatistics stats)</td>
</tr>
<tr>
<td>Gets Database statistics.</td>
</tr>
<tr>
<td><strong>boolean</strong> isClosed()</td>
</tr>
<tr>
<td>Gets if Database instance has been closed or not.</td>
</tr>
<tr>
<td><strong>Session</strong> newSession()</td>
</tr>
<tr>
<td>Creates a new Session.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

Methods inherited from interface java.io.Closeable

close
## Methods

### getAlias

```java
public String getAlias()
```

Gets the alias of the Database.

**Returns:**

The alias of the Database.

### getPath

```java
public String getPath()
```

Gets the path of the Database.

**Returns:**

The path of the Database.

### newSession

```java
public Session newSession()
```

Creates a new Session.

### isClosed

```java
public boolean isClosed()
```

Gets if Database instance has been closed or not.

**Returns:**

TRUE if the Database instance has been closed, FALSE otherwise.

**See Also:**

`close()`

### close

```java
public void close()
```

Closes the Database instance.

It must be called to ensure the integrity of all data.

### getStatistics

```java
public void getStatistics(DatabaseStatistics stats)
```

Page 84 of 288
Gets Database statistics.

**Parameters:**

```plaintext
stats - [out] The DatabaseStatistics instance.
```
public class DatabaseStatistics
extends Object

Database statistics.

Author:
Sparsity Technologies http://www.sparsity-technologies.com

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>long</td>
<td>getCache()</td>
<td>Gets cache size in KBytes.</td>
</tr>
<tr>
<td>long</td>
<td>getData()</td>
<td>Gets database size in KBytes.</td>
</tr>
<tr>
<td>long</td>
<td>getRead()</td>
<td>Gets total read data in KBytes.</td>
</tr>
<tr>
<td>long</td>
<td>getSessions()</td>
<td>Gets the number of sessions.</td>
</tr>
<tr>
<td>long</td>
<td>getTemp()</td>
<td>Gets temporary storage file size in KBytes.</td>
</tr>
<tr>
<td>long</td>
<td>getWrite()</td>
<td>Gets total written data in KBytes.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

Methods

gWrite

public long gWrite()

    Gets total written data in KBytes.

Returns:
    Total read written in KBytes.
getSessions
public long getSessions()

    Gets the number of sessions.

    Returns:
    The number of sessions.

data
public long getData()

    Gets database size in KBytes.

    Returns:
    Database size in KBytes.

temp
public long getTemp()

    Gets temporary storage file size in KBytes.

    Returns:
    Temporary storage file size in KBytes.

read
public long getRead()

    Gets total read data in KBytes.

    Returns:
    Total read data in KBytes.

cache
public long getCache()

    Gets cache size in KBytes.

    Returns:
    Cache size in KBytes.
public final class **DataType**
extends Enum

data type (domain) enumeration.

**Author:**
Sparsity Technologies http://www.sparsity-technologies.com

### Field Summary

<table>
<thead>
<tr>
<th>public static final</th>
<th>Boolean</th>
<th>Boolean data type.</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static final</td>
<td>Double</td>
<td>64-bit signed double data type.</td>
</tr>
<tr>
<td>public static final</td>
<td>Integer</td>
<td>32-bit signed integer data type.</td>
</tr>
<tr>
<td>public static final</td>
<td>Long</td>
<td>64-bit signed integer data type.</td>
</tr>
<tr>
<td>public static final</td>
<td>OID</td>
<td>Object identifier (OID) data type.</td>
</tr>
<tr>
<td>public static final</td>
<td>String</td>
<td>Unicode string data type.</td>
</tr>
<tr>
<td>public static final</td>
<td>Text</td>
<td>Large unicode character object (CLOB) data type.</td>
</tr>
<tr>
<td>public static final</td>
<td>Timestamp</td>
<td>Distance from Epoch (UTC) time in milliseconds precision.</td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>static <strong>DataType</strong></th>
<th>valueOf(String name)</th>
</tr>
</thead>
<tbody>
<tr>
<td>static <strong>DataType[]</strong></td>
<td>values()</td>
</tr>
</tbody>
</table>

Methods inherited from class **java.lang.Enum**
clone, compareTo, equals, finalize, getDeclaringClass, hashCode, name, ordinal, toString, valueOf

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

Methods inherited from interface java.lang.Comparable
compareTo

Fields

Boolean
public static final com.sparsity.sparksee.gdb.DataType Boolean
Boolean data type.

Integer
public static final com.sparsity.sparksee.gdb.DataType Integer
32-bit signed integer data type.

Long
public static final com.sparsity.sparksee.gdb.DataType Long
64-bit signed integer data type.

Double
public static final com.sparsity.sparksee.gdb.DataType Double
64-bit signed double data type.

Timestamp
public static final com.sparsity.sparksee.gdb.DataType Timestamp
Distance from Epoch (UTC) time in milliseconds precision.
It just works properly with timestamps in the range ['1970-01-01 00:00:01' UTC, '2038-01-19 03:14:07' UTC].
String
public static final com.sparsity.sparksee.gdb.DataType String

  Unicode string data type.
  2048 characters maximum length.

Text
public static final com.sparsity.sparksee.gdb.DataType Text

  Large unicode character object (CLOB) data type.

OID
public static final com.sparsity.sparksee.gdb.DataType OID

  Object identifier (OID) data type.

Methods

values
public static DataType[] values()

valueOf
public static DataType valueOf(String name)
com.sparsity.sparksee.gdb
Class DefaultExport

default implementation for ExportManager class.
It uses the default values from GraphExport, NodeExport and EdgeExport to export all node and edge types.

Author:
Sparsity Technologies http://www.sparsity-technologies.com

Constructor Summary

<table>
<thead>
<tr>
<th>public DefaultExport ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creates a new instance.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>boolean enableType (int type)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default implementation of the ExportManager class method EnableType.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean getEdge (long edge, EdgeExport edgeExport)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default implementation of the ExportManager class method GetEdge.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean getEdgeType (int type, EdgeExport edgeExport)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default implementation of the ExportManager class method GetEdgeType.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean getGraph (GraphExport graphExport)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default implementation of the ExportManager class method GetGraph.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean getNode (long node, NodeExport nodeExport)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default implementation of the ExportManager class method GetNode.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean getNodeType (int type, NodeExport nodeExport)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default implementation of the ExportManager class method GetNodeType.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void prepare (Graph graph)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default implementation of the ExportManager class method Prepare.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void release ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default implementation of the ExportManager class method Release.</td>
</tr>
</tbody>
</table>

Methods inherited from class com.sparsity.sparksee.gdb.ExportManager

| enableType, getEdge, getEdgeType, getGraph, getNode, getNodeType, prepare, release |

Methods inherited from class java.lang.Object
Constructors

DefaultExport

public DefaultExport()

Creates a new instance.

Methods

enableType

public boolean enableType(int type)

Default implementation of the ExportManager class method EnableType.
This enables all node and edge types to be exported.

Parameters:
  type - null

Returns:
  TRUE.

getEdge

public boolean getEdge(long edge,
  EdgeExport edgeExport)

Default implementation of the ExportManager class method GetEdge.
This sets the default EdgeExport values and sets the OID as the label. Also, it exports the edge as directed just if the edge is directed.

Parameters:
  edge - null
  edgeExport - null

Returns:
  TRUE.

graph

public boolean getGraph(GraphExport graphExport)

Default implementation of the ExportManager class method GetGraph.
This sets the default GraphExport values and "Graph" as the label.

Parameters:
  graphExport - null
**getEdgeType**

```java
public boolean getEdgeType(int type,
EdgeExport edgeExport)
```

Default implementation of the ExportManager class method GetEdgeType.

This sets de default EdgeExport values.

**Parameters:**
- type - null
- edgeExport - null

**getNodeType**

```java
public boolean getNodeType(int type,
NodeExport nodeExport)
```

Default implementation of the ExportManager class method GetNodeType.

This sets de default NodeExport values.

**Parameters:**
- type - null
- nodeExport - null

**release**

```java
public void release()
```

Default implementation of the ExportManager class method Release.

**getNode**

```java
public boolean getNode(long node,
NodeExport nodeExport)
```

Default implementation of the ExportManager class method GetNode.

This sets the default NodeExport values and sets the OID as the label.

**Parameters:**
- node - null
- nodeExport - null

**Returns:**
TRUE.

**prepare**

```java
public void prepare(Graph graph)
```

Default implementation of the ExportManager class method Prepare.

**Parameters:**
graph - null
com.sparsity.sparksee.gdb
Class EdgeData

java.lang.Object
   +-com.sparsity.sparksee.gdb.EdgeData

public class EdgeData
extends Object

Edge data class.

It stores the tail and the head of an edge instance.

In case of undirected edges, the tail and the head are just the two ends of the edge.

Author: Sparsity Technologies http://www.sparsity-technologies.com

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>long</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>long</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>long</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

<table>
<thead>
<tr>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>--------------------------------------</td>
</tr>
<tr>
<td><strong>getHead</strong></td>
</tr>
<tr>
<td>public long getHead()</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Gets the head of the edge.</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Returns:</strong></td>
</tr>
<tr>
<td>The Sparksee edge identifier of the head of the edge.</td>
</tr>
<tr>
<td>--------------------------------------</td>
</tr>
<tr>
<td>--------------------------------------</td>
</tr>
<tr>
<td><strong>getTail</strong></td>
</tr>
<tr>
<td>public long getTail()</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Gets the tail of the edge.</td>
</tr>
</tbody>
</table>
getEdge  
    public long getEdge()  
    
    Gets the edge identifier.  

    Returns:  
        The Sparksee edge identifier.
public class **EdgeExport**
extends Object

Stores edge exporting values.

Some properties may be ignored depending on the exportation type.

Default values are:
Label: "" (empty string).
As directed: TRUE.
Color: 13882323 (OxD3D3D3, Light gray).
Label color: 0 (Ox000000, Black).
Width: 5px.
Font size: 10.

**Author:**
Sparsity Technologies http://www.sparsity-technologies.com

---

**Constructor Summary**

<table>
<thead>
<tr>
<th>Public Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
<td><strong>EdgeExport()</strong></td>
<td>Creates a new instance.</td>
</tr>
</tbody>
</table>

---

**Method Summary**

<table>
<thead>
<tr>
<th>Return Type</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td><strong>asDirected()</strong></td>
<td>Gets if the edge should be managed as directed.</td>
</tr>
<tr>
<td><strong>java.awt.Color</strong></td>
<td><strong>getColor()</strong></td>
<td>Gets the color of the edge.</td>
</tr>
<tr>
<td>int</td>
<td><strong>getColorRGB()</strong></td>
<td>Gets the edge color.</td>
</tr>
<tr>
<td>int</td>
<td><strong>getFontSize()</strong></td>
<td>Gets the edge label font size.</td>
</tr>
<tr>
<td><strong>String</strong></td>
<td><strong>getLabel()</strong></td>
<td>Gets the edge label.</td>
</tr>
<tr>
<td><strong>java.awt.Color</strong></td>
<td><strong>getLabelColor()</strong></td>
<td>Gets the color of the label.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>int getLabelColorRGB()</td>
<td>Gets the edge label color.</td>
<td></td>
</tr>
<tr>
<td>int getWidth()</td>
<td>Gets the edge width.</td>
<td></td>
</tr>
<tr>
<td>void setAsDirected(boolean b)</td>
<td>Sets if the edge should be managed as directed.</td>
<td></td>
</tr>
<tr>
<td>void setColor(java.awt.Color c)</td>
<td>Sets the color of the edge.</td>
<td></td>
</tr>
<tr>
<td>void setColorRGB(int c)</td>
<td>Sets the edge color.</td>
<td></td>
</tr>
<tr>
<td>void setDefaults()</td>
<td>Sets to default values.</td>
<td></td>
</tr>
<tr>
<td>void setFontSize(int s)</td>
<td>Sets the edge label font size.</td>
<td></td>
</tr>
<tr>
<td>void setLabel(String l)</td>
<td>Sets the edge label.</td>
<td></td>
</tr>
<tr>
<td>void setLabelColor(java.awt.Color c)</td>
<td>Sets the color of the label.</td>
<td></td>
</tr>
<tr>
<td>void setLabelColorRGB(int c)</td>
<td>Sets the edge label color.</td>
<td></td>
</tr>
<tr>
<td>void setWidth(int w)</td>
<td>Sets the edge width.</td>
<td></td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object
clonen, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait,
wait

Constructors

**EdgeExport**

public **EdgeExport**()

Creates a new instance.

Methods

**getColor**

public java.awt.Color **getColor**()

Gets the color of the edge.
setLabelColorRGB
public void setLabelColorRGB(int c)

Sets the edge label color.

Parameters:
c - [in] The edge label color.

getFontSize
public int getFontSize()

Gets the edge label font size.

Returns:
The edge label font size.

setWidth
public void setWidth(int w)

Sets the edge width.

Parameters:
w - [in] The edge width.

setDefaults
public void setDefaults()

Sets to default values.

colorRGB
public int getColorRGB()

Gets the edge color.

Returns:
The edge color.

setFontSize
public void setFontSize(int s)
Sets the edge label font size.

**Parameters:**
- s - [in] The edge label font size.

---

**getAsDirected**

```java
public void setAsDirected(boolean b)
```

Sets if the edge should be managed as directed.

**Parameters:**
- b - [in] If TRUE, use as directed, otherwise use as undirected.

---

**getLabelColorRGB**

```java
public int getLabelColorRGB()
```

Gets the edge label color.

**Returns:**
- The edge label color.

---

**getWidth**

```java
public int getWidth()
```

Gets the edge width.

**Returns:**
- The edge width.

---

**getLabelColor**

```java
public java.awt.Color getLabelColor()
```

Gets the color of the label.

---

**setColor**

```java
public void setColor(java.awt.Color c)
```

Sets the color of the edge.

**Parameters:**
- c - New value.
setLabel

class com.sparsity.sparksee.gdb.EdgeExport

public void setLabel(String l)

Sets the edge label.

Parameters:

l - [in] The edge label.

asDirected

public boolean asDirected()

Gets if the edge should be managed as directed.

TRUE is the default value. If TRUE, use as directed, otherwise use as undirected.

Returns:

The edge direction.

setColorRGB

public void setColorRGB(int c)

Sets the edge color.

Parameters:

c - [in] The edge color.

getLabel

public String getLabel()

Gets the edge label.

Returns:

The edge label.

setLabelColor

public void setLabelColor(java.awt.Color c)

Sets the color of the label.

Parameters:

c - New value.
com.sparsity.sparksee.gdb
Class EdgesDirection

java.lang.Object
   |-java.lang.Enum
      |-com.sparsity.sparksee.gdb.EdgesDirection

All Implemented Interfaces:
   Serializable, Comparable

public final class EdgesDirection
extends Enum

Edges direction enumeration.

Author:
Sparsity Technologies http://www.sparsity-technologies.com

Field Summary

<table>
<thead>
<tr>
<th>public static final</th>
<th>Any</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>In-going or out-going edges.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public static final</th>
<th>Ingoing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>In-going edges.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public static final</th>
<th>Outgoing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Out-going edges.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>static EdgesDirection</th>
<th>valueOf(String name)</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>static EdgesDirection[]</th>
<th>values()</th>
</tr>
</thead>
</table>

Methods inherited from class java.lang.Enum

clon, compareTo, equals, finalize, getDeclaringClass, hashCode, name, ordinal, toString, valueOf

Methods inherited from class java.lang.Object

clon, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

Methods inherited from interface java.lang.Comparable

compareTo

Fields
**Ingoing**

public static final com.sparsity.sparksee.gdb.EdgesDirection Ingoing

In-going edges.

**Outgoing**

public static final com.sparsity.sparksee.gdb.EdgesDirection Outgoing

Out-going edges.

**Any**

public static final com.sparsity.sparksee.gdb.EdgesDirection Any

In-going or out-going edges.

**Methods**

**values**

public static EdgesDirection[] values()

**valueOf**

public static EdgesDirection valueOf(String name)
public class ExportManager
extends Object

Defines how to export a graph to an external format.

This is an interface which must be implemented by the user. While the export proces, a call for each node or edge type and node or edge object is done to get how to export that element.

It is possible to export a Graph to a different format. Nowadays, available formats are defined in the ExportType enum.

Author: Sparsity Technologies http://www.sparsity-technologies.com

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>boolean enableType(int type)</code></td>
<td>Gets whether a node or edge type must be exported or not.</td>
</tr>
<tr>
<td><code>boolean getEdge(long edge, EdgeExport edgeExport)</code></td>
<td>Gets the edge export definition for the given edge.</td>
</tr>
<tr>
<td><code>boolean getEdgeType(int type, EdgeExport edgeExport)</code></td>
<td>Gets the default node export definition for the given edge type.</td>
</tr>
<tr>
<td><code>boolean getGraph(GraphExport graphExport)</code></td>
<td>Gets the graph export definition.</td>
</tr>
<tr>
<td><code>boolean getNode(long node, NodeExport nodeExport)</code></td>
<td>Gets the node export definition for the given node.</td>
</tr>
<tr>
<td><code>boolean getNodeType(int type, NodeExport nodeExport)</code></td>
<td>Gets the default node export definition for the given node type.</td>
</tr>
<tr>
<td><code>void prepare(Graph graph)</code></td>
<td>Prepares the graph for the export process.</td>
</tr>
<tr>
<td><code>void release()</code></td>
<td>Ends the export process.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

Methods
getNodeType

public boolean getNodeType(int type,
                           NodeExport nodeExport)

Gets the default node export definition for the given node type.

GetNode has a higher priority than this function. That is, only if GetNode returns FALSE, the NodeExport of this function will be used.

Parameters:
  type - [in] Node type identifier.
  nodeExport - [out] The NodeExport which defines how to export the nodes of the given type.

Returns:
  TRUE.

getEdge

public boolean getEdge(long edge,
                        EdgeExport edgeExport)

Gets the edge export definition for the given edge.

Parameters:
  edge - Edge identifier.
  edgeExport - [out] The EdgeExport which defines how to export given edge.

Returns:
  TRUE if the given EdgeExport has been updated, otherwise FALSE will be returned and the default EdgeExport for the type the edge belongs to will be used.

getGraph

public boolean getGraph(GraphExport graphExport)

Gets the graph export definition.

Parameters:
  graphExport - [out] The GraphExport which defines how to export the graph.

Returns:
  TRUE.

getEdgeType

public boolean getEdgeType(int type,
                         EdgeExport edgeExport)

Gets the default node export definition for the given edge type.

GetEdge has a higher priority than this function. That is, only if GetEdge returns FALSE, the EdgeExport of this function will be used.

Parameters:
  type - [in] Edge type identifier.
edgeExport - [out] The EdgeExport which defines how to export the edges of the given type.

Returns:
TRUE.

prepare
public void prepare(Graph graph)

Prepares the graph for the export process.
It is called once before the export process.

Parameters:
graph - Graph to be exported.

getNode
public boolean getNode(long node, NodeExport nodeExport)

Gets the node export definition for the given node.

Parameters:
node - Node identifier.
nodeExport - [out] The NodeExport which defines how to export given node.

Returns:
TRUE if the given NodeExport has been updated, otherwise FALSE will be returned and the default NodeExport for the type the node belongs to will be used.

release
public void release()

Ends the export process.
It is called once after the export process.

enableType
public boolean enableType(int type)

Gets whether a node or edge type must be exported or not.

Parameters:
type - Node or edge type identifier.

Returns:
If TRUE all objects of the given type will be exported, otherwise they will not be exported.
com.sparsity.sparksee.gdb
Class ExportType

java.lang.Object
   +-java.lang.Enum
      +-com.sparsity.sparksee.gdb.ExportType

All Implemented Interfaces:
   Serializable, Comparable

public final class ExportType
extends Enum

Export type.

Author:
Sparsity Technologies http://www.sparsity-technologies.com

<table>
<thead>
<tr>
<th>Field Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static final GraphML Export to GraphML format.</td>
</tr>
<tr>
<td>public static final Graphviz Export to Graphviz format.</td>
</tr>
<tr>
<td>public static final YGraphML Export to YGRAPHML format.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>static ExportType valueOf(String name)</td>
</tr>
<tr>
<td>static ExportType[] values()</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Enum
clone, compareTo, equals, finalize, getDeclaringClass, hashCode, name, ordinal, toString, valueOf

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

Methods inherited from interface java.lang.Comparable
compareTo

Fields
**Graphviz**

`public static final com.sparsity.sparksee.gdb.ExportType Graphviz`  
Export to Graphviz format.  
Graphviz home page: http://www.graphviz.org

**GraphML**

`public static final com.sparsity.sparksee.gdb.ExportType GraphML`  
Export to GraphML format.  
GraphML home page: http://graphml.graphdrawing.org/

**YGraphML**

`public static final com.sparsity.sparksee.gdb.ExportType YGraphML`  
Export to YGRAPHML format.  
It is a GraphML format extended with a set of yWorks ("http://www.yworks.com") extensions. Thus, it allows for the visualization of the exported graph with the public yEd visualization tool ("http://www.yworks.com/products/yed").

**Methods**

**values**

`public static ExportType[] values()`

**valueOf**

`public static ExportType valueOf(String name)`
com.sparsity.sparksee.gdb

Class Graph

java.lang.Object

|--com.sparsity.sparksee.gdb.Graph

public class Graph extends Object

Graph class.

Each Database has a Graph associated, which is the persistent graph which contains all data stored into the graph database and is retrieved from a Session.

Check out the 'API' and the 'SPARKSEE graph database' sections in the SPARKSEE User Manual for more details on the use of this class.

Author: Sparsity Technologies http://www.sparsity-technologies.com

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>void backup(String file)</td>
</tr>
<tr>
<td>Dumps all the data to a backup file.</td>
</tr>
<tr>
<td>long countEdges()</td>
</tr>
<tr>
<td>Gets the number of edges.</td>
</tr>
<tr>
<td>long countNodes()</td>
</tr>
<tr>
<td>Gets the number of nodes.</td>
</tr>
<tr>
<td>long degree(long oid, int etype, EdgesDirection dir)</td>
</tr>
<tr>
<td>Gets the number of edges from or to the given node OID and for the given edge type.</td>
</tr>
<tr>
<td>void drop(long oid)</td>
</tr>
<tr>
<td>Drops the given OID.</td>
</tr>
<tr>
<td>void drop(Objects objs)</td>
</tr>
<tr>
<td>Drops all the OIDs from the given collection.</td>
</tr>
<tr>
<td>void dumpData(String file)</td>
</tr>
<tr>
<td>Dumps logical data to a file.</td>
</tr>
<tr>
<td>void dumpStorage(String file)</td>
</tr>
<tr>
<td>Dumps internal storage data to a file.</td>
</tr>
<tr>
<td>Objects edges(int etype, long tail, long head)</td>
</tr>
<tr>
<td>Gets all the edges of the given type between two given nodes (tail and head).</td>
</tr>
<tr>
<td>Objects explode(long oid, int etype, EdgesDirection dir)</td>
</tr>
<tr>
<td>Selects all edges from or to the given node OID and for the given edge type.</td>
</tr>
<tr>
<td>Objects explode(Objects objs, int etype, EdgesDirection dir)</td>
</tr>
<tr>
<td>Selects all edges from or to each of the node OID in the given collection and for the given edge type.</td>
</tr>
<tr>
<td>Method</td>
</tr>
<tr>
<td>---------------------------------------------</td>
</tr>
<tr>
<td><code>export(String file, ExportType type, ExportManager e)</code></td>
</tr>
<tr>
<td><code>findAttribute(int type, String name)</code></td>
</tr>
<tr>
<td><code>findAttributes(int type)</code></td>
</tr>
<tr>
<td><code>findEdge(int etype, long tail, long head)</code></td>
</tr>
<tr>
<td><code>findEdgeTypes()</code></td>
</tr>
<tr>
<td><code>findNodeTypes()</code></td>
</tr>
<tr>
<td><code>findObject(int attr, Value v)</code></td>
</tr>
<tr>
<td><code>findOrCreateEdge(int etype, long tail, long head)</code></td>
</tr>
<tr>
<td><code>findOrCreateObject(int attr, Value v)</code></td>
</tr>
<tr>
<td><code>findType(String name)</code></td>
</tr>
<tr>
<td><code>findTypes()</code></td>
</tr>
<tr>
<td><code>getAttribute(int attr)</code></td>
</tr>
<tr>
<td><code>getAttribute(long oid, int attr)</code></td>
</tr>
<tr>
<td><code>getAttribute(long oid, int attr, Value v)</code></td>
</tr>
<tr>
<td><code>getAttributeIntervalCount(int attr, Value lower, boolean includeLower, Value higher, boolean includeHigher)</code></td>
</tr>
<tr>
<td><code>getAttributeStatistics(int attr, boolean basic)</code></td>
</tr>
<tr>
<td><code>getAttributeText(long oid, int attr)</code></td>
</tr>
<tr>
<td><code>getEdgeData(long edge)</code></td>
</tr>
<tr>
<td><code>getEdgePeer(long edge, long node)</code></td>
</tr>
<tr>
<td>Method</td>
</tr>
<tr>
<td>---------------------------------------------</td>
</tr>
<tr>
<td><code>int </code>getObjectType(long oid)`</td>
</tr>
<tr>
<td><code>Type getType(int type)</code></td>
</tr>
<tr>
<td><code>Values getValues(int attr)</code></td>
</tr>
<tr>
<td><code>Objects heads(Objects edges)</code></td>
</tr>
<tr>
<td><code>void indexAttribute(int attr, AttributeKind kind)</code></td>
</tr>
<tr>
<td><code>Objects neighbors(long oid, int etype, EdgesDirection dir)</code></td>
</tr>
<tr>
<td><code>Objects neighbors(Objects objs, int etype, EdgesDirection dir)</code></td>
</tr>
<tr>
<td><code>int newAttribute(int type, String name, DataType dt, AttributeKind kind)</code></td>
</tr>
<tr>
<td><code>int newAttribute(int type, String name, DataType dt, AttributeKind kind, Value defaultValue)</code></td>
</tr>
<tr>
<td><code>long newEdge(int type, int tailAttr, Value tailV, int headAttr, Value headV)</code></td>
</tr>
<tr>
<td><code>long newEdge(int type, long tail, long head)</code></td>
</tr>
<tr>
<td><code>int newEdgeType(String name, boolean directed, boolean neighbors)</code></td>
</tr>
<tr>
<td><code>long newNode(int type)</code></td>
</tr>
<tr>
<td><code>int newNodeType(String name)</code></td>
</tr>
<tr>
<td><code>int newRestrictedEdgeType(String name, int tail, int head, boolean neighbors)</code></td>
</tr>
<tr>
<td><code>int newSessionAttribute(int type, DataType dt, AttributeKind kind)</code></td>
</tr>
<tr>
<td><code>int newSessionAttribute(int type, DataType dt, AttributeKind kind, Value defaultValue)</code></td>
</tr>
<tr>
<td><code>void removeAttribute(int attr)</code></td>
</tr>
<tr>
<td><code>void removeType(int type)</code></td>
</tr>
</tbody>
</table>
void renameAttribute(int attr, String newName)
    Renames an attribute.

void renameType(int type, String newName)
    Renames a type.

void renameType(String oldName, String newName)
    Renames a type.

Objects select(int type)
    Selects all OIDs belonging to the given type.

Objects select(int attr, Condition cond, Value v)
    Selects all OIDs satisfying the given condition for the given attribute.

Objects select(int attr, Condition cond, Value v, Objects restriction)
    Selects all OIDs satisfying the given condition for the given attribute.

Objects select(int attr, Condition cond, Value lower, Value higher)
    Selects all OIDs satisfying the given condition for the given attribute.

Objects select(int attr, Condition cond, Value lower, Value higher, Objects restriction)
    Selects all OIDs satisfying the given condition for the given attribute.

void setAttribute(long oid, int attr, Value v)
    Sets the Value for the given attribute and OID.

void setAttributeDefaultValue(int attr, Value v)
    Sets a default value for an attribute.

void setAttributeText(long oid, int attr, TextStream tstream)
    Sets the writable TextStream for the given text attribute and OID.

Objects tails(Objects edges)
    Gets all the tails from the given edges collection.

void tailsAndHeads(Objects edges, Objects tails, Objects heads)
    Gets all the tails and heads from the given edges collection.

Methods inherited from class java.lang.Object
    clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait,
    wait

Methods

setAttributeText

public void setAttributeText(long oid, int attr, TextStream tstream)
    Sets the writable TextStream for the given text attribute and OID.

Parameters:
oid - [in] Sparksee OID.
attr - [in] Sparksee attribute identifier.
tstream - [in] New Text value. This corresponds to a TextStream to write.

```java
public void dumpData(String file)
throws FileNotFoundException, 
          RuntimeException
```

Dumps logical data to a file.

**Parameters:**
- file - [in] Output file path.

**Throws:**
- java.io.FileNotFoundException - If the given file cannot be created.
- java.lang.RuntimeException - null

```java
public TypeList findTypes()
```

Gets all existing Sparksee node and edge type identifiers.

**Returns:**
- Sparksee node and edge type identifier list.

```java
public void setAttribute(long oid, 
                        int attr, 
                        Value v)
```

Sets the Value for the given attribute and OID.

**Parameters:**
- oid - [in] Sparksee OID.
- attr - [in] Sparksee attribute identifier.
- v - [in] Value for the given attribute and for the given OID.

```java
public Objects select(int attr, 
                      Condition cond, 
                      Value v)
```

Selects all OIDs satisfying the given condition for the given attribute.

**Parameters:**
- attr - [in] Sparksee attribute identifier.
- cond - [in] Condition to be satisfied.
select
public Objects select(int attr,
                     Condition cond,
                     Value v,
                     Objects restriction)

Selects all OIDs satisfying the given condition for the given attribute.

Parameters:
  attr - [in] Sparksee attribute identifier.
  cond - [in] Condition to be satisfied.
  v - [in] Value to be satisfied.
  restriction - [in] Objects to limit the select in this set of objects.

Returns:
  Objects instance.

renameAttribute
public void renameAttribute(int attr,
                            String newName)

Renames an attribute.

The new name must be available.

Parameters:
  attr - [in] Sparksee attribute identifier.
  newName - [in] The new name for the attribute.

getAttributeIntervalCount
public long getAttributeIntervalCount(int attr,
                                       Value lower,
                                       Boolean includeLower,
                                       Value higher,
                                       Boolean includeHigher)

Gets how many objects have a value into the given range for the given attribute.

This only works for the attributes with the AttributeKind Indexed or Unique.

Given values must belong to the same DataType than the attribute.

Parameters:
  attr - [in] Sparksee attribute identifier.
  lower - [in] Lower bound Value of the range.
  includeLower - [in] If TRUE, include lower bound Value of the range.
  higher - [in] Higher bound Value of the range.
  includeHigher - [in] If TRUE, include higher bound Value of the range.

Returns:
  Number of objects having a value into the given range.
tailsAndHeads

```java
public void tailsAndHeads(Objects edges,
                        Objects tails,
                        Objects heads)
```

 Gets all the tails and heads from the given edges collection.

**Parameters:**
- edges - [in] Sparksee edge identifier collection.
- tails - [in,out] If not NULL, all the tails will be stored here.
- heads - [in,out] If not NULL, all the heads will be stored here.

degree

```java
public long degree(long oid,
                   int etype,
                   EdgesDirection dir)
```

 Gets the number of edges from or to the given node OID and for the given edge type.

**Parameters:**
- oid - [in] Sparksee node OID.
- etype - [in] Sparksee edge type identifier.
- dir - [in] Direction.

**Returns:**
- The number of edges.

renameType

```java
public void renameType(String oldName,
                       String newName)
```

 Renames a type.

 The new name must be available.

**Parameters:**
- oldName - [in] The current name of the type to be renamed.
- newName - [in] The new name for the type.

dumpStorage

```java
public void dumpStorage(String file)
```

 Dumps internal storage data to a file.

**Parameters:**
- file - [in] Output file path.

**Throws:**
neighbors

public Objects neighbors(Objects objs,
int etype,
EdgesDirection dir)

Selects all neighbor nodes from or to each of the node OID in the given collection and for the given edge type.

Parameters:
- objs - [in] Sparksee node OID collection.
- etype - [in] Sparksee edge type identifier.
- dir - [in] Direction.

Returns:
- Objects instance.

event

public void export(String file,
ExportType type,
ExportManager e)
throws IOException

Exports the Graph.

Parameters:
- type - [in] Export type.
- e - [in] Defines how to do the export for each graph object.

Throws:
- java.io.IOException - null

getAttributes

public AttributeList getAttributes(long oid)

Gets all Sparksee attribute identifiers with a non-NULL value for the given Sparksee OID.

Parameters:
- oid - [in] Sparksee OID.

Returns:
- Sparksee attribute identifier list.

getAttributeStatistics

public AttributeStatistics getAttributeStatistics(int attr,
boolean basic)


Gets statistics from the given attribute.

**Parameters:**
- `attr` - [in] Sparksee attribute identifier.
- `basic` - [in] If FALSE all statistics are computed, if TRUE just those statistics marked as basic will be computed (see description of the AttributeStatistics class). Of course, computing just basic statistics will be faster than computing all of them.

**Returns:**
An AttributeStatistics instance.

### newNode

```java
public long newNode(int type)
```

Creates a new node instance.

**Parameters:**
- `type` - [in] Sparksee type identifier.

**Returns:**
Unique OID of the new node instance.

### getAttributeText

```java
public TextStream getAttributeText(long oid, int attr)
```

Gets the read-only TextStream for the given text attribute and OID.

**Parameters:**
- `oid` - [in] Sparksee OID.
- `attr` - [in] Sparksee attribute identifier.

**Returns:**
A TextStream. This returns a TextStream to read.

### countEdges

```java
public long countEdges()
```

Gets the number of edges.

**Returns:**
The number of edges.

### findEdgeTypes

```java
public TypeList findEdgeTypes()
```
Gets all existing Sparksee edge type identifiers.

**Returns:**
Sparksee edge type identifier list.

```java
select
public Objects select(int attr,
Condition cond,
Value lower,
Value higher)
```

Selects all OIDs satisfying the given condition for the given attribute.
This allows to perform the Between operation, thus it has two Value arguments.

**Parameters:**
- `attr` - [in] Sparksee attribute identifier.
- `cond` - [in] Condition to be satisfied. It must be the Between Condition.
- `lower` - [in] Lower-bound Value to be satisfied.
- `higher` - [in] Higher-bound Value to be satisfied.

**Returns:**
Objects instance.

```java
indexAttribute
public void indexAttribute(int attr,
AttributeKind kind)
```

Updates the index of the given attribute.
This just works if the current index of the attribute corresponds to the AttributeKind Basic and the new one is Indexed or Unique.

**Parameters:**
- `attr` - [in] Sparksee attribute identifier.
- `kind` - [in] Attribute kind.

```java
getType
public Type getType(int type)
```

Gets information about the given type.

**Parameters:**
- `type` - [in] Sparksee type identifier.

**Returns:**
The Type for the given Sparksee type identifier.

```java
setAttributeDefaultValue
public void setAttributeDefaultValue(int attr,
Value v)
```
Sets a default value for an attribute.

The default value will be applied to all the new nodes or edges.

The given value must have the same DataType as the attribute or be a NULL value to remove the current default value.

**Parameters:**

- `v`: [in] The default value to use for this attribute.

---

### findAttribute

**public int findAttribute(int type, String name)**

 Gets the Sparksee attribute identifier for the given type identifier and attribute name.

**Parameters:**

- `type`: [in] Sparksee type identifier.
- `name`: [in] Unique attribute name.

**Returns:**

- The Sparksee attribute identifier for the given type and attribute name or `InvalidAttribute` if there is no attribute with the given name for the given type.

---

### newAttribute

**public int newAttribute(int type, String name, DataType dt, AttributeKind kind)**

 Creates a new attribute.

**Parameters:**

- `type`: [in] Sparksee node or edge type identifier.
- `name`: [in] Unique name for the new attribute.
- `dt`: [in] Data type for the new attribute.
- `kind`: [in] Attribute kind.

**Returns:**

- Unique Sparksee attribute identifier.

---

### edges

**public Object edges(int etype, long tail, long head)**

 Gets all the edges of the given type between two given nodes (tail and head).

**Parameters:**

- `etype`: null
- `head`: [in] Head node identifier.
Returns: Objects instance.

**select**

```java
public Objects select(int type)
```

Selects all OIDs belonging to the given type.

**Parameters:**
- `type` - [in] Sparksee type identifier.

**Returns:** Objects instance.

**findNodeTypes**

```java
public TypeList findNodeTypes()
```

Gets all existing Sparksee node type identifiers.

**Returns:** Sparksee node type identifier list.

**getAttribute**

```java
public Value getAttribute(long oid, int attr)
```

Gets the Value for the given attribute and OID.

The other version of this call, where the Value is an output parameter instead of the return, is better because it allows the user to reuse an existing Value instance, whereas this call always creates a new Value instance to be returned.

It never returns NULL. Thus, in case the OID has a NULL value for the attribute it returns a NULL Value instance.

**Parameters:**
- `oid` - [in] Sparksee OID.
- `attr` - [in] Sparksee attribute identifier.

**Returns:** A new Value instance having the attribute value for the given OID.

**removeAttribute**

```java
public void removeAttribute(int attr)
```

Removes the given attribute.

**Parameters:**
- `attr` - [in] Sparksee attribute identifier.
backup

```java
public void backup(String file)
    throws FileNotFoundException, RuntimeException

Dumps all the data to a backup file.

See the Sparksee class Restore method.

Parameters:
    file - [in] Output backup file path.

Throws:
    java.io.FileNotFoundException - If the given file cannot be created.
    java.lang.RuntimeException - null
```

newSessionAttribute

```java
public int newSessionAttribute(int type,
        DataType dt,
        AttributeKind kind,
        Value defaultValue)

Creates a new Session attribute with a default value.

Session attributes are exclusive for the Session (just its Session can use the attribute) and are automatically removed when
the Session is closed (thus, attribute data is not persistent into the database).

Since they are not persistent, they cannot be retrieved from the database, so they do not have an identifier name.

Parameters:
    type - [in] Sparksee node or edge type identifier.
    dt - [in] Data type for the new attribute.
    kind - [in] Attribute kind.
    defaultValue - [in] The default value to use in each new node/edge.

Returns:
    Unique Sparksee attribute identifier.
```

findAttributes

```java
public AttributeList findAttributes(int type)

Gets all existing Sparksee attribute identifiers for the given type identifier.

Parameters:
    type - [in] Sparksee type identifier.

Returns:
    Sparksee attribute identifier list.
```

countNodes

```java
public long countNodes()

Gets the number of nodes.
```
Returns:
The number of nodes.

**getEdgeData**

```java
public EdgeData getEdgeData(long edge)
```

Gets information about an edge.

**Parameters:**
- `edge` - [in] Sparksee edge identifier.

**Returns:**
An EdgeData instance.

**neighbors**

```java
public Objects neighbors(long oid, int etype, EdgesDirection dir)
```

Selects all neighbor nodes from or to the given node OID and for the given edge type.

**Parameters:**
- `oid` - [in] Sparksee node OID.
- `etype` - [in] Sparksee edge type identifier.
- `dir` - [in] Direction.

**Returns:**
Objects instance.

**renameType**

```java
public void renameType(int type, String newName)
```

Renames a type.

The new name must be available.

**Parameters:**
- `type` - [in] The type to be renamed.
- `newName` - [in] The new name for the type.

**explode**

```java
public Objects explode(Objects objs, int etype, EdgesDirection dir)
```

Selects all edges from or to each of the node OID in the given collection and for the given edge type.

**Parameters:**
Objects - [in] Sparksee node OID collection.

etype - [in] Sparksee edge type identifier.

dir - [in] Direction.

Returns:
Objects instance.

newNodeType

public int newNodeType(String name)

Creates a new node type.

Parameters:
name - [in] Unique name for the new node type.

Returns:
Unique Sparksee type identifier.

newSessionAttribute

public int newSessionAttribute(int type,
Data Type dt,
Attribute Kind kind)

Creates a new Session attribute.

Session attributes are exclusive for the Session (just its Session can use the attribute) and are automatically removed when the Session is closed (thus, attribute data is not persistent into the database).

Since they are not persistent, they cannot be retrieved from the database, so they do not have an identifier name.

Parameters:
type - [in] Sparksee node or edge type identifier.
dt - [in] Data type for the new attribute.
dkind - [in] Attribute kind.

Returns:
Unique Sparksee attribute identifier.

tails

public Objects tails(Objects edges)

Gets all the tails from the given edges collection.

Parameters:
edges - [in] Sparksee edge identifier collection.

Returns:
The tails collection.
findOrCreateEdge

```java
public long findOrCreateEdge(int etype,
                           long tail,
                           long head)
```

Gets any of the edges of the specified type between two given nodes (tail and head).

If it can not find any edge of this type between them it tries to create a new one.

**Parameters:**
- `etype` - [in] Sparksee edge type identifier.
- `tail` - [in] Tail node identifier.
- `head` - [in] Head node identifier.

**Returns:**
Any of the edges or the Objects InvalidOID.

drop

```java
public void drop(long oid)
```

Drops the given OID.

It also removes its egdges as well as its attribute values.

**Parameters:**
- `oid` - [in] Sparksee OID to be removed.

ewEdgeType

```java
public int newEdgeType(String name,
                        boolean directed,
                        boolean neighbors)
```

Creates a new edge type.

**Parameters:**
- `name` - [in] Unique name for the new edge type.
- `directed` - [in] If TRUE, this creates a directed edge type, otherwise this creates a undirected edge type.
- `neighbors` - [in] If TRUE, this indexes neighbor nodes, otherwise not.

**Returns:**
Unique Sparksee type identifier.

heads

```java
public Objects heads(Objects edges)
```

Gets all the heads from the given edges collection.

**Parameters:**
- `edges` - [in] Sparksee edge identifier collection.

**Returns:**
The heads collection.
**findEdge**

```java
class com.sparsity.sparksee.gdb.Graph

public long findEdge(int etype,
                     long tail,
                     long head)
```

Gets any of the edges of the given type between two given nodes (tail and head).

If there are more than one, then any of them will be returned. And in case there are no edge between the given tail and head, the Objects InvalidOID will be returned.

**Parameters:**
- `etype` - [in] Sparksee edge type identifier.
- `tail` - [in] Tail node identifier.
- `head` - [in] Head node identifier.

**Returns:**
Any of the edges or the Objects InvalidOID.

**explode**

```java
class com.sparsity.sparksee.gdb.Graph

public Objects explode(long oid,
                       int etype,
                       EdgesDirection dir)
```

Selects all edges from or to the given node OID and for the given edge type.

**Parameters:**
- `oid` - [in] Sparksee node OID.
- `etype` - [in] Sparksee edge type identifier.
- `dir` - [in] Direction.

**Returns:**
Objects instance.

**drop**

```java
class com.sparsity.sparksee.gdb.Graph

public void drop(Objects objs)
```

Drops all the OIDs from the given collection.

See Drop method with the single OID parameter. This performs that call for all the elements into the collection.

**Parameters:**
- `objs` - [in] Objects collection with the OIDs to be removed.

**newAttribute**

```java
class com.sparsity.sparksee.gdb.Graph

public int newAttribute(int type,
                        String name,
                        DataType dt,
                        AttributeKind kind,
                        Value defaultValue)
```

Creates a new attribute with a default value.
Parameters:
  - type - [in] Sparksee node or edge type identifier.
  - name - [in] Unique name for the new attribute.
  - dt - [in] Data type for the new attribute.
  - kind - [in] Attribute kind.
  - defaultValue - [in] The default value to use in each new node/edge.

Returns:
  Unique Sparksee attribute identifier.

**getObjectType**

```java
public int getObjectType(long oid)
```

Get the Sparksee node or edge type identifier for the given OID.

Parameters:
  - oid - [in] Sparksee OID.

Returns:
  Sparksee node or edge type identifier.

**getAttribute**

```java
public Attribute getAttribute(int attr)
```

Gets information about the given attribute.

Parameters:
  - attr - [in] Sparksee attribute identifier.

Returns:
  The Attribute for the given Sparksee attribute identifier.

**newEdge**

```java
public long newEdge(int type,
                    int tailAttr,
                    Value tailV,
                    int headAttr,
                    Value headV)
```

Creates a new edge instance.

The tail of the edge will be any node having the given tailV Value for the given tailAttr attribute identifier, and the head of the edge will be any node having the given headV Value for the given headAttr attribute identifier.

Parameters:
  - type - [in] Sparksee type identifier.
  - headAttr - [in] Sparksee attribute identifier.
  - headV - [in] Value.

Returns:
  Unique OID of the new edge instance.
getValues
public Values getValues(int attr)

Gets the Value collection for the given attribute.

Parameters:
attr - [in] Sparksee attribute identifier.

getAddress
public long getEdgePeer(long edge, long node)

Gets the other end for the given edge.

Parameters:
edge - [in] Sparksee edge identifier.
node - [in] Sparksee node identifier. It must be one of the ends of the edge.

Returns:
The other end of the edge.

findObject
public long findObject(int attr, Value v)

Finds one object having the given Value for the given attribute.

If there are more than one, then any of them will be returned. And in case there are no object having this Value, the Objects InvalidOID will be returned.

Parameters:
attr - [in] Sparksee attribute identifier.
v - [in] Value.

Returns:
Sparksee OID or the Objects InvalidOID.

newEdge
public long newEdge(int type, long tail, long head)

Creates a new edge instance.

Parameters:
type - [in] Sparksee type identifier.
tail - [in] Source Sparksee OID.
head - [in] Target Sparksee OID.
Returns:
Unique OID of the new edge instance.

select
public Objects select(int attr,
                    Condition cond,
                    Value lower,
                    Value higher,
                    Objects restriction)

Selects all OIDs satisfying the given condition for the given attribute.
This allows to perform the Between operation, thus it has two Value arguments.

Parameters:
attr - [in] Sparksee attribute identifier.
cond - [in] Condition to be satisfied. It must be the Between Condition.
lower - [in] Lower-bound Value to be satisfied.
higher - [in] Higher-bound Value to be satisfied.
restriction - [in] Objects to limit the select in this set of objects.

Returns:
Objects instance.

findType
public int findType(String name)

Gets the Sparksee type identifier for the given type name.

Parameters:
name - [in] Unique type name.

Returns:
The Sparksee type identifier for the given type name or the Type InvalidType if there is no type with the given name.

getAttribute
public void getAttribute(long oid,
                          int attr,
                          Value v)

Gets the Value for the given attribute and OID.

Parameters:
oid - [in] Sparksee OID.
attr - [in] Sparksee attribute identifier.
v - [in|out] Value for the given attribute and for the given OID.

findOrCreateObject
public long findOrCreateObject(int attr,
                               Value v)
Finds one object having the given Value for the attribute or it creates one if it does not exist any.

If the attribute is a node or edge attribute and at least one node/edge with that value is found, it returns one of them. But if it does not exist, then: If it's a node attribute it will create it and set the attribute. If it's an edge attribute it will return the InvalidOID.

Using this method with a global attribute will return the InvalidOID.

**Parameters:**
- `attr` - [in] Sparksee attribute identifier.
- `v` - [in] Value.

**Returns:**
Sparksee OID or the Objects InvalidOID.

---

**removeType**

```java
public void removeType(int type)
```

Removes the given type.

This fails if there exist attributes defined for the type or if there exist restricted edges referencing this type.

**Parameters:**
- `type` - [in] Sparksee type identifier.

---

**newRestrictedEdgeType**

```java
public int newRestrictedEdgeType(String name, int tail, int head, boolean neighbors)
```

Creates a new restricted edge type.

**Parameters:**
- `name` - [in] Unique name for the new edge type.
- `tail` - [in] Tail Sparksee node type identifier.
- `head` - [in] Head Sparksee node type identifier.
- `neighbors` - [in] If TRUE, this indexes neighbor nodes, otherwise not.

**Returns:**
Unique Sparksee type identifier.
public class GraphExport extends Object

Stores the graph exporting values.

Author: Sparsity Technologies http://www.sparsity-technologies.com

**Constructor Summary**

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
<td>GraphExport()</td>
</tr>
</tbody>
</table>

**Method Summary**

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
<td>String</td>
<td>getLabel()</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Gets the graph label.</td>
</tr>
<tr>
<td>void</td>
<td></td>
<td>setDefaults()</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sets to default values.</td>
</tr>
<tr>
<td>void</td>
<td>String</td>
<td>setLabel(String l)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sets the graph label.</td>
</tr>
</tbody>
</table>

**Methods inherited from class java.lang.Object**

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

**Constructors**

**GraphExport**

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
<td>GraphExport()</td>
</tr>
</tbody>
</table>

- Creates a new GraphExport instance.

**Methods**

**setLabel**

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
<td>void setLabel(String l)</td>
</tr>
</tbody>
</table>

- Sets the graph label.
Sets the graph label.

**Parameters:**

- \( l \) - [in] The graph label.

---

**getLabel**

```java
public String getLabel()
```

Gets the graph label.

**Returns:**

The graph label.

---

**setDefaults**

```java
public void setDefaults()
```

Sets to default values.
com.sparsity.sparksee.gdb
Class Int32List

java.lang.Object
   ↳ com.sparsity.sparksee.gdb.Int32List

All Implemented Interfaces:
   Iterable

public class Int32List
extends Object
implements Iterable

Sparksee 32-bit signed integer list.
It stores a 32-bit signed integer list.

Use Int32ListIterator to access all elements into this collection.

Author:
Sparsity Technologies http://www.sparsity-technologies.com

Constructor Summary

<table>
<thead>
<tr>
<th>public</th>
<th>Int32List(Collection col)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Creates a new instance from an integer collection.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public</th>
<th>Int32List()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Constructor.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public</th>
<th>Int32List(int[] list)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Creates a new instance from an integer array.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>void</th>
<th>add(int value)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Adds an 32-bit signed integer at the end of the list.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>clear()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Clears the list.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>count()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Number of elements in the list.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Int32ListIterator</th>
<th>iterator()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets a new Int32ListIterator.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

Methods inherited from interface java.lang.Iterable

iterator
Constructors

**Int32List**

```java
public Int32List(Collection col)
```

Creates a new instance from an integer collection.

**Parameters:**
- `col` - Collection to initialize the instance.

**Int32List**

```java
public Int32List()
```

Constructor.

This creates an empty list.

**Int32List**

```java
public Int32List(int[] list)
```

Creates a new instance from an integer array.

**Parameters:**
- `list` - Integer array to initialize the instance.

Methods

**add**

```java
public void add(int value)
```

 Adds an 32-bit signed integer at the end of the list.

**Parameters:**
- `value` - null

**clear**

```java
public void clear()
```

Clears the list.

**iterator**

```java
public Int32ListIterator iterator()
```
Gets a new Int32ListIterator.

**Returns:**
Int32ListIterator instance.

---

### count

**public int count()**

Number of elements in the list.

**Returns:**
Number of elements in the list.
**com.sparsity.sparksee.gdb**

**Class Int32ListIterator**

```java
public class Int32ListIterator
    extends Object
    implements Iterator
```

Iterator to traverse all the integer into a Int32List instance.

**Author:**
Sparsity Technologies http://www.sparsity-technologies.com

---

**Method Summary**

<table>
<thead>
<tr>
<th>Method</th>
<th>Return Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hasNext()</td>
<td>boolean</td>
<td>Gets if there are more elements.</td>
</tr>
<tr>
<td>next()</td>
<td>Integer</td>
<td>See nextInt32().</td>
</tr>
<tr>
<td>nextInt32()</td>
<td>Integer</td>
<td>Gets the next element.</td>
</tr>
<tr>
<td>remove()</td>
<td>void</td>
<td>Operation not supported.</td>
</tr>
</tbody>
</table>

**Methods inherited from class java.lang.Object**

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

**Methods inherited from interface java.util.Iterator**

hasNext, next, remove

---

**Methods**

**nextInt32**

```java
public Integer nextInt32()
```

Gets the next element.
hasNext
public boolean hasNext()

Gets if there are more elements.

Returns:
TRUE if there are more elements, FALSE otherwise.

remove
public void remove()

Operation not supported.

next
public Integer next()

See nextInt32().
com.sparsity.sparksee.gdb
Class LogLevel

java.lang.Object
   +---java.lang.Enum
   +---com.sparsity.sparksee.gdb.LogLevel

All Implemented Interfaces:
   Serializable, Comparable

public final class LogLevel
extends Enum

Log level enumeration.

Log level priority order is as follows, from minimum to maximum log information: Off (log is disabled), Severe, Warning, Info, Config, Fine, Debug.

Author:
   Sparsity Technologies http://www.sparsity-technologies.com

<table>
<thead>
<tr>
<th>Field Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static final</td>
</tr>
<tr>
<td>public static final</td>
</tr>
<tr>
<td>public static final</td>
</tr>
<tr>
<td>public static final</td>
</tr>
<tr>
<td>public static final</td>
</tr>
<tr>
<td>public static final</td>
</tr>
<tr>
<td>public static final</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>static LogLevel</td>
</tr>
<tr>
<td>static LogLevel[]</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Enum

clone, compareTo, equals, finalize, getDeclaringClass, hashCode, name, ordinal, toString, valueOf
### Methods inherited from class java.lang.Object

- clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

### Methods inherited from interface java.lang.Comparable

- compareTo

### Fields

#### Off

`public static final com.sparsity.sparksee.gdb LogLevel Off`  
Disable log.

#### Severe

`public static final com.sparsity.sparksee.gdb LogLevel Severe`  
Severe log level.  
This is the lower log level, just errors are shown.

#### Warning

`public static final com.sparsity.sparksee.gdb LogLevel Warning`  
Warning log level.  
Errors and warnings are shown.

#### Info

`public static final com.sparsity.sparksee.gdb LogLevel Info`  
Info log level.  
Errors, warnings and information messages are shown.

#### Config

`public static final com.sparsity.sparksee.gdb LogLevel Config`  
Config log level.  
Errors, warnings, information messages and configuration details of the different components are shown.

#### Fine

`public static final com.sparsity.sparksee.gdb LogLevel Fine`
Fine log level.

This is the higher and finest public log level, everything is dumped to the log.

---

**Debug**

```java
public static final com.sparsity.sparksee.gdb.LogLevel Debug
```

Debug log level.

This is for Sparksee development purposes and just works with debug versions of the library.

---

### Methods

**values**

```java
public static LogLevel[] values()
```

**valueOf**

```java
public static LogLevel valueOf(String name)
```
**com.sparsity.sparksee.gdb**

**Class NodeExport**

```java
public class NodeExport extends Object

Stores the node exporting values.

When 'fit' is set to TRUE, then 'height' and 'width' will be ignored.

Some properties may be ignored depending on the exportation type.

Default values are:

Label: "" (empty string).

Shape: Box.

Color: 10863606 (0xa5c3f6).

Label color: 0 (0x000000, Black).

Height: 25px.

Width: 25px.

Fit: TRUE.

Font size: 10.

**Author:**
Sparsity Technologies http://www.sparsity-technologies.com
```

### Constructor Summary

<table>
<thead>
<tr>
<th>public</th>
<th>NodeExport()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creates a new instance.</td>
<td></td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>java.awt.Color</th>
<th>getColor()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gets the color of the node.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>getColorRGB()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gets the node color.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>getFontSize()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gets the node label font size.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>getHeight()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gets the node height.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>getLabel()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gets the node label.</td>
<td></td>
</tr>
<tr>
<td>java.awt.Color</td>
<td>getLabelColor()</td>
</tr>
<tr>
<td>----------------</td>
<td>----------------</td>
</tr>
<tr>
<td></td>
<td>Gets the color of the label.</td>
</tr>
<tr>
<td>int</td>
<td>getLabelColorRGB()</td>
</tr>
<tr>
<td></td>
<td>Gets the node label color.</td>
</tr>
<tr>
<td>NodeShape</td>
<td>getShape()</td>
</tr>
<tr>
<td></td>
<td>Gets the node shape.</td>
</tr>
<tr>
<td>int</td>
<td>getWidth()</td>
</tr>
<tr>
<td></td>
<td>Gets the node width.</td>
</tr>
<tr>
<td>boolean</td>
<td>isFit()</td>
</tr>
<tr>
<td></td>
<td>Gets whether the node size is fitted to the label or not.</td>
</tr>
<tr>
<td>void</td>
<td>setColor(java.awt.Color c)</td>
</tr>
<tr>
<td></td>
<td>Sets the color of the node.</td>
</tr>
<tr>
<td>void</td>
<td>setColorRGB(int c)</td>
</tr>
<tr>
<td></td>
<td>Sets the node color.</td>
</tr>
<tr>
<td>void</td>
<td>setDefaults()</td>
</tr>
<tr>
<td></td>
<td>Sets to default values.</td>
</tr>
<tr>
<td>void</td>
<td>setFit(boolean f)</td>
</tr>
<tr>
<td></td>
<td>Sets the node fit property.</td>
</tr>
<tr>
<td>void</td>
<td>setFontSize(int s)</td>
</tr>
<tr>
<td></td>
<td>Sets the node label font size.</td>
</tr>
<tr>
<td>void</td>
<td>setHeight(int h)</td>
</tr>
<tr>
<td></td>
<td>Sets the node height.</td>
</tr>
<tr>
<td>void</td>
<td>setLabel(String l)</td>
</tr>
<tr>
<td></td>
<td>Sets the node label.</td>
</tr>
<tr>
<td>void</td>
<td>setLabelColor(java.awt.Color c)</td>
</tr>
<tr>
<td></td>
<td>Sets the color of the label.</td>
</tr>
<tr>
<td>void</td>
<td>setLabelColorRGB(int c)</td>
</tr>
<tr>
<td></td>
<td>Sets the node label color.</td>
</tr>
<tr>
<td>void</td>
<td>setShape(NodeShape s)</td>
</tr>
<tr>
<td></td>
<td>Sets the node shape.</td>
</tr>
<tr>
<td>void</td>
<td>setWidth(int w)</td>
</tr>
<tr>
<td></td>
<td>Gets the node width.</td>
</tr>
</tbody>
</table>

**Methods inherited from class** java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

## Constructors
NodeExport

public NodeExport()

Creates a new instance.

Methods

getShape

public NodeShape getShape()

Gets the node shape.

Returns:
The node shape.

g getColor

public java.awt.Color getColor()

Gets the color of the node.

setLabelColorRGB

public void setLabelColorRGB(int c)

Sets the node label color.

Parameters:
c - [in] The node label color.

getFontSize

public int getFontSize()

Gets the node label font size.

Returns:
The node label font size.

setWidth

public void setWidth(int w)

Gets the node width.
Parameters:

- \( w \) - The node width in pixels.

**setDefaults**

```java
public void setDefaults()
```

Sets to default values.

**getColorRGB**

```java
public int getColorRGB()
```

Gets the node color.

**Returns:**

The node color.

**setFontSize**

```java
public void setFontSize(int s)
```

Sets the node label font size.

**Parameters:**

- \( s \) - [in] The node label font size.

**setHeight**

```java
public void setHeight(int h)
```

Sets the node height.

**Parameters:**

- \( h \) - [in] The node height in pixels.

**isFit**

```java
public boolean isFit()
```

Gets whether the node size is fitted to the label or not.

**Returns:**

If TRUE, then the node size is fitted to the label, otherwise the size is fixed with the values of 'height' and 'width'.

**setFit**

```java
public void setFit(boolean f)
```
Sets the node fit property.

Parameters:
- \(f\) - [in] If TRUE, then the node size is fitted to the label ('height' and 'width' will be ignored), otherwise the size is fixed with the values of 'height' and 'width'.

**getLabelColorRGB**

```java
public int getLabelColorRGB()
```

Gets the node label color.

**Returns:**
The node label color.

**getWidth**

```java
public int getWidth()
```

Gets the node width.

**Returns:**
The node width in pixels.

**getLabelColor**

```java
public java.awt.Color getLabelColor()
```

Gets the color of the label.

**setColor**

```java
public void setColor(java.awt.Color c)
```

Sets the color of the node.

**Parameters:**
- \(c\) - New value.

**setLabel**

```java
public void setLabel(String l)
```

Sets the node label.

**Parameters:**
- \(l\) - [in] The node label.
setColorRGB
public void setColorRGB(int c)
   Sets the node color.

Parameters:
   c - The node color.

getLabel
public String getLabel()
   Gets the node label.

Returns:
   The node label.

setShape
public void setShape(NodeShape s)
   Sets the node shape.

Parameters:
   s - [in] The node shape.

getHeight
public int getHeight()
   Gets the node height.

Returns:
   The node height in pixels.

setLabelColor
public void setLabelColor(java.awt.Color c)
   Sets the color of the label.

Parameters:
   c - New value.
com.sparsity.sparksee.gdb
Class NodeShape

java.lang.Object
   +-java.lang.Enum
   |   +-com.sparsity.sparksee.gdb.NodeShape

All Implemented Interfaces:
   Serializable, Comparable

public final class NodeShape
extends Enum

Node shape.

Author:
Sparsity Technologies http://www.sparsity-technologies.com

Field Summary

| public static final | Box | Box shape. |
| public static final | Round | Round shape. |

Method Summary

| static NodeShape | NodeShape | NodeShape[] | values() |
| static NodeShape.valueOf(String name) |
| static NodeShape[] | values() |

Methods inherited from class java.lang.Enum

clone, compareTo, equals, finalize, getDeclaringClass, hashCode, name, ordinal, toString, valueOf

Methods inherited from class java.lang.Object

close, compareTo, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface java.lang.Comparable

compareTo

Fields
Box
public static final com.sparsity.sparksee.gdb.NodeShape Box

Box shape.

Round
public static final com.sparsity.sparksee.gdb.NodeShape Round

Round shape.

Methods

values
public static NodeShape[] values()

valueOf
public static NodeShape valueOf(String name)
com.sparsity.sparksee.gdb
Class Objects

java.lang.Object
  +--com.sparsity.sparksee.gdb.Objects

All Implemented Interfaces:
  Iterable, Closeable, Set

public class Objects
  extends Object
  implements Set, Closeable, Iterable

Object identifier set class.

It stores a collection of Sparksee object identifiers as a set. As a set, there is no order and no duplicated elements.

This class should be used just to store large collections. Otherwise, it is strongly recommended to use common classes from the language API.

This class is not thread-safe.

ObjectsIterator must be used to traverse all the elements into the set.

When the Objects instance is closed, it closes all existing and non-closed ObjectsIterator instances too.

Author:
  Sparsity Technologies http://www.sparsity-technologies.com

Field Summary

<table>
<thead>
<tr>
<th>public static</th>
<th>InvalidOID</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Invalid object identifier constant.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>boolean</th>
<th>add(long e)</th>
<th>Adds an element into the collection.</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>add(Long e)</td>
<td>Adds the specified element to this set if it is not already present (optional operation).</td>
</tr>
<tr>
<td>boolean</td>
<td>addAll(Collection clctn)</td>
<td>Adds all of the elements in the specified collection to this set if they're not already present (optional operation).</td>
</tr>
<tr>
<td>long</td>
<td>any()</td>
<td>Gets an element from the collection.</td>
</tr>
<tr>
<td>void</td>
<td>clear()</td>
<td>Clears the collection removing all its elements.</td>
</tr>
<tr>
<td>void</td>
<td>close()</td>
<td>Closes the Objects instance.</td>
</tr>
<tr>
<td>Method</td>
<td>Signature</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>static Objects</td>
<td>combineDifference(Objects objs1, Objects objs2)</td>
<td>Creates a new Objects instance which is the difference of the two given.</td>
</tr>
<tr>
<td>static Objects</td>
<td>combineIntersection(Objects objs1, Objects objs2)</td>
<td>Creates a new Objects instance which is the intersection of the two given.</td>
</tr>
<tr>
<td>static Objects</td>
<td>combineUnion(Objects objs1, Objects objs2)</td>
<td>Creates a new Objects instance which is the union of the two given.</td>
</tr>
<tr>
<td>boolean</td>
<td>contains(Object o)</td>
<td>Returns true if this collections contains the specified element or Objects.</td>
</tr>
<tr>
<td>boolean</td>
<td>contains(Objects objs)</td>
<td>Check if this objects contains the other one.</td>
</tr>
<tr>
<td>boolean</td>
<td>containsAll(Collection clctn)</td>
<td>Returns true if this set contains all of the elements of the specified collection.</td>
</tr>
<tr>
<td>Objects</td>
<td>copy()</td>
<td>Creates a new Objects instance as a copy of the given one.</td>
</tr>
<tr>
<td>long</td>
<td>copy(Objects objs)</td>
<td>Performs the copy operation.</td>
</tr>
<tr>
<td>long</td>
<td>count()</td>
<td>Gets the number of elements into the collection.</td>
</tr>
<tr>
<td>long</td>
<td>difference(Objects objs)</td>
<td>Performs the difference operation.</td>
</tr>
<tr>
<td>boolean</td>
<td>equals(Object o)</td>
<td>Returns true if the collection is equal to the object.</td>
</tr>
<tr>
<td>boolean</td>
<td>equals(Objects objs)</td>
<td>Checks if the given Objects contains the same information.</td>
</tr>
<tr>
<td>boolean</td>
<td>exists(long e)</td>
<td>Gets if the given element exists into the collection.</td>
</tr>
<tr>
<td>long</td>
<td>intersection(Objects objs)</td>
<td>Performs the intersection operation.</td>
</tr>
<tr>
<td>boolean</td>
<td>isClosed()</td>
<td>Gets if Objects instance has been closed or not.</td>
</tr>
<tr>
<td>boolean</td>
<td>isEmpty()</td>
<td>Returns true if this Objects contains no elements.</td>
</tr>
<tr>
<td>ObjectsIterator</td>
<td>iterator()</td>
<td>Gets an ObjectsIterator.</td>
</tr>
<tr>
<td>ObjectsIterator</td>
<td>iteratorFromElement(long e)</td>
<td>Gets an ObjectsIterator starting from the given element.</td>
</tr>
<tr>
<td>ObjectsIterator</td>
<td>iteratorFromIndex(long index)</td>
<td>Gets an ObjectsIterator skipping index elements.</td>
</tr>
<tr>
<td>boolean</td>
<td>remove(long e)</td>
<td>Removes an element from the collection.</td>
</tr>
<tr>
<td>boolean</td>
<td><code>remove(Object o)</code></td>
<td>Removes the specified element from this set if it is present (optional operation).</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>boolean</td>
<td><code>removeAll(Collection clctn)</code></td>
<td>Removes from this set all of its elements that are contained in the specified collection (optional operation).</td>
</tr>
<tr>
<td>boolean</td>
<td><code>retainAll(Collection clctn)</code></td>
<td>Retains only the elements in this set that are contained in the specified collection (optional operation).</td>
</tr>
<tr>
<td><code>Objects</code></td>
<td><code>sample(Objects exclude, long samples)</code></td>
<td>Creates a new <code>Objects</code> instance which is a sample of the calling one.</td>
</tr>
<tr>
<td>int</td>
<td><code>size()</code></td>
<td>Gets the size of the collection.</td>
</tr>
<tr>
<td><code>Object[]</code></td>
<td><code>toArray()</code></td>
<td>Returns an array containing all of the object identifiers in this set.</td>
</tr>
<tr>
<td><code>Object[]</code></td>
<td><code>toArray(Object[] ts)</code></td>
<td>Returns an array containing all of the object identifiers in this set; the runtime type of the returned array is that of the specified array.</td>
</tr>
<tr>
<td>long</td>
<td><code>union(Objects objs)</code></td>
<td>Performs the union operation.</td>
</tr>
</tbody>
</table>

Methods inherited from class `java.lang.Object`:
- `clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`.

Methods inherited from interface `java.util.Set`:
- `add`, `addAll`, `clear`, `contains`, `containsAll`, `equals`, `hashCode`, `isEmpty`, `iterator`, `remove`, `removeAll`, `retainAll`, `size`, `toArray`, `toArray`.

Methods inherited from interface `java.util.Collection`:
- `add`, `addAll`, `clear`, `contains`, `containsAll`, `equals`, `hashCode`, `isEmpty`, `iterator`, `remove`, `removeAll`, `retainAll`, `size`, `toArray`, `toArray`.

Methods inherited from interface `java.lang.Iterable`:
- `iterator`.

Methods inherited from interface `java.io.Closeable`:
- `close`.

Methods inherited from interface `java.lang.Iterable`:
- `iterator`.

**Fields**
InvalidOID

public static int InvalidOID

Invalid object identifier constant.

Methods

any

public long any()
    throws NoSuchElementException,
             RuntimeException

    Gets an element from the collection.

    Returns:
    Any element from the collection.

    Throws:
    java.util.NoSuchElementException - whether the collection is empty.
    java.lang.RuntimeException - null

contains

public boolean contains(Object o)

    Returns true if this collections contains the specified element or Objects.

    Parameters:
    o - element or Objects whose presence in this set is to be tested.

    Returns:
    true if this set contains the specified element or Objects.

remove

public boolean remove(Object o)

    Removes the specified element from this set if it is present (optional operation).
    More formally, removes an element e such that (o==null ? e==null : o.equals(e)), if the set contains such an element.
    Returns true if the set contained the specified element (or equivalently, if the set changed as a result of the call). (The set
    will not contain the specified element once the call returns.)

    Parameters:
    o - object to be removed from this set, if present.

    Returns:
    true if the set contained the specified element.
equals
public boolean equals(Object o)

    Returns true if the collection is equal to the object.

Parameters:
o - object to compare with the collection.

Returns:
true if the objects are equal or false otherwise.

difference
public long difference(Objects objs)

    Performs the difference operation.
    This updates the Objects calling instance removing those existing elements at the given Objects instance.

Parameters:
obs - [in] Objects instance.

Returns:
Number of elements into the collection once the operation has been executed.

combineUnion
public static Objects combineUnion(Objects objs1, Objects objs2)

    Creates a new Objects instance which is the union of the two given.
    Two given Objects belong to the same Session.

Parameters:
obs1 - [in] Objects instance.
obs2 - [in] Objects instance.

Returns:
New Objects instance.

containsAll
public boolean containsAll(Collection clctn)

    Returns true if this set contains all of the elements of the specified collection.
    If the specified collection is also a set, this method returns true if it is a subset of this set.

Parameters:
clctn - collection to be checked for containment in this set.

Returns:
true if this set contains all of the elements of the specified collection.
iteratorFromElement

public ObjectsIterator iteratorFromElement(long e)

Gets an ObjectsIterator starting from the given element.

Objects collection has no order, so this method is implementation-dependent. e[in] The first element to traverse in the resulting

Parameters:

  e - [in] The first element to traverse in the resulting ObjectsIterator instance.

Returns:

ObjectsIterator instance.

equals

public boolean equals(Objects objs)

Checks if the given Objects contains the same information.

Parameters:

  objs - [in] Objects instance.

Returns:

True if the objects are equal or false otherwise.

add

public boolean add(Long e)

Adds the specified element to this set if it is not already present (optional operation).

More formally, adds the specified element, o, to this set if this set contains no element e such that (o==null ? e==null : o.equals(e)). If this set already contains the specified element, the call leaves this set unchanged and returns false. In combination with the restriction on constructors, this ensures that sets never contain duplicate elements. The stipulation above does not imply that sets must accept all elements; sets may refuse to add any particular element, including null, and throwing an exception, as described in the specification for Collection.add. Individual set implementations should clearly document any restrictions on the the elements that they may contain.

Parameters:

  e - element to be added to this set.

Returns:

true if this set did not already contain the specified element.

copy

public long copy(Objects objs)

Performs the copy operation.

This updates the Objects calling instance and copies the given Objects instance.

Parameters:

  objs - [in] Objects instance.

Returns:

Number of elements into the collection once the operation has been executed.
**combineIntersection**

```java
public static Objects combineIntersection(Objects objs1, Objects objs2)
```

Creates a new Objects instance which is the intersection of the two given.

Two given Objects belong to the same Session.

**Parameters:**
- `objs1` - [in] Objects instance.
- ` objs2` - [in] Objects instance.

**Returns:**
- New Objects instance.

---

**close**

```java
public void close()
```

Closes the Objects instance.

It must be called to ensure the integrity of all data.

---

**isEmpty**

```java
public boolean isEmpty()
```

Returns true if this Objects contains no elements.

**Returns:**
- true if the collection contains no elements.

---

**contains**

```java
public boolean contains(Objects objs)
```

Check if this objects contains the other one.

**Parameters:**
- `objs` - Objects collection.

**Returns:**
- True if it contains the given object.

---

**clear**

```java
public void clear()
```

Clears the collection removing all its elements.
count
public long count()

Gets the number of elements into the collection.

Returns:
The number of elements into the collection.

retainAll
public boolean retainAll(Collection clctn)

Retains only the elements in this set that are contained in the specified collection (optional operation).

In other words, removes from this set all of its elements that are not contained in the specified collection. If the specified collection is also a set, this operation effectively modifies this set so that its value is the intersection of the two sets.

Parameters:
clctn - collection that defines which elements this set will retain.

Returns:
true if this collection changed as a result of the call.

remove
public boolean remove(long e)

Removes an element from the collection.

Parameters:
e - [in] Element to be removed.

Returns:
TRUE if the element is removed, FALSE if the element was not into the collection.

iteratorFromIndex
public ObjectsIterator iteratorFromIndex(long index)

Gets an ObjectsIterator skipping index elements.

Objects collection has no order, so this method is implementation-dependent.

Parameters:
index - [in] The number of elements to skip from the beginning. It must be in the range [0..Size).

Returns:
ObjectsIterator instance.

union
public long union(Objects objs)
Performs the union operation.
This adds all existing elements of the given Objects instance to the Objects calling instance

**Parameters:**
- `objs` - [in] Objects instance.

**Returns:**
Number of elements into the collection once the operation has been executed.

---

### isClosed

**public boolean isClosed()**

Gets if Objects instance has been closed or not.

**Returns:**
TRUE if the Objects instance has been closed, FALSE otherwise.

**See Also:**
- `close()`

---

### add

**public boolean add(long e)**

Adds an element into the collection.

**Parameters:**
- `e` - [in] Element to be added.

**Returns:**
TRUE if the element is added, FALSE if the element was already into the collection.

---

### removeAll

**public boolean removeAll(Collection clctn)**

Removes from this set all of its elements that are contained in the specified collection (optional operation).
If the specified collection is also a set, this operation effectively modifies this set so that its value is the asymmetric set difference of the two sets.

**Parameters:**
- `clctn` - collection that defines which elements will be removed from this set.

**Returns:**
true if this set changed as a result of the call

---

### toArray

**public Object[] toArray()**

Returns an array containing all of the object identifiers in this set.
Obeys the general contract of the Collection.toArray method.
Returns:
an array containing all of the elements in this set.

toArray

public Object[] toArray(Object[] ts)

Returns an array containing all of the object identifiers in this set; the runtime type of the returned array is that of the
specified array.

Obeys the general contract of the Collection.toArray(Object[]) method.

Parameters:

   ts - the array into which the elements of this set are to be stored, if it is big enough; otherwise, a new array of the
   same runtime type is allocated for this purpose.

Returns:
an array containing the elements of this set.

addAll

public boolean addAll(Collection clctn)

Adds all of the elements in the specified collection to this set if they're not already present (optional operation).

If the specified collection is also a set, the addAll operation effectively modifies this set so that its value is the union of the
two sets. The behavior of this operation is unspecified if the specified collection is modified while the operation is in
progress.

Parameters:

   clctn - collection whose elements are to be added to this set.

Returns:

true if this set changed as a result of the call.

exists

public boolean exists(long e)

Gets if the given element exists into the collection.

Parameters:

   e - [in] Element.

Returns:

TRUE if the element exists into the collection, FALSE otherwise.

combineDifference

public static Objects combineDifference(Objects obj1, Objects obj2)

Creates a new Objects instance which is the difference of the two given.

Two given Objects belong to the same Session.

Parameters:

   obj1 - [in] Objects instance.
objs2 - [in] Objects instance.

**Returns:**
New Objects instance.

---

### sample

```java
public Objects sample(Objects exclude, long samples)
```

Creates a new Objects instance which is a sample of the calling one.

**Parameters:**
- `exclude` - [in] If not NULL, elements into this collection will be excluded from the resulting one.
- `samples` - [in] Number of elements into the resulting collection.

**Returns:**
Sample collection.

---

### size

```java
public int size()
```

Gets the size of the collection.

It is the same as `count()` if the number of elements is \(\leq\) `java.lang.Integer.MAX_VALUE`, otherwise `java.lang.Integer.MAX_VALUE` is returned.

**Returns:**
It returns the same as `count()` or `java.lang.Integer.MAX_VALUE`.

---

### intersection

```java
public long intersection(Objects objs)
```

Performs the intersection operation.

Updates the Objects calling instance setting those existing elements at both two collections and removing all others.

**Parameters:**
- `objs` - [in] Objects instance.

**Returns:**
Number of elements into the collection once the operation has been executed.

---

### iterator

```java
public ObjectsIterator iterator()
```

Gets an ObjectsIterator.

**Returns:**
ObjectsIterator instance.
**copy**

```java
public Objects copy()
```

Creates a new Objects instance as a copy of the given one.

**Returns:**

The new Objects instance.
### com.sparsity.sparksee.gdb
#### Class ObjectsIterator

java.lang.Object  

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
</table>
| -     | com.sparsity.sparksee.gdb.**ObjectsIterator**  

**All Implemented Interfaces:**
- Iterator, Closeable

---

**public class** **ObjectsIterator**

**extends Object**

**implements Closeable, Iterator**

ObjectsIterator class.

Iterator to traverse all the object identifiers from an Objects instance.

**Author:**
Sparsity Technologies http://www.sparsity-technologies.com

---

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void close()</td>
<td>Closes the ObjectsIterator instance.</td>
</tr>
<tr>
<td>boolean hasNext()</td>
<td>Gets if there are more elements to traverse.</td>
</tr>
<tr>
<td>boolean isClosed()</td>
<td>Gets if ObjectsIterator instance has been closed or not.</td>
</tr>
<tr>
<td>Long next()</td>
<td>See nextObject().</td>
</tr>
<tr>
<td>long nextObject()</td>
<td>Gets the next element.</td>
</tr>
<tr>
<td>void remove()</td>
<td>Operation not supported.</td>
</tr>
</tbody>
</table>

---

**Methods inherited from class** java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

**Methods inherited from interface** java.io.Closeable

close

**Methods inherited from interface** java.util.Iterator

hasNext, next, remove
Methods

nextObject

`public long nextObject()`

Gets the next element.

hasNext

`public boolean hasNext()`

Gets if there are more elements to traverse.

Returns:
TRUE if there are more elements to traverse, FALSE otherwise.

remove

`public void remove()`

Operation not supported.

next

`public Long next()`

See nextObject().

isClosed

`public boolean isClosed()`

Gets if ObjectsIterator instance has been closed or not.

Returns:
TRUE if the ObjectsIterator instance has been closed, FALSE otherwise.

See Also:
`close()`

close

`public void close()`

Closes the ObjectsIterator instance.

It must be called to ensure the integrity of all data.
com.sparsity.sparksee.gdb
Class ObjectType

java.lang.Object
   +-java.lang.Enum
      +-com.sparsity.sparksee.gdb.ObjectType

All Implemented Interfaces:
   Serializable, Comparable

public final class ObjectType
extends Enum

Object type enumeration.

Author:
  Sparsity Technologies http://www.sparsity-technologies.com

Field Summary

|public static final Edge       |
|Node object type.              |
|public static final Node       |
|Node object type.               |

Method Summary

|static ObjectType              |
|valueOf(String name)          |
|static ObjectType[]           |
|values()                      |

Methods inherited from class java.lang.Enum

clone, compareTo, equals, finalize, getDeclaringClass, hashCode, name, ordinal, toString, valueOf

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

Methods inherited from interface java.lang.Comparable

compareTo

Fields
Node
public static final com.sparsity.sparksee.gdb.ObjectType Node

Node object type.

Edge
public static final com.sparsity.sparksee.gdb.ObjectType Edge

Edge object type.

Methods

values
public static ObjectType[] values()
com.sparsity.sparksee.gdb
Class OIDList

java.lang.Object
     +--com.sparsity.sparksee.gdb.OIDList

All Implemented Interfaces:
     Iterable

public class OIDList
extends Object
implements Iterable

Sparksee object identifier list.
It stores a Sparksee object identifier list.

Use OIDListIterator to access all elements into this collection.

Author:
     Sparsity Technologies http://www.sparsity-technologies.com

Constructor Summary

| public | OIDList(int N) Constructor. |
| public | OIDList(long[] list) Creates a new instance from a long array. |
| public | OIDList(Collection col) Creates a new instance from a long collection. |
| public | OIDList() Constructor. |

Method Summary

| void | add(long attr) Adds a Sparksee object identifier at the end of the list. |
| void | clear() Clears the list. |
| int | count() Number of elements in the list. |
| OIDListIterator | iterator() Gets a new OIDListIterator. |
| void | set(int pos, long oid) Sets a Sparksee object identifier at the specified position of the list. |

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface java.lang.Iterable
iterator

Constructors

OIDList
public OIDList(int N)

Constructor.
This creates a list with N invalid oids.

Parameters:
N - null

OIDList
public OIDList(long[] list)

Creates a new instance from a long array.

Parameters:
list - Long array to initialize the instance.

OIDList
public OIDList(Collection col)

Creates a new instance from a long collection.

Parameters:
col - Collection to initialize the instance.

OIDList
public OIDList()

Constructor.
This creates an empty list.

Methods

add
public void add(long attr)
Adds a Sparksee object identifier at the end of the list.

**Parameters:**
- attr - [in] Sparksee object identifier.

### clear

```java
public void clear()
```

Clears the list.

### set

```java
public void set(int pos,
                 long oid)
```

Sets a Sparksee object identifier at the specified position of the list.

**Parameters:**
- pos - [in] List position [0..Count()-1].
- oid - [in] Sparksee object identifier.

### iterator

```java
public OIDListIterator iterator()
```

Gets a new OIDListIterator.

**Returns:**
- OIDListIterator instance.

### count

```java
public int count()
```

Number of elements in the list.

**Returns:**
- Number of elements in the list.
com.sparsity.sparksee.gdb

Class OIDListIterator

java.lang.Object
    +-com.sparsity.sparksee.gdb.OIDListIterator

All Implemented Interfaces:
    Iterator

public class OIDListIterator
extends Object
implements Iterator

OIDList iterator class.

Iterator to traverse all the Sparksee object identifier into a OIDList instance.

Author:
    Sparsity Technologies http://www.sparsity-technologies.com

Method Summary

<table>
<thead>
<tr>
<th>boolean</th>
<th>hasNext()</th>
</tr>
</thead>
</table>
|          | Gets if there are more elements.

<table>
<thead>
<tr>
<th>Long</th>
<th>next()</th>
</tr>
</thead>
</table>
|          | See nextOID().

<table>
<thead>
<tr>
<th>long</th>
<th>nextOID()</th>
</tr>
</thead>
</table>
|          | Gets the next element.

<table>
<thead>
<tr>
<th>void</th>
<th>remove()</th>
</tr>
</thead>
</table>
|          | Operation not supported.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

Methods inherited from interface java.util.Iterator

hasNext, next, remove

Methods

hasNext

public boolean hasNext()

    Gets if there are more elements.
Returns:
TRUE if there are more elements, FALSE otherwise.

remove
public void remove()

Operation not supported.

next
public Long next()

See nextOID().

nextOID
public long nextOID()

Gets the next element.
public final class **Order**
extends Enum

Order enumeration.

**Author:**
Sparsity Technologies http://www.sparsity-technologies.com

### Field Summary

<table>
<thead>
<tr>
<th>public static final Ascendent</th>
<th>Ascendent</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static final Descendent</td>
<td>Descendent</td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>static Order</th>
<th>valueOf(String name)</th>
</tr>
</thead>
<tbody>
<tr>
<td>static Order[]</td>
<td>values()</td>
</tr>
</tbody>
</table>

### Methods inherited from class java.lang.Enum

| clone, compareTo, equals, finalize, getDeclaringClass, hashCode, name, ordinal, toString, valueOf |

### Methods inherited from class java.lang.Object

| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

### Methods inherited from interface java.lang.Comparable

| compareTo |

### Fields
Ascendent

public static final com.sparsity.sparksee.gdb.Order Ascendent

From lower to higher.

Descendent

public static final com.sparsity.sparksee.gdb.Order Descendent

From higher to lower.

Methods

values

public static Order[] values()

valueOf

public static Order valueOf(String name)
**com.sparsity.sparksee.gdb**

**Class Platform**

java.lang.Object

---

public class **Platform**

extends Object

Platform class.

**Author:**

Sparsity Technologies http://www.sparsity-technologies.com

### Method Summary

<table>
<thead>
<tr>
<th>static void</th>
<th>getStatistics(PlatformStatistics stats)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gets platform data and statistics.</td>
<td></td>
</tr>
</tbody>
</table>

### Methods

**getStatistics**

```java
public static void getStatistics(PlatformStatistics stats)
```

Gets platform data and statistics.

**Parameters:**

`stats` - [in][out] This updates the given PlatformStatistics.
public class PlatformStatistics extends Object

Platform data and statistics.

Author: Sparsity Technologies http://www.sparsity-technologies.com

### Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
<td>PlatformStatistics()</td>
</tr>
<tr>
<td></td>
<td>Creates a new instance setting all values to 0.</td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>long getAvailableMem()</td>
<td>Gets available (free) memory size in Bytes.</td>
</tr>
<tr>
<td>int getNumCPUs()</td>
<td>Gets the number of CPUs.</td>
</tr>
<tr>
<td>long getRealTime()</td>
<td>Gets time in microseconds (since epoch).</td>
</tr>
<tr>
<td>long getSystemTime()</td>
<td>Gets CPU system time.</td>
</tr>
<tr>
<td>long getTotalMem()</td>
<td>Gets physical memory size in Bytes.</td>
</tr>
<tr>
<td>long getUserTime()</td>
<td>Gets CPU user time.</td>
</tr>
</tbody>
</table>

### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

### Constructors

**PlatformStatistics**

<table>
<thead>
<tr>
<th></th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
<td>PlatformStatistics()</td>
</tr>
<tr>
<td></td>
<td>Creates a new instance setting all values to 0.</td>
</tr>
</tbody>
</table>
Creates a new instance setting all values to 0.

## Methods

### getRealTime

**public long getRealTime()**

Gets time in microseconds (since epoch).

**Returns:**
- Time in microseconds (since epoch).

### getAvailableMem

**public long getAvailableMem()**

Gets available (free) memory size in Bytes.

**Returns:**
- Available (free) memory size in Bytes.

### getTotalMem

**public long getTotalMem()**

Gets physical memory size in Bytes.

**Returns:**
- Physical memory size in Bytes.

### getSystemTime

**public long getSystemTime()**

Gets CPU system time.

**Returns:**
- CPU system time.

### getUserTime

**public long getUserTime()**

Gets CPU user time.

**Returns:**
-
getNumCPUs

public int getNumCPUs()

Gets the number of CPUs.

**Returns:**

The number of CPUs.
public class Session
extends Object
implements Closeable

Session class.

A Session is a stateful period of activity of a user with the Database.

All the manipulation of a Database must be enclosed into a Session. A Session can be initiated from a Database instance and allows for getting a Graph instance which represents the persistent graph (the graph database).

Also, temporary data is associated to the Session, thus when a Session is closed, all the temporary data associated to the Session is removed too. Objects or Values instances or even session attributes are an example of temporary data.

Moreover, a Session is exclusive for a thread, thus if it is shared among threads results may be fatal or unexpected.

Check out the 'Processing' and 'Transactions' sections in the SPARKSEE User Manual for details about how Sessions work and the use of transactions.

Author: Sparsity Technologies http://www.sparsity-technologies.com

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void begin()</td>
<td>Begins a transaction.</td>
</tr>
<tr>
<td>void close()</td>
<td>Closes the Session instance.</td>
</tr>
<tr>
<td>void commit()</td>
<td>Commits a transaction.</td>
</tr>
<tr>
<td>Graph getGraph()</td>
<td>Gets the Graph instance.</td>
</tr>
<tr>
<td>boolean isClosed()</td>
<td>Gets if Session instance has been closed or not.</td>
</tr>
<tr>
<td>Objects newObjects()</td>
<td>Creates a new Objects instance.</td>
</tr>
</tbody>
</table>

### Methods inherited from class java.lang.Object

- clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

### Methods inherited from interface java.io.Closeable
**Methods**

**isClosed**

```java
public boolean isClosed()
```

Gets if Session instance has been closed or not.

**Returns:**

TRUE if the Session instance has been closed, FALSE otherwise.

**See Also:**

close()

---

**commit**

```java
public void commit()
```

Commits a transaction.

---

**getGraph**

```java
public Graph getGraph()
```

Gets the Graph instance.

**Returns:**

The Graph instance.

---

**close**

```java
public void close()
```

Closes the Session instance.

It must be called to ensure the integrity of all data.

---

**begin**

```java
public void begin()
```

Begins a transaction.

---

**newObjects**

```java
public Objects newObjects()
```
Creates a new Objects instance.

**Returns:**
The new Objects instance.
public class Sparksee extends Object implements Closeable

Sparksee class.

All Sparksee programs must have one single Sparksee instance to manage one or more Database instances.

This class allows for the creation of new Databases or open an existing one.

Author:
Sparsity Technologies http://www.sparsity-technologies.com

### Field Summary

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static</td>
<td>Version</td>
<td>Sparksee version.</td>
</tr>
</tbody>
</table>

### Constructor Summary

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
<td>Sparksee(SparkseeConfig config)</td>
<td>Creates a new instance.</td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>close()</td>
<td>Closes the Sparksee instance.</td>
</tr>
<tr>
<td>Database</td>
<td>create(String path, String alias)</td>
<td>Creates a new Database instance.</td>
</tr>
<tr>
<td>boolean</td>
<td>isClosed()</td>
<td>Gets if Sparksee instance has been closed or not.</td>
</tr>
<tr>
<td>Database</td>
<td>open(String path, boolean read)</td>
<td>Opens an existing Database instance.</td>
</tr>
<tr>
<td>Database</td>
<td>restore(String path, String backupFile)</td>
<td>Restores a Database from a backup file.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

Methods inherited from interface java.io.Closeable
Fields

Version
public static java.lang.String Version
Sparksee version.

Constructors

Sparksee
public Sparksee(SparkseeConfig config)
Creates a new instance.

Parameters:
config - [in] Sparksee configuration.

Methods

create
public Database create(String path,
String alias)
throws FileNotFoundException,
RuntimeException
Creates a new Database instance.

Parameters:
path - [in] Database storage file.
alias - [in] Database alias name.

Returns:
A Database instance.

Throws:
java.io.FileNotFoundException - If the given file cannot be created.
java.lang.RuntimeException - null

restore
public Database restore(String path,
String backupFile)
throws FileNotFoundException,
RuntimeException

Restores a Database from a backup file.

See the Graph class Backup method.

**Parameters:**
- `backupFile` - [in] The Backup file to be restored.

**Returns:**
A Database instance.

**Throws:**
- `java.io.FileNotFoundException` - If the given file cannot be created, or the exported data file does not exists.
- `java.lang.RuntimeException` - null

---

**isClosed**

```java
public boolean isClosed()
```

Gets if Sparksee instance has been closed or not.

**Returns:**
TRUE if the Sparksee instance has been closed, FALSE otherwise.

**See Also:**
- `close()`

---

**close**

```java
public void close()
```

Closes the Sparksee instance.

It must be called to ensure the integrity of all data.

---

**open**

```java
public Database open(String path,
                      boolean read)
```

throws `FileNotFoundException`, `RuntimeException`

Opens an existing Database instance.

**Parameters:**
- `read` - [in] If TRUE, open Database in read-only mode.

**Returns:**
A Database instance.

**Throws:**
- `java.io.FileNotFoundException` - If the given file does not exist.
- `java.lang.RuntimeException` - null
public class SparkseeConfig
extends Object

Sparksee configuration class.

If not specified, 0 means unlimited which is the maximum available. For the pools that's the total cache size. For the cache
unlimited means nearly all the physical memory of the computer.

For each field, there is a default value. This value can be overridden with values from a properties file (see SparkseeProperties
class). Also, this settings can be overridden calling a specific setter.

For each field, it is shown its default value and the property to override this value:

Extent size: 4KB ('sparksee.storage.extentsize' at SparkseeProperties).
Pages per extent: 1 page ('sparksee.storage.extentpages' at SparkseeProperties).
Pool frame size: 1 extent ('sparksee.io.pool.frame.size' at SparkseeProperties).
Minimum size for the persistent pool: 64 frames ('sparksee.io.pool.persistent.minsize' at SparkseeProperties).
Maximum size for the persistent pool: 0 frames ('sparksee.io.pool.persistent.maxsize' at SparkseeProperties).
Maximum size for the temporary pool: 0 frames ('sparksee.io.pool.temporal.maxsize' at SparkseeProperties).
Maximum size for the cache (all pools): 0 MB ('sparksee.io.cache.maxsize' at SparkseeProperties).
License code: "" (sparksee.license' at SparkseeProperties). No license code means evaluation license.
Log level: Info ('sparksee.log.level' at SparkseeProperties).
Log file: "sparksee.log" (sparksee.log.file' at SparkseeProperties).
Cache statistics snapshot time: 1000 msecs [TimeUnit] (sparksee.cache.statisticsSnapshotTime' at SparkseeProperties).
Recovery enabled: false (sparksee.io.recovery' at SparkseeProperties).
Recovery log file: "" (sparksee.io.recovery.logfile' at SparkseeProperties).
Recovery cache max size: 1MB (sparksee.io.recovery.cachessize' at SparkseeProperties).
Recovery checkpoint time: 60 seconds [TimeUnit] (sparksee.io.recovery.checkpointTime' at SparkseeProperties).
High-availability: false (disabled) (sparksee.ha' at SparkseeProperties).
High-availability coordinators: "" (sparksee.ha.coordinators' at SparkseeProperties).
High-availability IP: "" ('sparksee.ha.ip' at SparkseeProperties).

High-availability sync polling: 0 (disabled) [TimeUnit] ('sparksee.ha.sync' at SparkseeProperties).

High-availability master history: 1D (1 day) [TimeUnit] ('sparksee.ha.master.history' at SparkseeProperties).

Use of TimeUnit:

Those variables using TimeUnit allow for:

[D][H][M][S][m][u]

where is a number followed by an optional character which represents the unit: D for days, H for hours, M for minutes, S or s for seconds, m for milliseconds and u for microseconds. If no unit character is given, seconds are assumed.

Author:
Sparsity Technologies http://www.sparsity-technologies.com

---

**Constructor Summary**

<table>
<thead>
<tr>
<th>public</th>
<th>SparkseeConfig()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Creates a new instance.</td>
</tr>
</tbody>
</table>

---

**Method Summary**

<table>
<thead>
<tr>
<th>int</th>
<th>getCacheMaxSize()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the maximum size for the cache (all pools) in MB.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>getCacheStatisticsEnabled()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets whether cache statistics are enabled or disabled.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>getCacheStatisticsFile()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the cache statistics log file.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>long</th>
<th>getCacheStatisticsSnapshotTime()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the cache statistics snapshot time in microseconds.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>getExtentPages()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the number of pages per extent.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>getExtentSize()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the size of a extent.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>getHighAvailabilityCoordinators()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the coordinators address and port list.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>getHighAvailabilityEnabled()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets whether high availability mode is enabled or disabled.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>getHighAvailabilityIP()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the IP address and port of the instance.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>long</th>
<th>getHighAvailabilityMasterHistory()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the master's history log.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>long</th>
<th>getHighAvailabilitySynchronization()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the synchronization polling time.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>getLicense()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the license code.</td>
</tr>
<tr>
<td>Method Type</td>
<td>Method Name</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>String</td>
<td><code>getLogFile()</code></td>
</tr>
<tr>
<td>LogLevel</td>
<td><code>getLogLevel()</code></td>
</tr>
<tr>
<td>int</td>
<td><code>getPoolFrameSize()</code></td>
</tr>
<tr>
<td>int</td>
<td><code>getPoolPersistentMaxSize()</code></td>
</tr>
<tr>
<td>int</td>
<td><code>getPoolPersistentMinSize()</code></td>
</tr>
<tr>
<td>int</td>
<td><code>getPoolTemporaryMaxSize()</code></td>
</tr>
<tr>
<td>int</td>
<td><code>getPoolTemporaryMinSize()</code></td>
</tr>
<tr>
<td>int</td>
<td><code>getRecoveryCacheMaxSize()</code></td>
</tr>
<tr>
<td>long</td>
<td><code>getRecoveryCheckpointTime()</code></td>
</tr>
<tr>
<td>boolean</td>
<td><code>getRecoveryEnabled()</code></td>
</tr>
<tr>
<td>String</td>
<td><code>getRecoveryLogFile()</code></td>
</tr>
<tr>
<td>void</td>
<td><code>setCacheMaxSize(int v)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>setCacheStatisticsEnabled(boolean v)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>setCacheStatisticsFile(String v)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>setCacheStatisticsSnapshotTime(long v)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>setExtentPages(int v)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>setExtentSize(int v)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>setHighAvailabilityCoordinators(String v)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>setHighAvailabilityEnabled(boolean v)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>setHighAvailabilityIP(String v)</code></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>setHighAvailabilityMasterHistory(long v)</td>
<td>Sets the master's history log.</td>
</tr>
<tr>
<td>setHighAvailabilitySynchronization(long v)</td>
<td>Sets the synchronization polling time.</td>
</tr>
<tr>
<td>setLicense(String v)</td>
<td>Sets the license code.</td>
</tr>
<tr>
<td>setLogFile(String v)</td>
<td>Sets the log file.</td>
</tr>
<tr>
<td>setLogLevel(LogLevel v)</td>
<td>Sets the log level.</td>
</tr>
<tr>
<td>setPoolFrameSize(int v)</td>
<td>Sets the size of a pool frame in number of extents.</td>
</tr>
<tr>
<td>setPoolPersistentMaxSize(int v)</td>
<td>Sets the maximum size for the persistent pool in number of frames.</td>
</tr>
<tr>
<td>setPoolPersistentMinSize(int v)</td>
<td>Sets the minimum size for the persistent pool in number of frames.</td>
</tr>
<tr>
<td>setPoolTemporaryMaxSize(int v)</td>
<td>Sets the maximum size for the temporary pool in number of frames.</td>
</tr>
<tr>
<td>setPoolTemporaryMinSize(int v)</td>
<td>Sets the minimum size for the temporary pool in number of frames.</td>
</tr>
<tr>
<td>setRecoveryCacheMaxSize(int v)</td>
<td>Sets the maximum size for the recovery log cache in extents.</td>
</tr>
<tr>
<td>setRecoveryCheckpointTime(long v)</td>
<td>Sets the delay time (in microseconds) between automatic checkpoints.</td>
</tr>
<tr>
<td>setRecoveryEnabled(boolean v)</td>
<td>Enables or disables the recovery.</td>
</tr>
<tr>
<td>setRecoveryLogFile(String v)</td>
<td>Sets the recovery log file.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

Constructors

SparkseeConfig

public SparkseeConfig()

Creates a new instance.

Values are set with default values.
Methods

getHighAvailabilitySynchronization

public long getHighAvailabilitySynchronization()

Gets the synchronization polling time.

Returns:
The Synchronization polling time.

setRecoveryCacheMaxSize

public void setRecoveryCacheMaxSize(int v)

Sets the maximum size for the recovery log cache in extents.

Parameters:
v - [in] The maximum size for the recovery log cache in extents. A 0 sets the default value (extents up to 1MB).

setHighAvailabilityIP

public void setHighAvailabilityIP(String v)

Sets the IP address and port of the instance.

Parameters:
v - [in] The IP address and port of the instance.

setPoolPersistentMinSize

public void setPoolPersistentMinSize(int v)

Sets the minimum size for the persistent pool in number of frames.

Parameters:
v - [in] The minimum size for the persistent pool in number of frames. It must be non-negative.

getExtentSize

public int getExtentSize()

Gets the size of a extent.

Returns:
The size of a extent in KB.
setHighAvailabilityEnabled

public void setHighAvailabilityEnabled(boolean v)

Enables or disables high availability mode.

Parameters:

v - [in] If TRUE this enables high availability mode, if FALSE this disables high availability mode.

setHighAvailabilityCoordinators

public void setHighAvailabilityCoordinators(String v)

Sets the coordinators address and port list.

Parameters:

v - [in] The coordinators address and port list.

getLogFile

public String getLogFile()

Gets the log file.

Returns:

The log file.

getRecoveryCheckpointTime

public long getRecoveryCheckpointTime()

Gets the delay time (in microseconds) between automatic checkpoints.

Returns:

The delay time (in microseconds) between automatic checkpoints.

setCacheStatisticsSnapshotTime

public void setCacheStatisticsSnapshotTime(long v)

Sets the cache statistics snapshot time.

Useless if cache statistics are disabled.

Parameters:

v - [in] The cache statistics snapshot time in microseconds.

getCacheStatisticsEnabled

public boolean getCacheStatisticsEnabled()
Gets whether cache statistics are enabled or disabled.

**Returns:**
TRUE if cache statistics are enabled, FALSE otherwise.

---

**getPoolPersistentMaxSize**

```java
public int getPoolPersistentMaxSize()
```

Gets the maximum size for the persistent pool in number of frames.

**Returns:**
The maximum size for the persistent pool in number of frames.

---

**setLogLevel**

```java
public void setLogLevel(LogLevel v)
```

Sets the log level.

**Parameters:**

---

**setExtentSize**

```java
public void setExtentSize(int v)
```

Sets the size of a pool frame in number of extents.

**Parameters:**
- *v* - [in] The size of a extent size in KB. An extent can have a size between 4KB and 64KB, and it must be a power of 2.

---

**getRecoveryLogFile**

```java
public String getRecoveryLogFile()
```

Gets the recovery log file.

**Returns:**
The recovery log file.

---

**getExtentPages**

```java
public int getExtentPages()
```

Gets the number of pages per extent.
Returns:
The number of pages per extent.

**setPoolPersistentMaxSize**

```java
public void setPoolPersistentMaxSize(int v)
```

Sets the maximum size for the persistent pool in number of frames.

**Parameters:**

- `v` - [in] The maximum size for the persistent pool in number of frames. It must be non-negative.

**setCacheMaxSize**

```java
public void setCacheMaxSize(int v)
```

Sets the maximum size for the cache (all pools) in MB.

**Parameters:**

- `v` - [in] The maximum size for the cache (all pools) in MB. It must be non-negative.

**getHighAvailabilityEnabled**

```java
public boolean getHighAvailabilityEnabled()
```

Gets whether high availability mode is enabled or disabled.

**Returns:**

TRUE if high availability mode is enabled, FALSE otherwise.

**setRecoveryLogFile**

```java
public void setRecoveryLogFile(String v)
```

Sets the recovery log file.

**Parameters:**

- `v` - [in] The recovery log file. Left it empty for the default log file (same as .log)

**getRecoveryEnabled**

```java
public boolean getRecoveryEnabled()
```

Gets whether the recovery is enabled or disabled.

**Returns:**

TRUE if the recovery is enabled, FALSE otherwise.
getHighAvailabilityCoordinators

public String getHighAvailabilityCoordinators()

Gets the coordinators address and port list.

**Returns:**
The coordinators address and port list.

setLicense

public void setLicense(String v)

Sets the license code.

**Parameters:**

- **v** - [in] The license code.

setRecoveryEnabled

public void setRecoveryEnabled(boolean v)

Enables or disables the recovery.

**Parameters:**

- **v** - [in] If TRUE this enables the recovery, if FALSE then disables it.

setPoolTemporaryMaxSize

public void setPoolTemporaryMaxSize(int v)

Sets the maximum size for the temporary pool in number of frames.

**Parameters:**

- **v** - [in] The maximum size for the temporary pool in number of frames. It must be non-negative.

setHighAvailabilitySynchronization

public void setHighAvailabilitySynchronization(long v)

Sets the synchronization polling time.

**Parameters:**

- **v** - [in] The synchronization polling time.

setPoolFrameSize

public void setPoolFrameSize(int v)
Sets the size of a pool frame in number of extents.

Parameters:
  v - [in] The size of a pool frame in number of extents. It must be non-negative.

getCacheStatisticsFile
public String getCacheStatisticsFile()

  Gets the cache statistics log file.
  Useless if cache statistics are disabled.

  Returns:
  The cache statistics log file.

setCacheStatisticsFile
public void setCacheStatisticsFile(String v)

  Sets the cache statistics log file.
  Useless if cache statistics are disabled.

  Parameters:

getPoolFrameSize
public int getPoolFrameSize()

  Gets the size of a pool frame in number of extents.

  Returns:
  The size of a pool frame in number of extents.

getCacheMaxSize
public int getCacheMaxSize()

  Gets the maximum size for the cache (all pools) in MB.

  Returns:
  The maximum size for the cache (all pools) in MB.

getCacheStatisticsSnapshotTime
public long getCacheStatisticsSnapshotTime()

  Gets the cache statistics snapshot time in microseconds.
  Useless if cache statistics are disabled.

  Returns:
The cache statistics snapshot time in microseconds.

**getPoolTemporaryMaxSize**

```java
public int getPoolTemporaryMaxSize()
```

Gets the maximum size for the temporary pool in number of frames.

**Returns:**

The maximum size for the temporary pool in number of frames.

**setExtentPages**

```java
public void setExtentPages(int v)
```

Sets the number of pages per extent.

**Parameters:**

- `v` - [in] The number of pages. It must be at least 1 page and the page size must be greater than or equal to 4KB.

**setPoolTemporaryMinSize**

```java
public void setPoolTemporaryMinSize(int v)
```

Sets the minimum size for the temporary pool in number of frames.

**Parameters:**

- `v` - [in] The minimum size for the temporary pool in number of frames. It must be non-negative.

**getLicense**

```java
public String getLicense()
```

Gets the license code.

**Returns:**

The license code.

**getPoolTemporaryMinSize**

```java
public int getPoolTemporaryMinSize()
```

Gets the minimum size for the temporary pool in number of frames.

**Returns:**

The minimum size for the temporary pool in number of frames.
setHighAvailabilityMasterHistory
public void setHighAvailabilityMasterHistory(long v)

Sets the master's history log.

Parameters:
  v - [in] The master's history log.

setCacheStatisticsEnabled
public void setCacheStatisticsEnabled(boolean v)

Enables or disables cache statistics.

Parameters:
  v - [in] If TRUE this enables cache statistics, if FALSE this disables cache statistics.

getRecoveryCacheMaxSize
public int getRecoveryCacheMaxSize()

Gets the maximum size for the recovery log cache in extents.

Returns:
  The maximum size for the recovery log cache in extents.

getHighAvailabilityMasterHistory
public long getHighAvailabilityMasterHistory()

Gets the master's history log.

Returns:
  The master's history log.

gGetHighAvailabilityIP
public String getHighAvailabilityIP()

Gets the IP address and port of the instance.

Returns:
  The IP address and port of the instance.

getLogLevel
public LogLevel getLogLevel()
Gets the log level.

**Returns:**
The LogLevel.

---

### `setRecoveryCheckpointTime`

**public void setRecoveryCheckpointTime(long v)**

Sets the delay time (in microseconds) between automatic checkpoints.

**Parameters:**
- v - [in] The delay time (in microseconds) between automatic checkpoints. A 0 forces a checkpoint after each committed transaction.

---

### `setLogFile`

**public void setLogFile(String v)**

Sets the log file.

**Parameters:**

---

### `getPoolPersistentMinSize`

**public int getPoolPersistentMinSize()**

Gets the minimum size for the persistent pool in number of frames.

**Returns:**
The minimum size for the persistent pool in number of frames.
public class SparkseeProperties extends Object

Sparksee properties file.

This class is implemented as a singleton, so all public methods are static.

It allows for getting the property values stored in a properties file. A properties file is a file where there is one line per property. A property is defined by a key and a value as follows: key=value

By default, this loads properties from the file './sparksee.cfg'. The user may choose to load a different file by calling the method Load().

If the default properties file or the one loaded by the user do not exist, then this behaves as loading an empty properties file.

Method Summary

<table>
<thead>
<tr>
<th>static String</th>
<th>get(String key, String def)</th>
<th>Gets a property.</th>
</tr>
</thead>
<tbody>
<tr>
<td>static boolean</td>
<td>getBoolean(String key, boolean def)</td>
<td>Gets a property as a boolean.</td>
</tr>
<tr>
<td>static int</td>
<td>getInteger(String key, int def)</td>
<td>Gets a property as an integer.</td>
</tr>
<tr>
<td>static long</td>
<td>getTimeUnit(String key, long def)</td>
<td>Gets a property as a time unit.</td>
</tr>
<tr>
<td>static void</td>
<td>load(String path)</td>
<td>Loads properties from the given file path.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

Methods

get

public static String get(String key, String def)

Gets a property.
Parameters:
  key - [in] The name of the property to lookup.
  def - [in] Default value to be returned in case there is no property with the name key.

Returns:
The value of the property, or def if the key is not found.

getTimeUnit

```java
public static long getTimeUnit(String key, long def)
```

Gets a property as a time unit.

A time unit is a string representation of a time duration with a time unit such as '10s' or '3H'.

Valid format for the string representation: Blanks at the beginning or at the end are ignored. No blanks are allowed between the time duration and the unit time.

Allowed time units: 'D' for days, 'H' for hours, 'M' for minutes, 'S' o 's' for seconds, 'm' for milliseconds and 'u' for microseconds.

There is a special case: If no time unit is given, seconds is the default. So, '10' means 10 seconds.

Parameters:
  key - [in] The name of the property to lookup.
  def - [in] The default value (in microseconds) to be returned in case there is no property with the name key.

Returns:
The time duration in microseconds, or def if the key is not found or in case of error.

getBoolean

```java
public static boolean getBoolean(String key, boolean def)
```

Gets a property as a boolean.

Parameters:
  key - [in] The name of the property to lookup.
  def - [in] Default value to be returned in case there is no property with the name key.

Returns:
The property value, or def if the key is not found or in case of error.

load

```java
public static void load(String path)
```

Loads properties from the given file path.

Parameters:
  path - [in] File path to load properties from.
getInteger

public static int getInteger(String key,
    int def)

Gets a property as an integer.

Parameters:
   key - [in] The name of the property to lookup.
   def - [in] Default value to be returned in case there is no property with the name key.

Returns:
   The property value, or def if the key is not found or in case of error.
com.sparsity.sparksee.gdb
Class StringList

java.lang.Object
   - com.sparsity.sparksee.gdb.StringList

All Implemented Interfaces:
   Iterable

public class StringList
extends Object
implements Iterable

String list.
It stores a String (unicode) list.

Use StringListIterator to access all elements into this collection.

Author:
Sparsity Technologies http://www.sparsity-technologies.com

Constructor Summary

| public | StringList(Collection col) | Creates a new instance from an string collection. |
| public | StringList() | Constructor. |
| public | StringList(String[] list) | Creates a new instance from an string array. |

Method Summary

| void | add(String str) | Adds a String at the end of the list. |
| void | clear() | Clears the list. |
| int | count() | Number of elements in the list. |

StringListIterator | iterator() | Gets a new StringListIterator. |

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait,

Methods inherited from interface java.lang.Iterable

iterator
Constructors

StringList

public StringList(Collection col)

Creates a new instance from an string collection.

Parameters:

  col - Collection to initialize the instance.

StringList

public StringList()

Constructor.
This creates an empty list.

StringList

public StringList(String[] list)

Creates a new instance from an string array.

Parameters:

  list - String array to initialize the instance.

Methods

clear

public void clear()

Clears the list.

iterator

public StringListIterator iterator()

Gets a new StringListIterator.

Returns:

  StringListIterator instance.

count

public int count()
Number of elements in the list.

**Returns:**
Number of elements in the list.

---

**add**

```java
public void add(String str)
```

Adds a String at the end of the list.

**Parameters:**

com.sparsity.sparksee.gdb
Class StringListIterator

java.lang.Object
    +-com.sparsity.sparksee.gdb.StringListIterator

All Implemented Interfaces:
    Iterator

public class StringListIterator
extends Object
implements Iterator

StringList iterator class.
Iterator to traverse all the strings into a StringList instance.

Author:
    Sparsity Technologies http://www.sparsity-technologies.com

Method Summary

<table>
<thead>
<tr>
<th>boolean</th>
<th>hasNext()</th>
<th>Gets if there are more elements.</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>next()</td>
<td>See nextString().</td>
</tr>
<tr>
<td>String</td>
<td>nextString()</td>
<td>Gets the next element.</td>
</tr>
<tr>
<td>void</td>
<td>remove()</td>
<td>Operation not supported.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

Methods inherited from interface java.util.Iterator

hasNext, next, remove

Methods

hasNext

public boolean hasNext()  
Gets if there are more elements.
Returns:
TRUE if there are more elements, FALSE otherwise.

remove
public void remove()
Operation not supported.

next
public String next()
See nextString().

nextString
public String nextString()
Gets the next element.
public class TextStream extends Object implements Closeable

TextStream class.

It allows for reading and writing Text attribute values.

It is very important to close the stream once no more reading or writing operations will be performed to ensure data is successfully stored.

Whereas string attributes are set and got using the Value class, text attributes are operated using a stream pattern.

Use of TextStream for writing: (i) Create a TextStream instance and (ii) set the stream for a text attribute of a node or edge instance with the graph SetAttributeText method. Once the set attribute text has been done, (iii) perform as many write operations as you need to the TextStream instance. Lastly, (iv) execute Close to flush and close the stream.

Use of TextStream for reading: (i) Get the stream of a text attribute of a node or edge instance with the GetAttributeText graph method. Once you have the TextStream instance, (ii) you can execute Read operations to read from the stream. (iii) The end of the stream is reached when Read returns 0. Finally, (iv) execute Close to close stream resources.

Check out the 'Attributes and values' section in the SPARKSEE User Manual for more details on this.

Author: Sparsity Technologies http://www.sparsity-technologies.com
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

Methods inherited from interface java.io.Closeable

close

Constructors

TextStream

public TextStream(boolean append)

Creates a new instance.

A TextStream only can be created by the user to write data.

Parameters:

append - [in] If TRUE, the it is created in append mode to write from the end of the stream, otherwise it is created to write from the beginning of the stream.

Methods

read

public int read(char[] dataOUT,
int length)

Read data from the stream.

Parameters:

dataOUT - [out] Buffer to read data to. It must be allocated by the user.
length - [in] Length of the given data buffer. It must be > 0.

Returns:

Amount of read data (<= length). If 0, there is no more data to be read from the stream.

isNull

public boolean isNull()

Returns TRUE if the stream is not available.

It returns for reading or writing data.

Returns:

FALSE if the stream is ready

write

public void write(char[] dataIN,
int length)
Write data to the stream.

**Parameters:**
- `dataIN` - [in] Buffer to write data from.
- `length` - [in] Length of the data buffer. It must be > 0.

### close

```java
public void close()
```

Closes the stream.

Once the Stream is closed, it cannot be used again.
com.sparsity.sparksee.gdb
Class Type

public class Type
extends Object

Type data class.

It contains information about a node or edge type.

Author:
Sparsity Technologies http://www.sparsity-technologies.com

Field Summary

<table>
<thead>
<tr>
<th>public static</th>
<th>GlobalType</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Global type identifier constant.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public static</th>
<th>InvalidType</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Invalid type identifier constant.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>boolean</th>
<th>getAreNeighborsIndexed()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets if this is an edge type with neighbors index.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>getId()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the Sparksee type identifier.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>getIsDirected()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets if this is a directed edge type.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>getIsRestricted()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets if this is a restricted edge type.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>getName()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the unique type name.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>long</th>
<th>getNumObjects()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the number of objects belonging to the type.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ObjectType</th>
<th>getObjectType()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the object type.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>getRestrictedFrom()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the tail or source type identifier for restricted edge types.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>getRestrictedTo()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the head or target type identifier for restricted edge types.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait,
wait

Fields

GlobalType

public static int GlobalType

Global type identifier constant.

InvalidType

public static int InvalidType

Invalid type identifier constant.

Methods

getRestrictedFrom

public int getRestrictedFrom()

Gets the tail or source type identifier for restricted edge types.

Returns:
For restricted edge types, the tail or source type identifier, the Type InvalidType otherwise.

getAreNeighborsIndexed

public boolean getAreNeighborsIndexed()

Gets if this is an edge type with neighbors index.

Returns:
TRUE for edges types with neighbors index, FALSE otherwise.

getObjectType

public ObjectType getObjectType()

Gets the object type.

Returns:
The object type.
getRestrictedTo

public int getRestrictedTo()

Gets the head or target type identifier for restricted edge types.

Returns:
For restricted edge types, the head or target type identifier, the Type InvalidType otherwise.

getIsRestricted

public boolean getIsRestricted()

Gets if this is a restricted edge type.

Returns:
TRUE for restricted edge types, FALSE otherwise.

getNumObjects

public long getNumObjects()

Gets the number of objects belonging to the type.

Returns:
The number of objects belonging to the type.

getId

public int getId()

Gets the Sparksee type identifier.

Returns:
The Sparksee type identifier.

getIsDirected

public boolean getIsDirected()

Gets if this is a directed edge type.

Returns:
TRUE for directed edge types, FALSE otherwise.

getName

public String getName()
Gets the unique type name.

**Returns:**

The unique type name.
com.sparsity.sparksee.gdb
Class TypeList

java.lang.Object
   +-com.sparsity.sparksee.gdb.TypeList

All Implemented Interfaces:
   Iterable

public class TypeList
extends Object
implements Iterable

Sparksee type identifier list.
It stores a Sparksee node or edge type identifier list.

Use TypeListIterator to access all elements into this collection.

Author:
   Sparsity Technologies http://www.sparsity-technologies.com

Constructor Summary

<table>
<thead>
<tr>
<th>public</th>
<th>TypeList(int[] list)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Creates a new instance from an integer array.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public</th>
<th>TypeList(Collection col)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Creates a new instance from an integer collection.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public</th>
<th>TypeList()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Constructor.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>void</th>
<th>add(int type)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Adds a Sparksee type identifier at the end of the list.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>clear()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Clears the list.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>count()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Number of elements in the list.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TypeListIterator</th>
<th>iterator()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets a new TypeListIterator.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface java.lang.Iterable

iterator
Constructors

**TypeList**

public **TypeList**(int[] list)

Creates a new instance from an integer array.

**Parameters:**

- list - Integer array to initialize the instance.

**TypeList**

public **TypeList**(Collection col)

Creates a new instance from an integer collection.

**Parameters:**

- col - Collection to initialize the instance.

**TypeList**

public **TypeList**()

Constructor.

This creates an empty list.

Methods

**add**

public void **add**(int type)

Adds a Sparksee type identifier at the end of the list.

**Parameters:**

- type - [in] Sparksee type identifier.

**clear**

public void **clear**()

Clears the list.

**iterator**

public **TypeListIterator** **iterator**()
Gets a new TypeListIterator.

**Returns:**
TypeListIterator instance.

count

```java
public int count()
```

Number of elements in the list.

**Returns:**
Number of elements in the list.
public class TypeListIterator extends Object implements Iterator

TypeList iterator class.

Iterator to traverse all the Sparksee node or edge type identifiers into a TypeList instance.

Author:
Sparsity Technologies http://www.sparsity-technologies.com

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hasNext</td>
<td>Gets if there are more elements.</td>
</tr>
<tr>
<td>next</td>
<td>See nextType().</td>
</tr>
<tr>
<td>nextType</td>
<td>Gets the next element.</td>
</tr>
<tr>
<td>remove</td>
<td>Operation not supported.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

Methods inherited from interface java.util.Iterator

hasNext, next, remove

Methods

hasNext

public boolean hasNext()

Gets if there are more elements.
Returns:
TRUE if there are more elements, FALSE otherwise.

remove
public void remove()

Operation not supported.

next
public Integer next()

See nextType().

nextType
public int nextType()

Gets the next element.
com.sparsity.sparksee.gdb

Class Value

(java.lang.Object
   +com.sparsity.sparksee.gdb.Value

public class Value
extends Object

Value class.

It is a container which stores a value and its data type (domain). A Value can be NULL.

Author:
   Sparsity Technologies http://www.sparsity-technologies.com

Field Summary

<table>
<thead>
<tr>
<th>public static</th>
<th>MaxLengthString</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Maximum number of characters allowed for a String.</td>
</tr>
</tbody>
</table>

Constructor Summary

<table>
<thead>
<tr>
<th>public</th>
<th>Value(Value v)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Copy constructor.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public</th>
<th>Value()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Creates a new instance.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>int</th>
<th>compare(Value v)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Compares with the given Value.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>compareTo(Object v)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>See compare().</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>compareTo(Value v)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>See compare().</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>equals(Object other)</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>equals(Value v)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Compares with the given Value.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>getBoolean()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets Boolean Value.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DataType</th>
<th>getDataType()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the DataType.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>double</th>
<th>getDouble()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets Double Value.</td>
</tr>
<tr>
<td>Type</td>
<td>Method</td>
</tr>
<tr>
<td>--------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>int</td>
<td><code>getInteger()</code></td>
</tr>
<tr>
<td>long</td>
<td><code>getLong()</code></td>
</tr>
<tr>
<td>long</td>
<td><code>getOID()</code></td>
</tr>
<tr>
<td>String</td>
<td><code>getString()</code></td>
</tr>
<tr>
<td>long</td>
<td><code>getTimestamp()</code></td>
</tr>
<tr>
<td>Calendar</td>
<td><code>getTimestampAsCalendar()</code></td>
</tr>
<tr>
<td>Date</td>
<td><code>getTimestampAsDate()</code></td>
</tr>
<tr>
<td>int</td>
<td><code>hashCode()</code></td>
</tr>
<tr>
<td>boolean</td>
<td><code>isNull()</code></td>
</tr>
<tr>
<td>Value</td>
<td><code>set(Value v)</code></td>
</tr>
<tr>
<td>Value</td>
<td><code>setBoolean(boolean v)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>setBooleanVoid(boolean v)</code></td>
</tr>
<tr>
<td>Value</td>
<td><code>setDouble(double v)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>setDoubleVoid(double v)</code></td>
</tr>
<tr>
<td>Value</td>
<td><code>setInteger(int v)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>setIntegerVoid(int v)</code></td>
</tr>
<tr>
<td>Value</td>
<td><code>setLong(long v)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>setLongVoid(long v)</code></td>
</tr>
<tr>
<td>Value</td>
<td><code>setNull()</code></td>
</tr>
<tr>
<td>void</td>
<td><code>setNullVoid()</code></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td><code>setOID</code> (long v)</td>
<td>Sets the Value.</td>
</tr>
<tr>
<td><code>setOIDVoid</code> (long v)</td>
<td>Sets the OID Value.</td>
</tr>
<tr>
<td><code>setString</code> (String v)</td>
<td>Sets the Value.</td>
</tr>
<tr>
<td><code>setStringVoid</code> (String v)</td>
<td>Sets the Value.</td>
</tr>
<tr>
<td><code>setText</code> (TextStream v)</td>
<td>Sets the Value.</td>
</tr>
<tr>
<td><code>setTimestamp</code> (Calendar v)</td>
<td>Sets the Value.</td>
</tr>
<tr>
<td><code>setTimestamp</code> (Date v)</td>
<td>Sets the Value.</td>
</tr>
<tr>
<td><code>setTimestamp</code> (int year, int month, int day, int hour, int minutes, int seconds, int millisec)</td>
<td>Sets the Value.</td>
</tr>
<tr>
<td><code>setTimestampVoid</code> (int year, int month, int day, int hour, int minutes, int seconds, int millisecs)</td>
<td>Sets the Value.</td>
</tr>
<tr>
<td><code>setTimestampVoid</code> (long v)</td>
<td>Sets the Value.</td>
</tr>
<tr>
<td><code>setVoid</code> (Value v)</td>
<td>Sets the Value.</td>
</tr>
<tr>
<td><code>toString</code> ()</td>
<td>Gets a String representation of the Value.</td>
</tr>
<tr>
<td><code>toString</code> (String str)</td>
<td>Gets a string representation of the Value.</td>
</tr>
</tbody>
</table>

Methods inherited from class `java.lang.Object`:
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

### Fields

**MaxLengthString**

public static int **MaxLengthString**

Maximum number of characters allowed for a String.

### Constructors
Value
public Value(Value v)

    Copy constructor.

Parameters:
    v - [in] Value to be copied.

Value
public Value()

    Creates a new instance.
    It creates a NULL Value.

Methods

setLongVoid
public void setLongVoid(long v)

    Sets the Value.

Parameters:
    v - [in] New Long value.

gETimestampAsCalendar
public Calendar getTimestampAsCalendar()

    Gets the Value as a Calendar instance.

Returns:
    The returning Calendar instance.

equals
public boolean equals(Object other)

Parameters:
    other - null

getOID
public long getOID()
Gets OID Value.
This must be an non-NULL OID Value.

**Returns:**
The OID Value.

---

**set**

```java
public Value set(Value v)
```

Sets the Value.

**Parameters:**
- `v` - New value.

**Returns:**
The calling instance.

---

**setString**

```java
public Value setString(String v)
```

Sets the Value.

**Parameters:**
- `v` - New value.

**Returns:**
The calling instance.

---

**setBooleanVoid**

```java
public void setBooleanVoid(boolean v)
```

Sets the Value.

**Parameters:**
- `v` - [in] New Boolean value.

---

**hashCode**

```java
public int hashCode()
```

---

**compare**

```java
public int compare(Value v)
```
Compares with the given Value.

It does not work for Text or if given Value objects does not have the same DataType.

**Parameters:**

- v - Given value to compare to.

**Returns:**

0 if this Value is equal to the given one; a value less than 0 if this Value is less than the given one; and a value greater than 0 if this Value is greater than the given one.

---

### setDoubleVoid

**public void setDoubleVoid(double v)**

Sets the Value.

**Parameters:**

- v - [in] New Double value.

### setText

**public Value setText(TextStream v)**

Sets the Value.

**Parameters:**

- v - New value.

**Returns:**

The calling instance.

### setBoolean

**public Value setBoolean(boolean v)**

Sets the Value.

**Parameters:**

- v - New value.

**Returns:**

The calling instance.

### setTimestamp

**public Value setTimestamp(Calendar v)**

Sets the Value.

**Parameters:**

- v - New value.
Returns:
The calling instance.

getBoolean

public boolean getBoolean()

Gets Boolean Value.

This must be a non-NULL Boolean Value.

Returns:
The Boolean Value.

setTimestampVoid

public void setTimestampVoid(long v)

Sets the Value.

Parameters:

setDouble

public Value setDouble(double v)

Sets the Value.

Parameters:
  v - New value.

Returns:
The calling instance.

compareTo

public int compareTo(Value v)

See compare().

Parameters:
  v - null

setNullVoid

public void setNullVoid()

Sets the Value to NULL.
setOIDVoid
public void setOIDVoid(long v)

Sets the OID Value.

Parameters:
  v - [in] New OID value.

getDataType
public DataType getDataType()

Gets the DataType.

Value cannot be NULL.

Returns:
The DataType.

setLong
public Value setLong(long v)

Sets the Value.

Parameters:
  v - New value.

Returns:
The calling instance.

getTimestampAsDate
public Date getTimestampAsDate()

Gets the Value as a Date instance.

Returns:
The returning Date instance.

getString
public String getString()

Gets String Value.

This must be a non-NULL String Value.

Returns:
The String Value.
isNull

public boolean isNull()

    Gets if this is a NULL Value.

    Returns:
    TRUE if this is a NULL Value, FALSE otherwise.

setTimestamp

public Value setTimestamp(int year,
    int month,
    int day,
    int hour,
    int minutes,
    int seconds,
    int millisec)

Sets the Value.

Parameters:
    year - The year (>=1970).
    month - The month ([1..12]).
    day - The day of the month ([1..31]).
    hour - The hour ([0..23]).
    minutes - The minutes ([0..59]).
    seconds - The seconds ([0..59]).
    millisec - The milliseconds ([0..999]).

Returns:
    The calling instance.

setTimestampVoid

public void setTimestampVoid(int year,
    int month,
    int day,
    int hour,
    int minutes,
    int seconds,
    int millisecs)

Sets the Value.

Parameters:
    month - [in] The month ([1..12]).
    day - [in] The day of the month ([1..31]).
    hour - [in] The hour ([0..23]).
    minutes - [in] The minutes ([0..59]).
    seconds - [in] The seconds ([0..59]).
    millisecs - [in] The milliseconds ([0..999]).
compareTo

public int compareTo(Object v)

See compare().

This just works if the given object is a Value instance.

Parameters:
   v - null

equals

public boolean equals(Value v)

Compares with the given Value.

It does not work for Text or if given Value objects does not have the same DataType.

Parameters:
   v - Given value to compare to.

Returns:
   TRUE if this Value is equal to the given one; FALSE otherwise.

getDouble

public double getDouble()

Gets Double Value.

This must be a non-NULL Double Value.

Returns:
   The Double Value.

setIntegerVoid

public void setIntegerVoid(int v)

Sets the Value.

Parameters:
   v - [in] New Integer value.

toString

public String toString(String str)

Gets a string representation of the Value.

It does not work for Text.

Parameters:
   str - String to be used. It is cleared and set with the string representation of the Value.

Returns:
   The given string which has been updated.
**setVoid**

```java
public void setVoid(Value v)
```

Sets the Value.

**Parameters:**

v - [in] New value.

---

**setNull**

```java
public Value setNull()
```

Sets the Value to NULL.

**Returns:**

The calling instance.

---

**setOID**

```java
public Value setOID(long v)
```

Sets the Value.

**Parameters:**

v - New value.

**Returns:**

The calling instance.

---

**setStringVoid**

```java
public void setStringVoid(String v)
```

Sets the Value.

**Parameters:**

v - [in] New String value.

---

**getLong**

```java
public long getLong()
```

Gets Long Value.

This must be a non-NULL Long Value.

**Returns:**

The Long Value.
**toString**

```java
public String toString()
```

Gets a String representation of the Value.

It does not work for Text Value instances.

---

**getTimestamp**

```java
public long getTimestamp()
```

Gets Timestamp Value.

This must be a non-NULL Timestamp Value.

**Returns:**

The Timestamp Value.

---

**setInteger**

```java
public Value setInteger(int v)
```

Sets the Value.

**Parameters:**

- v - New value.

**Returns:**

The calling instance.

---

**setTimestamp**

```java
public Value setTimestamp(Date v)
```

Sets the Value.

**Parameters:**

- v - New value.

**Returns:**

The calling instance.

---

**getInteger**

```java
public int getInteger()
```

Gets Integer Value.

This must be a non-NULL Integer Value.

**Returns:**

The Integer Value.
com.sparsity.sparksee.gdb
Class Values

public class Values
extends Object
implements Iterable, Closeable

Value set class.
This is a set of Value instances, that is there is no duplicated elements.
Use a ValuesIterator to traverse all the elements into the set.
When the Values instance is closed, it closes all existing and non-closed ValuesIterator instances too.

Author:
Sparsity Technologies http://www.sparsity-technologies.com

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void close()</td>
<td>Closes the Values instance.</td>
</tr>
<tr>
<td>long count()</td>
<td>Gets the number of elements into the collection.</td>
</tr>
<tr>
<td>boolean isClosed()</td>
<td>Gets if Values instance has been closed or not.</td>
</tr>
<tr>
<td>ValuesIterator iterator(Order order)</td>
<td>Gets a ValuesIterator.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

Methods inherited from interface java.lang.Iterable
iterator

Methods inherited from interface java.io.Closeable
close
Methods

**iterator**

```java
public ValuesIterator iterator()
```

See iterator().

Creates an Ascendent iterator.

**count**

```java
public long count()
```

Gets the number of elements into the collection.

**Returns:**
The number of elements into the collection.

**iterator**

```java
public ValuesIterator iterator(Order order)
```

Gets a ValuesIterator.

**Parameters:**
order - [in] Ascending or descending order.

**Returns:**
ValuesIterator instance.

**isClosed**

```java
public boolean isClosed()
```

Gets if Values instance has been closed or not.

**Returns:**
TRUE if the Values instance has been closed, FALSE otherwise.

**See Also:**
`close()`

**close**

```java
public void close()
```

Closes the Values instance.

It must be called to ensure the integrity of all data.
Class ValuesIterator

public class ValuesIterator extends Object implements Closeable, Iterator

Values iterator class.

It allows for traversing all the elements into a Values instance.

Author:
Sparsity Technologies http://www.sparsity-technologies.com

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>close()</td>
<td>Closes the ValuesIterator instance.</td>
</tr>
<tr>
<td>hasNext()</td>
<td>Gets if there are more elements to traverse.</td>
</tr>
<tr>
<td>isClosed()</td>
<td>Gets if ValuesIterator instance has been closed or not.</td>
</tr>
<tr>
<td>next()</td>
<td>Gets the next element to traverse.</td>
</tr>
<tr>
<td>remove()</td>
<td>Operation not supported.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

Methods inherited from interface java.io.Closeable

close

Methods inherited from interface java.util.Iterator

hasNext, next, remove

Methods
hasNext
public boolean hasNext()

Gets if there are more elements to traverse.

Returns:
TRUE if there are more elements to traverse, FALSE otherwise.

remove
public void remove()

Operation not supported.

next
public Value next()

Gets the next element to traverse.

Returns:
The next element.

isClosed
public boolean isClosed()

Gets if ValuesIterator instance has been closed or not.

Returns:
TRUE if the ValuesIterator instance has been closed, FALSE otherwise.

See Also:
close()

close
public void close()

Closes the ValuesIterator instance.

It must be called to ensure the integrity of all data.
Package

com.sparsity.sparksee.io
public class CSVReader
extends RowReader

CSVReader interface.
A very simple CSV reader.
It works as any other RowReader, but open must be called once before the first read operation.
Using the format RFC 4180.
Except: leading and trailing spaces, adjacent to CSV separator character, are trimmed.
You can use your own separators and quote characters. By default the separator is the comma (,) and the quote character is the double quotes (").
Fields with multiple lines can be allowed (and the maximum lines specified), but the default is a single line.
The locale string can be used to set the language, country and the file encoding. The format must be "[language_territory][.codeset]". But only the file encoding is being used in the current version.
The languages supported are: "en_US", "es_ES" and "ca_ES".
The file encodings supported are: "utf8" and "iso88591".
For example:
To don't change the default locales, use an empty string: "".
To read a file in utf8 with the default language settings use ".utf8".
To read a file in iso88591 with English language use: "en_US.iso88591".
Check out the 'Data import' section in the SPARKSEE User Manual for more details on this.

Author:
Sparsity Technologies http://www.sparsity-technologies.com

Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public CSVReader()</td>
<td>Constructs CSVReader.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void close()</td>
<td>Closes the reader.</td>
</tr>
<tr>
<td>int getRow()</td>
<td>The row number for the current row.</td>
</tr>
</tbody>
</table>
### Methods

<table>
<thead>
<tr>
<th>void</th>
<th>open(String f)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Opens the source file path.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>read(StringList row)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Reads the next row as a string array.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>reset()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Moves the reader to the beginning.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>setLocale(String localeStr)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sets the locale that will be used to read the file.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>setMultilines(int numExtralines)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Allows the use of fields with more than one line.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>setNumLines(int numLines)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Used to limit the number of lines that will be read.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>setQuotes(String quotes)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sets the character used to quote fields.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>setSeparator(String sep)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sets the character used to separate fields in the file.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>setSingleLine()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Only allows single line fields.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>setStartLine(int startLine)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sets the number of lines to be skipped from the beginning.</td>
</tr>
</tbody>
</table>

Methods inherited from class `com.sparsity.sparksee.io.RowReader`

- close, getRow, read, reset

Methods inherited from class `java.lang.Object`

- clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### Constructors

**CSVReader**

```java
public CSVReader()
```

Constructs CSVReader.

### Methods

**reset**

```java
public boolean reset()
```

throws IOException
Moves the reader to the beginning.
Restarts the reader.

**Returns:**
true if the reader can be restarted, false otherwise.

**Throws:**
java.io.IOException - If bad things happen during the restart.

---

**close**

```java
close()
```

throws IOException

Closes the reader.

---

**setNumLines**

```java
setNumLines(int numLines)
```

Used to limit the number of lines that will be read.

**Parameters:**

- numLines - [in] The maximum number of lines to read (0 == unlimited)

---

**open**

```java
open(String f)
```

throws IOException

Opens the source file path.

**Parameters:**

- f - [in] CSV file path.

**Throws:**

java.io.IOException - If bad things happen opening the file.

---

**setMultilines**

```java
setMultilines(int numExtralines)
```

Allows the use of fields with more than one line.

**Parameters:**

- numExtralines - [in] Maximum number of extra lines for each column (0==unlimited, N==N+1 total rows).

---

**setSeparator**

```java
setSeparator(String sep)
```

throws RuntimeException


Sets the character used to separate fields in the file.

**Parameters:**

**Throws:**
- `java.lang.RuntimeException` - null

### setQuotes

```java
public void setQuotes(String quotes) throws RuntimeException
```

Sets the character used to quote fields.

**Parameters:**
- *quotes* - [in] Quote character.

**Throws:**
- `java.lang.RuntimeException` - null

### setSingleLine

```java
public void setSingleLine()
```

Only allows single line fields.

### read

```java
public boolean read(StringList row) throws IOException
```

Reads the next row as a string array.

**Parameters:**
- *row* - [out] A string list with each comma-separated element as a separate entry.

**Returns:**
Returns true if a row had been read or false otherwise.

**Throws:**
- `java.io.IOException` - If bad things happen during the read.

### setStartLine

```java
public void setStartLine(int startLine)
```

Sets the number of lines to be skipped from the beginning.
Parameters:
   startLine - [in] The line number to skip for start reading

setLocale
public void setLocale(String localeStr)

   Sets the locale that will be used to read the file.

   Parameters:
   localeStr - [in] The locale string for the file encoding.

getRow
public int getRow()
   throws IOException

   The row number for the current row.

   Returns:
   The current row number; 0 if there is no current row.

   Throws:
   java.io.IOException - If it fails.
com.sparsity.sparksee.io
Class CSVWriter

```java
java.lang.Object
    +-com.sparsity.sparksee.io.RowWriter
    |   \-com.sparsity.sparksee.io.CSVWriter

public class CSVWriter
extends RowWriter
```

CSVWriter interface.

A very simple CSV writer implementing RowWriter.

It works as any other RowWriter, but open must be called once before the first write operation.

It uses the format RFC 4180: http://tools.ietf.org/html/rfc4180

You can use your own separators and quote characters. By default the separator is the comma (,) and the quote character is the double quotes (")) and autoquote is enabled.

See the CSVReader locale documentation or the SPARKSEE User Manual.

Check out the 'Data export' section in the SPARKSEE User Manual for more details on this.

**Author:**
Sparsity Technologies http://www.sparsity-technologies.com

### Constructor Summary

<table>
<thead>
<tr>
<th>Public Constructor</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
<td>CSVWriter()</td>
<td>Creates a new instance.</td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Void Method</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>close()</td>
<td>Closes the writer.</td>
</tr>
<tr>
<td>void</td>
<td>open(String f)</td>
<td>Opens the output file path.</td>
</tr>
<tr>
<td>void</td>
<td>setAutoQuotes(boolean autoquotes)</td>
<td>Sets on/off the automatic quote mode.</td>
</tr>
<tr>
<td>void</td>
<td>setForcedQuotes(BooleanList forcequotes)</td>
<td>Disables the automatic quote mode and forces to be quoted those positions set to TRUE in the given vector.</td>
</tr>
<tr>
<td>void</td>
<td>setLocale(String localeStr)</td>
<td>Sets the locale that will be used to write the file.</td>
</tr>
<tr>
<td>void</td>
<td>setQuotes(String quotes)</td>
<td>Sets the character used to quote fields.</td>
</tr>
<tr>
<td>void</td>
<td>setSeparator(String sep)</td>
<td>Sets the character used to separate fields in the file.</td>
</tr>
</tbody>
</table>
void \texttt{write(StringList \texttt{row})}

Writes the next row.

Methods inherited from class \texttt{com.sparsity.sparksee.io.RowWriter}

\texttt{close, write}

Methods inherited from class \texttt{java.lang.Object}

\texttt{clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait}

Constructors

\textbf{CSVWriter}

\texttt{public CSVWriter()}

Creates a new instance.

Methods

\textbf{setAutoQuotes}

\texttt{public void setAutoQuotes(boolean autoquotes)}

Sets on/off the automatic quote mode.

If there are forced quotes, setting autoquotes on will clear them. If the autoquotes is set to off and no forced quotes are provided, there will not be any quote.

\textbf{Parameters:}

autoquotes - [in] If TRUE it enables the automatic quote mode, if FALSE it disables it.

\textbf{setSeparator}

\texttt{public void setSeparator(String sep)}

\texttt{throws RuntimeException}

Sets the character used to separate fields in the file.

\textbf{Parameters:}

sep - [in] Separator character.

\textbf{Throws:}

java.lang.RuntimeException - null

\textbf{setQuotes}

\texttt{public void setQuotes(String quotes)}

\texttt{throws RuntimeException}
Sets the character used to quote fields.

**Parameters:**
- quotes - [in] Quote character.

**Throws:**
- java.lang.RuntimeException - null

---

**setLocale**

*public void setLocale(String localeStr)*

Sets the locale that will be used to write the file.

**Parameters:**
- localeStr - [in] The locale string for the file encoding.

---

**write**

*public void write(StringList row)*

throws IOException, RuntimeException

Writes the next row.

**Parameters:**
- row - [in] Row of data.

**Throws:**
- java.io.IOException - If bad things happen during the write.
- java.lang.RuntimeException - null

---

**setForcedQuotes**

*public void setForcedQuotes(BooleanList forcequotes)*

Disables the automatic quote mode and forces to be quoted those positions set to TRUE in the given vector.

**Parameters:**
- forcequotes - [in] A booleanList with the position for each column that must be quoted set to true.

---

**close**

*public void close()*

throws IOException, RuntimeException

Closes the writer.
open

public void open(String f)
throws IOException

Opens the output file path.

Parameters:
   f - [in] Output file path.

Throws:
   java.io.IOException - If bad things happen opening the file.
**com.sparsity.sparksee.io**

**Class EdgeTypeExporter**

```java
public class EdgeTypeExporter extends TypeExporter
```

EdgeTypeExporter class.

Specific TypeExporter implementation for edge types.

Check out the 'Data export' section in the SPARKSEE User Manual for more details on this.

**Author:**
Sparsity Technologies http://www.sparsity-technologies.com

---

### Constructor Summary

<table>
<thead>
<tr>
<th>Public</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
<td><code>public EdgeTypeExporter(RowWriter rw, Graph g, int t, AttributeList attrs, int hPos, int tPos, int hAttr, int tAttr)</code></td>
</tr>
<tr>
<td></td>
<td>Creates a new instance.</td>
</tr>
<tr>
<td>public</td>
<td><code>public EdgeTypeExporter()</code></td>
</tr>
<tr>
<td></td>
<td>Creates a new instance.</td>
</tr>
</tbody>
</table>

---

### Method Summary

<table>
<thead>
<tr>
<th>void</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>register(TypeExporterListener tel)</code></td>
</tr>
<tr>
<td></td>
<td>Registers a new listener.</td>
</tr>
<tr>
<td></td>
<td><code>run()</code></td>
</tr>
<tr>
<td></td>
<td>See the TypeExporter class Run method.</td>
</tr>
<tr>
<td></td>
<td><code>setAttributes(AttributeList attrs)</code></td>
</tr>
<tr>
<td></td>
<td>Sets the list of Attributes.</td>
</tr>
<tr>
<td></td>
<td><code>setFrequency(int freq)</code></td>
</tr>
<tr>
<td></td>
<td>Sets the frequency of listener notification.</td>
</tr>
<tr>
<td></td>
<td><code>setGraph(Graph g)</code></td>
</tr>
<tr>
<td></td>
<td>Sets the graph that will be exported.</td>
</tr>
<tr>
<td></td>
<td><code>setHeadAttribute(int attr)</code></td>
</tr>
<tr>
<td></td>
<td>Sets the attribute that will be used to get the value to be dumped for the head of the edge.</td>
</tr>
<tr>
<td></td>
<td><code>setHeader(boolean header)</code></td>
</tr>
<tr>
<td></td>
<td>Sets the presence of a header row.</td>
</tr>
<tr>
<td></td>
<td><code>setHeadPosition(int pos)</code></td>
</tr>
<tr>
<td></td>
<td>Sets the position (index column) of the head attribute in the exported data.</td>
</tr>
</tbody>
</table>
```java
void setRowWriter(RowWriter rw)
    Sets the output data destination.

void setTailAttribute(int attr)
    Sets the attribute that will be used to get the value to be dumped for the tail of the edge.

void setTailPosition(int pos)
    Sets the position (index column) of the tail attribute in the exported data.

void setType(int t)
    Sets the type to be exported.
```

Methods inherited from class `com.sparsity.sparksee.io.TypeExporter`

`register`, `run`, `setAttributes`, `setFrequency`, `setGraph`, `setHeader`, `setRowWriter`, `setType`

Methods inherited from class `java.lang.Object`

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

## Constructors

### EdgeTypeExporter

```java
public EdgeTypeExporter(RowWriter rw,
    Graph g,
    int t,
    AttributeList attrs,
    int hPos,
    int tPos,
    int hAttr,
    int tAttr)
```

Creates a new instance.

**Parameters:**
- `g` - [in] Graph.
- `t` - [in] Type identifier.
- `attrs` - [in] Attribute identifiers to be exported.
- `hPos` - [in] The position (index column) for the head value.
- `tPos` - [in] The position (index column) for the tail value.
- `hAttr` - [in] The attribute identifier to get the value to be dumped for the head.
- `tAttr` - [in] The attribute identifier to get the value to be dumped for the tail.

### EdgeTypeExporter

```java
public EdgeTypeExporter() {
```

Creates a new instance.
Methods

setTailAttribute

public void setTailAttribute(int attr)

Sets the attribute that will be used to get the value to be dumped for the tail of the edge.

Parameters:
  attr - [in] Tail Attribute

setFrequency

public void setFrequency(int freq)

Sets the frequency of listener notification.

freq [in] Frequency in number of rows managed to notify progress to all listeners

Parameters:
  freq - null

setType

public void setType(int t)

Sets the type to be exported.

Parameters:
  t - [in] Type identifier.

setHeadAttribute

public void setHeadAttribute(int attr)

Sets the attribute that will be used to get the value to be dumped for the head of the edge.

Parameters:
  attr - [in] Head Attribute

setTailPosition

public void setTailPosition(int pos)

Sets the position (index column) of the tail attribute in the exported data.

Parameters:
  pos - [in] Tail position
**setGraph**

```java
public void setGraph(Graph g)
```

Sets the graph that will be exported.

**Parameters:**
- `g` - [in] Graph.

**setRowWriter**

```java
public void setRowWriter(RowWriter rw)
```

Sets the output data destination.

**Parameters:**
- `rw` - null

**register**

```java
public void register(TypeExporterListener tel)
```

Registers a new listener.

**Parameters:**
- `tel` - [in] TypeExporterListener to be registered.

**run**

```java
public void run()
    throws IOException,
             RuntimeException
```

See the TypeExporter class Run method.

**setHeader**

```java
public void setHeader(boolean header)
```

Sets the presence of a header row.

**Parameters:**
- `header` - [in] If TRUE, a header row is dumped with the name of the attributes.

**setHeadPosition**

```java
public void setHeadPosition(int pos)
```
Sets the position (index column) of the head attribute in the exported data.

**Parameters:**
- **pos** - [in] Head position

### setAttributes

```java
public void setAttributes(AttributeList attrs)
```

Sets the list of Attributes.

**Parameters:**
- **attrs** - [in] Attribute identifiers to be exported
com.sparsity.sparksee.io
Class EdgeTypeLoader

docs.com.sparsity.sparksee.io

java.lang.Object
   +com.sparsity.sparksee.io.TypeLoader
   +com.sparsity.sparksee.io.EdgeTypeLoader

public class EdgeTypeLoader
extends TypeLoader

EdgeTypeLoader class.

Specific TypeLoader implementation for edge types.

Check out the 'Data import' section in the SPARKSEE User Manual for more details on this.

Author:
Sparsity Technologies http://www.sparsity-technologies.com

### Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
<td>EdgeTypeLoader(RowReader rr, Graph g, int t, AttributeList attrs, Int32List attrsPos, int hPos, int tPos, int hAttr, int tAttr)</td>
<td>Creates a new instance.</td>
</tr>
<tr>
<td>public</td>
<td>EdgeTypeLoader()</td>
<td>Creates a new instance.</td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>register(TypeLoaderListener tel)</td>
<td>Registers a new listener.</td>
</tr>
<tr>
<td>void</td>
<td>run()</td>
<td>See the TypeLoader class Run method.</td>
</tr>
<tr>
<td>void</td>
<td>runNPhases(int partitions)</td>
<td>See the TypeLoader class RunNPhases method.</td>
</tr>
<tr>
<td>void</td>
<td>runTwoPhases()</td>
<td>See the TypeLoader class RunTwoPhases method.</td>
</tr>
<tr>
<td>void</td>
<td>setAttributePositions(Int32List attrsPos)</td>
<td>Sets the list of attribute positions.</td>
</tr>
<tr>
<td>void</td>
<td>setAttributes(AttributeList attrs)</td>
<td>Sets the list of Attributes.</td>
</tr>
<tr>
<td>void</td>
<td>setFrequency(int freq)</td>
<td>Sets the frequency of listener notification.</td>
</tr>
<tr>
<td>void</td>
<td>setGraph(Graph g)</td>
<td>Sets the graph where the data will be loaded.</td>
</tr>
</tbody>
</table>
### void setHeadAttribute(int attr)
Sets the attribute that will be used to find the head of the edge.

### void setHeadPosition(int pos)
Sets the position of the head attribute in the source data.

### void setLocale(String localeStr)
Sets the locale that will be used to read the data.

### void setLogError(String path)
Sets a log error file.

### void setLogOff()
Turns off all the error reporting.

### void setRowReader(RowReader rr)
Sets the input data source.

### void setTailAttribute(int attr)
Sets the attribute that will be used to find the tail of the edge.

### void setTailPosition(int pos)
Sets the position of the tail attribute in the source data.

### void setTimestampFormat(String timestampFormat)
Sets a specific timestamp format.

### void setType(int t)
Sets the type to be loaded.

**Methods inherited from class** com.sparsity.sparksee.io.TypeLoader

- register, run, runNPhases, runTwoPhases, setAttributePositions, setAttributes, setFrequency, setGraph, setLocale, setLogError, setLogOff, setRowReader, setTimestampFormat, setType

**Methods inherited from class** java.lang.Object

- clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### Constructors

**EdgeTypeLoader**

```java
public EdgeTypeLoader(RowReader rr, Graph g, int t, AttributeList attrs, Int32List attrsPos, int hPos, int tPos, int hAttr, int tAttr)
```

Creates a new instance.
Parameters:
  rr - [in] Input RowReader.
  g - [in] Graph.
  t - [in] Type identifier.
  attrs - [in] Attribute identifiers to be loaded.
  attrsPos - [in] Attribute positions (column index >=0) to all listeners.
  hPos - [in] The position (index column) for the head value.
  tPos - [in] The position (index column) for the tail value.
  hAttr - [in] The attribute identifier for the head.
  tAttr - [in] The attribute identifier for the tail.

**EdgeTypeLoader**

**public** EdgeTypeLoader()

Creates a new instance.

**Methods**

**setTailAttribute**

**public** **void** setTailAttribute(int attr)

Sets the attribute that will be used to find the tail of the edge.

This method is protected because only the Edge loaders should have it.

**Parameters:**
  attr - [in] Tail Attribute

**setFrequency**

**public** **void** setFrequency(int freq)

Sets the frequency of listener notification.

freq [in] Frequency in number of rows managed to notify progress to all listeners

**Parameters:**
  freq - null

**setLogOff**

**public** **void** setLogOff()

Truns off all the error reporting.

The log file will not be created and no exceptions for invalid data will be thrown. If you just want to turn off the logs, but abort at the first error what you should do is not call this method and not set a logError file.

**setType**

**public** **void** setType(int t)

Sets the type to be loaded.
Parameters:
  \( t \) - [in] Type identifier.

**setLogError**

```java
public void setLogError(String path)
   throws IOException
```

Sets a log error file.

By default errors are thrown as a exception and the load process ends. If a log file is set, errors are logged there and the load process does not stop.

Parameters:
  path - null

Throws:
  java.io.IOException - If bad things happen opening the file.

**setHeadAttribute**

```java
public void setHeadAttribute(int attr)
```

Sets the attribute that will be used to find the head of the edge.

This method is protected because only the Edge loaders should have it.

Parameters:
  attr - [in] Head Attribute

**runTwoPhases**

```java
public void runTwoPhases()
   throws IOException, RuntimeException
```

See the TypeLoader class RunTwoPhases method.

**setTailPosition**

```java
public void setTailPosition(int pos)
```

Sets the position of the tail attribute in the source data.

This method is protected because only the Edge loaders should have it.

Parameters:
  pos - [in] Tail position

**setGraph**

```java
public void setGraph(Graph g)
```

Sets the graph where the data will be loaded.
Parameters:
  \( g \) - [in] Graph.

setRowReader

```java
public void setRowReader(RowReader rr)
```

Sets the input data source.

Parameters:
  \( rr \) - [in] Input RowReader.

setAttributePositions

```java
public void setAttributePositions(Int32List attrsPos)
```

Sets the list of attribute positions.

- \( attrsPos \) [in] Attribute positions (column index \( \geq 0 \)).

Parameters:
  \( attrsPos \) - null

register

```java
public void register(TypeLoaderListener tel)
```

Registers a new listener.

Parameters:
  \( tel \) - TypeLoaderListener to be registered.

setLocale

```java
public void setLocale(String localeStr)
```

Sets the locale that will be used to read the data.

- It should match the locale used in the rowreader.

Parameters:
  \( localeStr \) - [in] The locale string for the read data. See CSVReader.

run

```java
public void run()
    throws IOException, RuntimeException
```

See the TypeLoader class Run method.
runNPhases

public void runNPhases(int partitions) throws IOException, RuntimeException

See the TypeLoader class RunNPhases method.

Parameters:
  partitions - null

Throws:
  java.io.IOException - null
  java.lang.RuntimeException - null

setTimestampFormat

public void setTimestampFormat(String timestampFormat)

Sets a specific timestamp format.

Parameters:
  timestampFormat - null

setHeadPosition

public void setHeadPosition(int pos)

Sets the position of the head attribute in the source data.

This method is protected because only the Edge loaders should have it.

Parameters:
  pos - [in] Head position

setAttributes

public void setAttributes(AttributeList attrs)

Sets the list of Attributes.

Parameters:
  attrs - [in] Attribute identifiers to be loaded
public class NodeExporter
extends TypeExporter

NodeTypeExporter class.
Specific TypeExporter implementation for node types.

Check out the 'Data export' section in the SPARKSEE User Manual for more details on this.

Author:
Sparsity Technologies http://www.sparsity-technologies.com

Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public NodeTypeExporter(RowWriter rw, Graph g, int t, AttributeList attrs)</td>
<td>Creates a new instance.</td>
</tr>
<tr>
<td>public NodeTypeExporter()</td>
<td>Creates a new instance.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void register(TypeExporterListener tel)</td>
<td>Registers a new listener.</td>
</tr>
<tr>
<td>void run()</td>
<td>See the TypeExporter class Run method.</td>
</tr>
<tr>
<td>void setAttributes(AttributeList attrs)</td>
<td>Sets the list of Attributes.</td>
</tr>
<tr>
<td>void setFrequency(int freq)</td>
<td>Sets the frequency of listener notification.</td>
</tr>
<tr>
<td>void setGraph(Graph g)</td>
<td>Sets the graph that will be exported.</td>
</tr>
<tr>
<td>void setHeader(boolean header)</td>
<td>Sets the presence of a header row.</td>
</tr>
<tr>
<td>void setRowWriter(RowWriter rw)</td>
<td>Sets the output data destination.</td>
</tr>
<tr>
<td>void setType(int t)</td>
<td>Sets the type to be exported.</td>
</tr>
</tbody>
</table>

Methods inherited from class com.sparsity.sparksee.io.TypeExporter
Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

NodeTypeExporter

public NodeTypeExporter(RowWriter rw,
Graph g,
int t,
AttributeList attrs)

Creates a new instance.

Parameters:
   g - [in] Graph.
   t - [in] Type identifier.
   attrs - [in] Attribute identifiers to be exported.

NodeTypeExporter

public NodeTypeExporter()

Creates a new instance.

Methods

setRowWriter

public void setRowWriter(RowWriter rw)

Sets the output data destination.

rw - [in] Input RowWriter.

Parameters:
   rw - null

setFrequency

public void setFrequency(int freq)

Sets the frequency of listener notification.

freq [in] Frequency in number of rows managed to notify progress to all listeners

Parameters:
public void register(TypeExporterListener tel)

Registers a new listener.

Parameters:
  tel - [in] TypeExporterListener to be registered.

public void run()
  throws IOException, RuntimeException

See the TypeExporter class Run method.

public void setType(int t)

Sets the type to be exported.

Parameters:
  t - [in] Type identifier.

public void setHeader(boolean header)

Sets the presence of a header row.

Parameters:
  header - [in] If TRUE, a header row is dumped with the name of the attributes.

public void setGraph(Graph g)

Sets the graph that will be exported.

Parameters:
  g - [in] Graph.

public void setAttributes(AttributeList attrs)
Sets the list of Attributes.

**Parameters:**

```plaintext
attrs - [in] Attribute identifiers to be exported
```
**Class NodeTypeLoader**

java.lang.Object

- com.sparsity.sparksee.io.TypeLoader

- com.sparsity.sparksee.io.NodeTypeLoader

public class **NodeTypeLoader**

extends **TypeLoader**

NodeTypeLoader class.

Specific TypeLoader implementation for node types.

Check out the 'Data import' section in the SPARKSEE User Manual for more details on this.

**Author:**
Sparsity Technologies http://www.sparsity-technologies.com

### Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public NodeTypeLoader()</td>
<td>Creates a new instance.</td>
</tr>
<tr>
<td>public NodeTypeLoader(RowReader rr, Graph g, int t, AttributeList attrs, Int32List attrsPos)</td>
<td>Creates a new instance.</td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void register(TypeLoaderListener tel)</td>
<td>Registers a new listener.</td>
</tr>
<tr>
<td>void run()</td>
<td>See the TypeLoader class Run method.</td>
</tr>
<tr>
<td>void runNPhases(int partitions)</td>
<td>See the TypeLoader class RunNPhases method.</td>
</tr>
<tr>
<td>void runTwoPhases()</td>
<td>See the TypeLoader class RunTwoPhases method.</td>
</tr>
<tr>
<td>void setAttributePositions(Int32List attrsPos)</td>
<td>Sets the list of attribute positions.</td>
</tr>
<tr>
<td>void setAttributes(AttributeList attrs)</td>
<td>Sets the list of Attributes.</td>
</tr>
<tr>
<td>void setFrequency(int freq)</td>
<td>Sets the frequency of listener notification.</td>
</tr>
<tr>
<td>void setGraph(Graph g)</td>
<td>Sets the graph where the data will be loaded.</td>
</tr>
</tbody>
</table>
void `setLocale` (String localeStr)
Sets the locale that will be used to read the data.

void `setLogError` (String path)
Sets a log error file.

void `setLogOff`
Turns off all the error reporting.

void `setRowReader` (RowReader `rr`)
Sets the input data source.

void `setTimestampFormat` (String `timestampFormat`)
Sets a specific timestamp format.

void `setType` (int `t`)
Sets the type to be loaded.

Methods inherited from class `com.sparsity.sparksee.io.TypeLoader`

`register`, `run`, `runNPhases`, `runTwoPhases`, `setAttributePositions`, `setAttributes`,
`setFrequency`, `setGraph`, `setLocale`, `setLogError`, `setLogOff`, `setRowReader`,
`setTimestampFormat`, `setType`

Methods inherited from class `java.lang.Object`

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Constructors

`NodeTypeLoader`

public `NodeTypeLoader` ()
Creates a new instance.

`NodeTypeLoader`

public `NodeTypeLoader` (RowReader `rr`,
Graph `g`,
int `t`,
AttributeList `attrs`,
Int32List `attrsPos`)

Creates a new instance.

Parameters:
`rr` - [in] Input RowReader.
`g` - [in] Graph.
`t` - [in] Type identifier.
`attrs` - [in] Attribute identifiers to be loaded.
`attrsPos` - [in] Attribute positions (column index >=0).
Methods

**setFrequency**

```java
public void setFrequency(int freq)
```

Sets the frequency of listener notification.

freq [in] Frequency in number of rows managed to notify progress to all listeners

**Parameters:**

freq - null

**setLogOff**

```java
public void setLogOff()
```

Turns off all the error reporting.

The log file will not be created and no exceptions for invalid data will be thrown. If you just want to turn off the logs, but abort at the first error what you should do is not call this method and not set a logError file.

**setLogError**

```java
public void setLogError(String path)
throws IOException
```

Sets a log error file.

By default errors are thrown as an exception and the load process ends. If a log file is set, errors are logged there and the load process does not stop.

**Parameters:**

path - null

**Throws:**

java.io.IOException - If bad things happen opening the file.

**setType**

```java
public void setType(int t)
```

Sets the type to be loaded.

**Parameters:**

`t` - [in] Type identifier.

**runTwoPhases**

```java
public void runTwoPhases()
throws IOException,
RuntimeException
```

See the TypeLoader class RunTwoPhases method.
setGraph

public void setGraph(Graph g)

Sets the graph where the data will be loaded.

Parameters:

    g - [in] Graph.

setRowReader

public void setRowReader(RowReader rr)

Sets the input data source.

Parameters:

    rr - [in] Input RowReader.

setAttributePositions

public void setAttributePositions(Int32List attrsPos)

Sets the list of attribute positions.

attrsPos [in] Attribute positions (column index >=0).

Parameters:

    attrsPos - null

register

public void register(TypeLoaderListener tel)

Registers a new listener.

Parameters:

    tel - TypeLoaderListener to be registered.

setLocale

public void setLocale(String localeStr)

Sets the locale that will be used to read the data.

It should match the locale used in the rowreader.

Parameters:

    localeStr - [in] The locale string for the read data. See CSVReader.
run

public void run()
    throws IOException,
            RuntimeException

    See the TypeLoader class Run method.

runNPhases

public void runNPhases(int partitions)
    throws IOException,
            RuntimeException

    See the TypeLoader class RunNPhases method.

Parameters:

    partitions - null

Throws:

    java.io.IOException - null
    java.lang.RuntimeException - null

setTimestampFormat

public void setTimestampFormat(String timestampFormat)

    Sets a specific timestamp format.

Parameters:

    timestampFormat - null

setAttributes

public void setAttributes(AttributeList attrs)

    Sets the list of Attributes.

Parameters:

    attrs - [in] Attribute identifiers to be loaded
com.sparsity.sparksee.io
Class RowReader

public class RowReader
extends Object

RowReader interface.

Common interface for those readers which get the data as an string array.

It works as follows: perform as many read operations as necessary and call close once at the end. Once close is called no more read operations can be executed.

Check out the 'Data import' section in the SPARKSEE User Manual for more details on this.

Author:
Sparsity Technologies http://www.sparsity-technologies.com

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>close</td>
<td>void</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Closes the reader.</td>
</tr>
<tr>
<td>close</td>
<td>void</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>The row number for the current row.</td>
</tr>
<tr>
<td>read</td>
<td>boolean</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reads the next row as a string array.</td>
</tr>
<tr>
<td>reset</td>
<td>boolean</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Moves the reader to the beginning.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods

close

public void close()
throws IOException

Closes the reader.
getRow

public int getRow() throws IOException

The row number for the current row.

Returns:
The current row number; 0 if there is no current row.

Throws:
java.io.IOException - If it fails.

reset

public boolean reset() throws IOException

Moves the reader to the beginning.
Restarts the reader.

Returns:
true if the reader can be restarted, false otherwise.

Throws:
java.io.IOException - If bad things happen during the restart.

read

public boolean read(StringList row) throws IOException

Reads the next row as a string array.

Parameters:
row - [out] A string list with each comma-separated element as a separate entry.

Returns:
Returns true if a row had been read or false otherwise.

Throws:
java.io.IOException - If bad things happen during the read.
com.sparsity.sparksee.io
Class RowWriter

public class RowWriter extends Object

RowWriter interface.

Common interface for those writers which dump the data from an string array.

It works as follows: perform as many write operations as necessary and call close once at the end. Once close is called no more write operations can be executed.

Check out the 'Data export' section in the SPARKSEE User Manual for more details on this.

Author:
Sparsity Technologies http://www.sparsity-technologies.com

---

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>close()</td>
<td>Closes the writer.</td>
</tr>
<tr>
<td>write(row)</td>
<td>Writes the next row.</td>
</tr>
</tbody>
</table>

### Methods inherited from class java.lang.Object

- clone
- equals
- finalize
- getClass
- hashCode
- notify
- notifyAll
- toString
- wait
- wait
- wait

---

### Methods

**write**

```java
public void write(StringList row)
```

**Throws:**

- IOException
- RuntimeException

Writes the next row.

**Parameters:**

- `row` - [in] Row of data.

**Throws:**

- java.io.IOException - If bad things happen during the write.
public void close() throws IOException, RuntimeException

            Closes the writer.
com.sparsity.sparksee.io
Class TypeExporter

java.lang.Object
    +-com.sparsity.sparksee.io.TypeExporter

Direct Known Subclasses:
    NodeTypeExporter, EdgeTypeExporter

public class TypeExporter
    extends Object

Base TypeExporter class.

Base class to export a node or edge type from a graph using a RowWriter.

TypeExporterListener can be registered to receive information about the progress of the export process by means of
TypeExporterEvent. The default frequency of notification to listeners is 100000.

By default no header row is created.

Check out the 'Data export' section in the SPARKSEE User Manual for more details on this.

Author:
    Sparsity Technologies http://www.sparsity-technologies.com

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>register(TypeExporterListener tel)</td>
<td>Registers a new listener.</td>
</tr>
<tr>
<td>run()</td>
<td>Runs export process.</td>
</tr>
<tr>
<td>setAttributes(AttributeList attrs)</td>
<td>Sets the list of Attributes.</td>
</tr>
<tr>
<td>setFrequency(int freq)</td>
<td>Sets the frequency of listener notification.</td>
</tr>
<tr>
<td>setGraph(Graph g)</td>
<td>Sets the graph that will be exported.</td>
</tr>
<tr>
<td>setHeader(boolean header)</td>
<td>Sets the presence of a header row.</td>
</tr>
<tr>
<td>setRowWriter(RowWriter rw)</td>
<td>Sets the output data destination.</td>
</tr>
<tr>
<td>setType(int t)</td>
<td>Sets the type to be exported.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait,
**Methods**

**setRowWriter**

```java
public void setRowWriter(RowWriter rw)
```

Sets the output data destination.

*Parameters:*
- `rw` - null

**setFrequency**

```java
public void setFrequency(int freq)
```

Sets the frequency of listener notification.

*Parameters:*
- `freq` - null

**run**

```java
public void run()
    throws IOException,
            RuntimeException
```

Runs export process.

**register**

```java
public void register(TypeExporterListener tel)
```

Registers a new listener.

*Parameters:*
- `tel` - [in] TypeExporterListener to be registered.

**setType**

```java
public void setType(int t)
```

Sets the type to be exported.

*Parameters:*
- `t` - [in] Type identifier.
setHeader

public void setHeader(boolean header)

Sets the presence of a header row.

Parameters:
header - [in] If TRUE, a header row is dumped with the name of the attributes.

setGraph

public void setGraph(Graph g)

Sets the graph that will be exported.

Parameters:
g - [in] Graph.

setAttributes

public void setAttributes(AttributeList attrs)

Sets the list of Attributes.

Parameters:
attrs - [in] Attribute identifiers to be exported
com.sparsity.sparksee.io
Class TypeExporterEvent

public class TypeExporterEvent
extends Object

Provides information about the progress of a TypeExporter instance.

Check out the 'Data export' section in the SPARKSEE User Manual for more details on this.

Author: Sparsity Technologies http://www.sparsity-technologies.com

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>long</td>
<td>getCount()</td>
</tr>
<tr>
<td></td>
<td>Gets the current number of objects exported.</td>
</tr>
<tr>
<td>long</td>
<td>getTotal()</td>
</tr>
<tr>
<td></td>
<td>Gets the total number of objects exported.</td>
</tr>
<tr>
<td>int</td>
<td>getTypeId()</td>
</tr>
<tr>
<td></td>
<td>Gets the type identifier.</td>
</tr>
<tr>
<td>boolean</td>
<td>isLast()</td>
</tr>
<tr>
<td></td>
<td>Gets if this is the last event or not.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

Methods

getCount

public long getCount()

    Gets the current number of objects exported.

Returns:
    The current number of objects exported.

isLast

public boolean isLast()
Gets if this is the last event or not.

**Returns:**
TRUE if this is the last event, FALSE otherwise.

---

**getTypeId**

```java
public int getTypeId()
```

Gets the type identifier.

**Returns:**
The type identifier.

---

**getTotal**

```java
public long getTotal()
```

Gets the total number of objects exported.

**Returns:**
The total number of objects exported.
public class TypeExporterListener
extends Object

Interface to be implemented to receive TypeExporterEvent events from a TypeExporter.

Check out the 'Data export' section in the SPARKSEE User Manual for more details on this.

Author: Sparsity Technologies http://www.sparsity-technologies.com

Method Summary

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>notifyEvent</td>
<td>Method to be notified from a TypeExporter.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods

notifyEvent

public void notifyEvent (TypeExporterEvent tee)

Method to be notified from a TypeExporter.

Parameters:
tee - [in] Notified event.
com.sparsity.sparksee.io
Class TypeLoader

java.lang.Object
   +-com.sparsity.sparksee.io.TypeLoader

Direct Known Subclasses:
   NodeTypeLoader, EdgeTypeLoader

public class TypeLoader
extends Object

Base TypeLoader class.

Base class to load a node or edge type from a graph using a RowReader.

TypeLoaderListener can be registered to receive information about the progress of the load process by means of TypeLoaderEvent. The default frequency of notification to listeners is 100000.

Check out the 'Data import' section in the SPARKSEE User Manual for more details on this.

Author:
Sparsity Technologies http://www.sparsity-technologies.com

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
</table>
| void register(TypeLoaderListener tel)  
  Registers a new listener. |
| void run()  
  Run the loader. |
| void runNPhases(int partitions)  
  Run the loader for N phases loading. |
| void runTwoPhases()  
  Run the loader for two phases loading. |
| void setAttributePositions(Int32List attrsPos)  
  Sets the list of attribute positions. |
| void setAttributes(AttributeList attrs)  
  Sets the list of Attributes. |
| void setFrequency(int freq)  
  Sets the frequency of listener notification. |
| void setGraph(Graph g)  
  Sets the graph where the data will be loaded. |
| void setLocale(String localeStr)  
  Sets the locale that will be used to read the data. |
| void setLogError(String path)  
  Sets a log error file. |
### Methods

#### runTwoPhases

```java
generic void runTwoPhases()
    throws IOException, RuntimeException
```

Run the loader for two phases loading.

Firstly load all objects (and create them if necessary) and secondly loads all the attributes.

Working on this mode it is necessary to build a temporary file.

#### runNPhases

```java
generic void runNPhases(int partitions)
    throws IOException, RuntimeException
```

Run the loader for N phases loading.

Firstly load all objects (and create them if necessary) and secondly loads all the attributes. But in this case, attributes are loaded one by one. This way, if there are three attributes, then 4 traverses are necessary.

Working on this mode it is necessary to build a temporary file.

**Parameters:**

- `partitions` - [in] Number of horizontal partitions to perform the load.

**Throws:**

- `java.io.IOException` - null
- `java.lang.RuntimeException` - null

#### setFrequency

```java
public void setFrequency(int freq)
```

Sets the frequency of listener notification.

freq [in] Frequency in number of rows managed to notify progress to all listeners
Parameters:

- `freq` - null

**setLogOff**

```java
public void setLogOff()
```

Truns off all the error reporting.

The log file will not be created and no exceptions for invalid data will be thrown. If you just want to turn off the logs, but abort at the first error what you should do is not call this method and not set a logError file.

**run**

```java
public void run()
```

```java
throws IOException, RuntimeException
```

Run the loader.

**setLogError**

```java
public void setLogError(String path)
```

```java
throws IOException
```

Sets a log error file.

By default errors are thrown as an exception and the load process ends. If a log file is set, errors are logged there and the load process does not stop.

**Parameters:**

- `path` - null

**Throws:**

- `java.io.IOException` - If bad things happen opening the file.

**setType**

```java
public void setType(int t)
```

Sets the type to be loaded.

**Parameters:**

- `t` - [in] Type identifier.

**setGraph**

```java
public void setGraph(Graph g)
```

Sets the graph where the data will be loaded.

**Parameters:**

- `g` - [in] Graph.
setRowReader

public void setRowReader(RowReader rr)

Sets the input data source.

Parameters:

  rr - [in] Input RowReader.

register

public void register(TypeLoaderListener tel)

Registers a new listener.

Parameters:

tel - TypeLoaderListener to be registered.

setAttributePositions

public void setAttributePositions(Int32List attrsPos)

Sets the list of attribute positions.

  attrsPos [in] Attribute positions (column index >=0).

Parameters:

  attrsPos - null

setLocale

public void setLocale(String localeStr)

Sets the locale that will be used to read the data.

  It should match the locale used in the rowreader.

Parameters:

  localeStr - [in] The locale string for the read data. See CSVReader.

setTimestampFormat

public void setTimestampFormat(String timestampFormat)

Sets a specific timestamp format.

Parameters:

  timestampFormat - null

setAttributes

public void setAttributes(AttributeList attrs)
Sets the list of Attributes.

**Parameters:**

- **attrs** - [in] Attribute identifiers to be loaded
public class TypeLoaderEvent
extends Object

Provides information about the progress of a TypeLoader instance.

Check out the 'Data import' section in the SPARKSEE User Manual for more details on this.

Author: Sparsity Technologies http://www.sparsity-technologies.com

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getCount()</td>
<td>Gets the current number of objects created.</td>
</tr>
<tr>
<td>getPartition()</td>
<td>Gets the current partition.</td>
</tr>
<tr>
<td>getPhase()</td>
<td>Gets the current phase.</td>
</tr>
<tr>
<td>getTotalPartitions()</td>
<td>Gets the total number of partitions.</td>
</tr>
<tr>
<td>getTotalPartitionSteps()</td>
<td>Gets the total number of steps in the current partition.</td>
</tr>
<tr>
<td>getTotalPhases()</td>
<td>Gets the total number of phases.</td>
</tr>
<tr>
<td>getTypeId()</td>
<td>Gets the type identifier.</td>
</tr>
<tr>
<td>isLast()</td>
<td>Gets if this is the last event or not.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

Methods

getTotalPhases

public int getTotalPhases()
Gets the total number of phases.

**Returns:**
The total number of phases.

---

### getCount

```java
public long getCount()
```

Gets the current number of objects created.

**Returns:**
The current number of objects created.

---

### getTotalPartitionSteps

```java
public int getTotalPartitionSteps()
```

Gets the total number of steps in the current partition.

**Returns:**
The total number steps in the current partition.

---

### isLast

```java
public boolean isLast()
```

Gets if this is the last event or not.

**Returns:**
TRUE if this is the last event, FALSE otherwise.

---

### getPartition

```java
public int getPartition()
```

Gets the current partition.

**Returns:**
The current partition.

---

### getTypeId

```java
public int getTypeId()
```

Gets the type identifier.

**Returns:**
The type identifier.

### getTotalPartitions

```java
public int getTotalPartitions()
```

Gets the total number of partitions.

**Returns:**

The total number of partitions.

### getPhase

```java
public int getPhase()
```

Gets the current phase.

**Returns:**

The current phase.
public class TypeLoaderListener
extends Object

Interface to be implemented to receive TypeLoaderEvent events from a TypeLoader.

Check out the 'Data import' section in the SPARKSEE User Manual for more details on this.

Author:
Sparsity Technologies http://www.sparsity-technologies.com

Method Summary

<table>
<thead>
<tr>
<th>void</th>
<th>notifyEvent(TypeLoaderEvent ev)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Method to receive events from a Loader.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods

**notifyEvent**

public void notifyEvent(TypeLoaderEvent ev)

Method to receive events from a Loader.

Parameters:

- ev - Loader.LoaderEvent with information from a running Loader.
Package

com.sparsity.sparksee.script
**Class ScriptParser**

The ScriptParser can create schemas and load data from a set of commands in a sparksee script.

A SPARKSEE script contains an ordered list of commands. ScriptParser will execute each one of them in order. Commands may create schemas, define nodes and edges, and load data into a previous defined SPARKSEE schema.

Check out the 'Scripting' chapter in the SPARKSEE User Manual for a comprehensive explanation on the grammar of the SPARKSEE commands and how they work.

**Author:** Sparsity Technologies http://www.sparsity-technologies.com

### Constructor Summary

<table>
<thead>
<tr>
<th>public</th>
<th>ScriptParser()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constructor.</td>
<td></td>
</tr>
</tbody>
</table>

### Method Summary

- **static void** `generateSchemaScript(String path, Database db)`
  - Writes an script with the schema definition for the given database.

- **static void** `main()`
  - Executes ScriptParser for the given file path.

- **boolean** `parse(String path, boolean execute, String localeStr)`
  - Parses the given input file.

- **void** `setErrorLog(String path)`
  - Sets the error log.

- **void** `setOutputLog(String path)`
  - Sets the output log.

### Methods inherited from class java.lang.Object

- clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait

### Constructors
**ScriptParser**

public ScriptParser()

Constructor.

---

## Methods

### parse

public boolean parse(String path,  
boolean execute,  
String localeStr)

throws IOException

Parses the given input file.

**Parameters:**
- path - [in] Input file path.
- execute - [in] If TRUE the script is executed, if FALSE it is just parsed.
- localeStr - [in] The locale string for reading the input file. See CSVReader.

**Returns:**
 TRUE if ok, FALSE in case of error.

**Throws:**
 java.io.IOException - If bad things happen opening the file.

---

### setErrorLog

public void setErrorLog(String path)

throws IOException

Sets the error log.

If not set, error log corresponds to standard error output.

**Parameters:**
- path - [in] Path of the error log.

**Throws:**
 java.io.IOException - If bad things happen opening the file.

---

### generateSchemaScript

public static void generateSchemaScript(String path,  
Database db)

throws IOException

Writes an script with the schema definition for the given database.

**Parameters:**
- path - [in] Filename of the script to be written.
db - [in] Database.

Throws:
java.io.IOException - If bad things happen opening or writing the file.

**setOutputLog**

```java
public void setOutputLog(String path)
throws IOException
```

Sets the output log.

If not set, output log corresponds to standard output.

Parameters:
- path - [in] Path of the output log.

Throws:
java.io.IOException - If bad things happen opening the file.

**main**

```java
public static void main()
```

Executes ScriptParser for the given file path.

One argument is required, a file path which contains the script to be parsed.

A second argument may be given, a boolean to set if the script must be executed or just parsed. If not given, the script will be executed.
Index

A

add 67, 75, 133, 153, 156, 165, 199, 210
addAll 157
addAllEdgeTypes 11, 15, 19, 23, 29, 33, 37, 40, 45, 49, 52, 57
addAllNodeTypes 7, 12, 14, 19, 24, 30, 32, 37, 41, 44, 48, 52, 57
addEdgeType 10, 14, 18, 23, 28, 32, 36, 40, 44, 48, 52, 56
addNodeType 7, 11, 14, 19, 24, 29, 33, 37, 41, 45, 49, 53, 57
addWeightedEdgeType 29
Any 103
any 151
Ascendent 169
asDirected 101
AttributeList 66

B

backup 120
Basic 63
begin 176
Between 81
Boolean 89
BooleanList 75
Box 146

C

checkOnlyExistence 24
clear 66, 75, 133, 154, 166, 198, 210
close 5, 8, 12, 16, 41, 84, 154, 161, 176, 180, 204, 227, 229, 233, 238, 260, 263
combineDifference 157
combineIntersection 154
combineUnion 152
commit 176
cmpare 218
cmpareTo 220, 222
cmpute 11
Config 138
ConnectedComponents 4
contains 151, 154
containsAll 152

Context 10
copy 153, 158
count 66, 76, 134, 154, 166, 198, 211, 227
countEdges 117
countNodes 121
create 179
CSVReader 232
CSVWriter 237

D

Debug 139
DefaultExport 92
degree 115
Descendent 170
difference 152
Double 89
drop 124, 125
dumpData 113
dumpStorage 115

E

Edge 163
EdgeExport 98
dges 119
EdgeTypeExporter 241
EdgeTypeLoader 246, 247
enableType 92, 106
Equal 80
equals 151, 153, 217, 223
excludeEdges 8, 12, 15, 20, 25, 30, 33, 38, 42, 46, 50, 53, 58
excludeNodes 7, 10, 14, 18, 23, 28, 32, 37, 40, 45, 49, 53, 57
exists 18, 22, 28, 157
explode 122, 125
export 116

F

findAttribute 119
findAttributes 121
findEdge 125
findEdgeTypes 117
findNodeTypes 120
findObject 127
findOrCreateEdge 123
findOrCreateObject 128
findType 128
findTypes 113
Fine 138

G

generateSchemaScript 281
get 194
getAlias 83
getAreNeighborsIndexed 206
getAttribute 120, 126, 128
getAttributeIntervalCount 114
getAttributes 116
getAttributeStatistics 116
getAttributeText 117
getAvailableMem 173
getAvgLengthString 73
getBoolean 195, 220
getCache 87
getCacheMaxSize 190
getCacheStatisticsEnabled 186
getCacheStatisticsFile 190
getCacheStatisticsSnapshotTime 190
g getColor 98, 142
g getColorRGB 99, 143
g getConnectedComponent 4
g getConnectedComponents 8, 33, 38, 53, 58
g getCost 20, 24, 30
g getCount 4, 61, 267, 276
g getCurrentDepth 41, 45, 49
g getData 87
g getDataType 62, 221
g getDistinct 72
g getDouble 223
g getEdge 92, 96, 105
g getEdgeData 122
g getEdgePeer 127
g getEdgeType 93, 105
g getExtentPages 187
g getExtentSize 185
g getFontSize 99, 142
g getGraph 92, 105, 176
g getHead 95
g getHeight 145
g getHighAvailabilityCoordinators 188
g getHighAvailabilityEnabled 188
g getHighAvailabilityIP 192
g getHighAvailabilityMasterHistory 192
g getHighAvailabilitySynchronization 184
g getID 62, 207
g getInteger 195, 225
g getIsDirected 207
g getIsRestricted 207
g getKind 61
g getLabel 101, 131, 145
g getLabelColor 100, 144
g getLabelColorRGB 100, 144
g getLicense 191
g getLogFile 186
g getLogLevel 192
g getLong 224
g getMax 72
g getMaxLengthString 73
g getMean 72
g getMedian 72
g getMin 71
g getMinLengthString 71
g getMode 71
g getModeCount 73
g getName 62, 207
g getNode 93, 106
g getNodes 5
g getNodeType 93, 104
g getNull 72
g getNumCPUs 174
g getNumObjects 207
g getObjectType 126, 206
g getOID 217
g getPartition 276
g getPath 84
g getPathAsEdges 19, 23, 28
g getPathAsNodes 20, 23, 28
g getPhase 277
g getPoolFrameSize 190
g getPoolPersistentMaxSize 187
g getPoolPersistentMinSize 193
g getPoolTemporaryMaxSize 191
g getPoolTemporaryMinSize 191
<table>
<thead>
<tr>
<th>Function/Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>getRead</td>
<td>87</td>
</tr>
<tr>
<td>getRealTime</td>
<td>173</td>
</tr>
<tr>
<td>getRecoveryCacheMaxSize</td>
<td>192</td>
</tr>
<tr>
<td>getRecoveryCheckpointTime</td>
<td>186</td>
</tr>
<tr>
<td>getRecoveryEnabled</td>
<td>188</td>
</tr>
<tr>
<td>getRecoveryLogFile</td>
<td>187</td>
</tr>
<tr>
<td>getRestrictedFrom</td>
<td>206</td>
</tr>
<tr>
<td>getRestrictedTo</td>
<td>207</td>
</tr>
<tr>
<td>getRow</td>
<td>235, 261</td>
</tr>
<tr>
<td>getSize</td>
<td>4, 61</td>
</tr>
<tr>
<td>getStatistics</td>
<td>84, 171</td>
</tr>
<tr>
<td>getString</td>
<td>221</td>
</tr>
<tr>
<td>getSystemTime</td>
<td>173</td>
</tr>
<tr>
<td>getTail</td>
<td>95</td>
</tr>
<tr>
<td>getTemp</td>
<td>87</td>
</tr>
<tr>
<td>getTimestamp</td>
<td>225</td>
</tr>
<tr>
<td>getTimestampAsCalendar</td>
<td>217</td>
</tr>
<tr>
<td>getTimestampAsDate</td>
<td>221</td>
</tr>
<tr>
<td>getTimeUnit</td>
<td>195</td>
</tr>
<tr>
<td>getTotal</td>
<td>73, 268</td>
</tr>
<tr>
<td>getTotalMem</td>
<td>173</td>
</tr>
<tr>
<td>getTotalPartitions</td>
<td>277</td>
</tr>
<tr>
<td>getTotalPartitionSteps</td>
<td>276</td>
</tr>
<tr>
<td>getTotalPhases</td>
<td>275</td>
</tr>
<tr>
<td>getType</td>
<td>118</td>
</tr>
<tr>
<td>getTypeId</td>
<td>61, 268, 276</td>
</tr>
<tr>
<td>getUserTime</td>
<td>173</td>
</tr>
<tr>
<td>getValues</td>
<td>127</td>
</tr>
<tr>
<td>getWidth</td>
<td>100, 144</td>
</tr>
<tr>
<td>getWrite</td>
<td>86</td>
</tr>
<tr>
<td>GlobalType</td>
<td>206</td>
</tr>
<tr>
<td>GraphExport</td>
<td>130</td>
</tr>
<tr>
<td>GraphML</td>
<td>108</td>
</tr>
<tr>
<td>Graphviz</td>
<td>107</td>
</tr>
<tr>
<td>GreaterEqual</td>
<td>80</td>
</tr>
<tr>
<td>GreaterThan</td>
<td>80</td>
</tr>
<tr>
<td>indexAttribute</td>
<td>118</td>
</tr>
<tr>
<td>Indexed</td>
<td>64</td>
</tr>
<tr>
<td>Info</td>
<td>138</td>
</tr>
<tr>
<td>Ingoing</td>
<td>102</td>
</tr>
<tr>
<td>Int32List</td>
<td>133</td>
</tr>
<tr>
<td>Integer</td>
<td>89</td>
</tr>
<tr>
<td>intersection</td>
<td>158</td>
</tr>
<tr>
<td>InvalidAttribute</td>
<td>60</td>
</tr>
<tr>
<td>InvalidOID</td>
<td>150</td>
</tr>
<tr>
<td>InvalidType</td>
<td>206</td>
</tr>
<tr>
<td>isClosed</td>
<td>5, 8, 12, 15, 42, 84, 156, 161, 176, 180, 227, 229</td>
</tr>
<tr>
<td>isEmpty</td>
<td>154</td>
</tr>
<tr>
<td>isFit</td>
<td>143</td>
</tr>
<tr>
<td>isLast</td>
<td>267, 276</td>
</tr>
<tr>
<td>isNull</td>
<td>203, 222</td>
</tr>
<tr>
<td>isSessionAttribute</td>
<td>61</td>
</tr>
<tr>
<td>iterator</td>
<td>66, 75, 133, 158, 166, 198, 210, 226, 227</td>
</tr>
<tr>
<td>iteratorFromElement</td>
<td>152</td>
</tr>
<tr>
<td>iteratorFromIndex</td>
<td>155</td>
</tr>
<tr>
<td>LessEqual</td>
<td>80</td>
</tr>
<tr>
<td>LessThan</td>
<td>80</td>
</tr>
<tr>
<td>Like</td>
<td>81</td>
</tr>
<tr>
<td>LikeNoCase</td>
<td>81</td>
</tr>
<tr>
<td>load</td>
<td>195</td>
</tr>
<tr>
<td>Long</td>
<td>89</td>
</tr>
<tr>
<td>main</td>
<td>282</td>
</tr>
<tr>
<td>MaxLengthString</td>
<td>216</td>
</tr>
<tr>
<td>neighbors</td>
<td>116, 122</td>
</tr>
<tr>
<td>newAttribute</td>
<td>119, 125</td>
</tr>
<tr>
<td>newEdge</td>
<td>126, 127</td>
</tr>
<tr>
<td>newEdgeType</td>
<td>124</td>
</tr>
<tr>
<td>newNode</td>
<td>117</td>
</tr>
<tr>
<td>newNodeType</td>
<td>123</td>
</tr>
</tbody>
</table>

**H**

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>hashCode</td>
<td>218</td>
</tr>
<tr>
<td>hasNext</td>
<td>40, 44, 48, 68, 77, 136, 161, 167, 200, 212, 228</td>
</tr>
<tr>
<td>heads</td>
<td>124</td>
</tr>
</tbody>
</table>
newObjects  176
newRestrictedEdgeType  129
newSession  84
newSessionAttribute  121, 123
next  42, 45, 49, 69, 78, 136, 161, 168, 201, 213, 229
nextAttribute  69
nextBoolean  78
nextInt32  135
nextObject  160
nextOID  168
nextString  201
nextType  213
Node  162
NodeExport  141
NodeTypeExporter  252
NodeTypeLoader  256
NotEqual  81
notifyEvent  269, 278
reset  232, 261
restore  179
retainAll  155
Round  147
run  7, 14, 19, 24, 30, 32, 37, 52, 243, 249, 253, 258, 265, 272
runNPhases  249, 259, 271
runTwoPhases  248, 257, 271
sample  158
ScriptParser  280
select  113, 114, 118, 120, 128
set  166, 218
setAsDirected  100
setAttribute  113
setAttributeDefaultValue  118
setAttributePositions  249, 258, 273
setAttributeText  112
setAutoQuotes  237
setBoolean  219
setBooleanVoid  218
setCacheMaxSize  188
setCacheStatisticsEnabled  192
setCacheStatisticsFile  190
setCacheStatisticsSnapshotTime  186
setColor  100, 144
colorRGB  101, 145
defaults  99, 131, 143
double  220
doubleVoid  219
setErrorLog  281
setExtentPages  191
setExtentSize  187
setFit  143
setFontSize  99, 143
setForcedQuotes  238
setFrequency  242, 247, 252, 256, 265, 271
setGraph  242, 248, 253, 258, 266, 272
setHeadAttribute  242, 248
setHeader  243, 253, 265
setHeadPosition  243, 250
setHeight  143

Page 286 of 288
setHighAvailabilityCoordinators 186
setHighAvailabilityEnabled 185
setHighAvailabilityIP 185
setHighAvailabilityMasterHistory 191
setHighAvailabilitySynchronization 189
setInteger 225
setIntegerVoid 223
setLabel 101, 130, 144
setLabelColor 101, 145
setLabelColorRGB 99, 142
setLicense 189
setLocale 235, 238, 249, 258, 273
setLogError 248, 257, 272
setLogFile 193
setLogLevel 187
setLogOff 247, 257, 272
setLong 221
setLongVoid 217
setMaterializedAttribute 7, 33, 37, 53, 57
setMaximumHops 12, 15, 19, 24, 29, 41, 46, 50
setMultilines 233
setNull 224
setNullVoid 220
setNumLines 233
setOID 224
setOIDVoid 221
setOutputLog 282
setPoolFrameSize 189
setPoolPersistentMaxSize 188
setPoolPersistentMinSize 185
setPoolTemporaryMaxSize 189
setPoolTemporaryMinSize 191
setQuotes 234, 237
setRecoveryCacheMaxSize 185
setRecoveryCheckpointTime 193
setRecoveryEnabled 189
setRecoveryLogFile 188
setRowReader 249, 258, 273
setRowWriter 243, 252, 265
setSeparator 233, 237
setShape 145
setSingleLine 234
setStartLine 234
setString 218
setStringVoid 224
setTailAttribute 241, 247
setTailPosition 242, 248
setText 219
setTimestamp 219, 222, 225
setTimestampFormat 250, 259, 273
setTimestampVoid 220, 222
setType 242, 247, 253, 257, 265, 272
setUnweightedEdgeCost 29
setVoid 224
setWidth 99, 142
Severe 138
SinglePairShortestPathBFS 22
SinglePairShortestPathDijkstra 27
size 158
Sparksee 179
SparkseeConfig 184
String 89
StringList 198
StrongConnectivityGabow 36
T
tails 123
tailsAndHeads 115
Text 90
TextStream 203
Timestamp 89
toArray 156, 157
toString 223, 225
TraversalBFS 44
TraversalDFS 48
TypeList 210
U
union 155
Unique 64
V
Value 216, 217
valueOf 64, 82, 90, 103, 108, 139, 147, 163, 170
values 64, 81, 90, 103, 108, 139, 147, 163, 170
Version 179
Warning 138
WeakConnectivityDFS 56
write 203, 238, 262

Y

YGraphML 108