Package
com.sparsity.dex.algorithms
com.sparsity.dex.algorithms
Class ConnectedComponents

java.lang.Object
    +com.sparsity.dex.algorithms.ConnectedComponents

All Implemented Interfaces:
    Closeable

public class ConnectedComponents
extends Object
implements Closeable

ConnectedComponents class.

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 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 DEX User Manual for more details on this.

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

---

### Constructor Summary

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

### Method Summary

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

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

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

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

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

<table>
<thead>
<tr>
<th>boolean method</th>
<th>isClosed()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets if ConnectedComponents instance has been closed or not.</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

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

getCount

public long getCount()  

Returns the number of connected components found in the graph.

Returns:
  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.
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()

getSize

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.

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

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.

close

public void close()

Closes the ConnectedComponents instance.

It must be called to ensure the integrity of all data.
com.sparsity.dex.algorithms
Class Connectivity

java.lang.Object
    +-com.sparsity.dex.algorithms.Connectivity

All Implemented Interfaces:
    Closeable

Direct Known Subclasses:
    WeakConnectivity, StrongConnectivity

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 DEX 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 addAllNodeTypes()</td>
<td>Allows connectivity through all node types of the graph.</td>
</tr>
<tr>
<td>void addNodeType(int t)</td>
<td>Allows connectivity through nodes of the given type.</td>
</tr>
<tr>
<td>void close()</td>
<td>Closes the Connectivity instance.</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>
<tr>
<td>boolean isClosed()</td>
<td>Gets if Connectivity instance has been closed or not.</td>
</tr>
<tr>
<td>void run()</td>
<td>Runs the algorithm in order to find the connected components.</td>
</tr>
<tr>
<td>void setMaterializedAttribute(String attributeName)</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

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.

### 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.
addNodeType
public void addNodeType(int t)

Allows connectivity through nodes of the given type.

Parameters:
   t - null

isClosed
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
public void close()

Closes the Connectivity instance.

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

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.

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.
com.sparsity.dex.algorithms

Class Context

java.lang.Object
   +---com.sparsity.dex.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 DEX User Manual for more details on this.

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

### Constructor Summary

<table>
<thead>
<tr>
<th>public</th>
<th>Context(Session s, long node)</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>close()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Closes the Context instance.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Objects</th>
<th>compute()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the resulting collection of nodes.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static Objects</th>
<th>compute(Session s, long node, TypeList nodeTypes, TypeList edgeTypes, EdgesDirection dir, int maxhops, boolean include)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Helper method to easily compute a context from a node.</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>
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.

Constructors

Context

public Context(Session s,
    Long node)

    Creates a new instance.

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

Methods

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.

compute

public Objects compute()

    Gets the resulting collection of nodes.
Returns:
The resulting collection of nodes.

**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.

**addNodeType**

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

Allows for traversing nodes of the given type.

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

**compute**

```java
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.

**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.

**close**

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

Closes the Context instance.

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

**addAllNodeTypes**

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

Allows for traversing all node types of the graph.

**addAllEdgeTypes**

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

Allows for traversing all edge types of the graph.

Parameters:
  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.

**isClosed**

```java
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.dex.algorithms  
Class ShortestPath

java.lang.Object
   +-com.sparsity.dex.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 DEX User Manual for more details on this.

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

Method Summary

| void  | addEdgeTypes(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 ShortestPath instance.                 |
| 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 ShortestPath instance has been closed or not. |
| void  | run()                               | Runs the algorithm.                              |
void **setMaximumHops** (int maxhops)

Sets the maximum hops restriction.

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**

public void **addAllNodeTypes** ()

Allows for traversing all node types of the graph.

**addAllEdgeTypes**

public void **addAllEdgeTypes** (**EdgesDirection** d)

Allows for traversing all edge types of the graph.

Parameters:

d - [in] Edge direction.

**run**

public void **run** ()

Runs the algorithm.

This method can only be called 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.
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.

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

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.
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.
Class SinglePairShortestPath

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 DEX User Manual for more details on this.

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</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>boolean exists()</td>
<td>Returns TRUE If a path exists or FALSE otherwise.</td>
</tr>
<tr>
<td>double getCost()</td>
<td>Gets the cost of the shortest path.</td>
</tr>
<tr>
<td>OIDList getPathAsEdges()</td>
<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.dex.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

Methods

exists

public boolean exists()

Returns TRUE if a path exists or FALSE otherwise.

run

public void run()

Runs the algorithm.

This method can only be called once.

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.
addNodeType
public void addNodeType(int t)

    Allows for traversing nodes of the given type.

    Parameters:
        t - null

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.

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.

addAllNodeTypes
public void addAllNodeTypes()

    Allows for traversing all node types of the graph.

addAllEdgeTypes
public void addAllEdgeTypes(EdgesDirection d)

    Allows for traversing all edge types of the graph.

    Parameters:
        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.

---

**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.

---

**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
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 DEX 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 SinglePairShortestPathBFS(Session s, long src, long dst)</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 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 checkOnlyExistence()</td>
<td>Set that only the path existence must be calculated and not the path itself.</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>boolean exists()</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.dex.algorithms.SinglePairShortestPath
addAllEdgeTypes, addAllNodeTypes, addEdgeType, addNodeType, excludeEdges, excludeNodes, exists, getCost, getPathAsEdges, getPathAsNodes, run, setMaximumHops

Methods inherited from class com.sparsity.dex.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

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 - [dst] Destination node.

Methods
exists
public boolean exists()

Returns TRUE If a path exists or FALSE otherwise.

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.

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.

addNodeType
public void addNodeType(int t)

Allows for traversing nodes of the given type.

Parameters:
  t - null

gGetCost
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.

---

**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.

---

**addAllNodeTypes**

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

Allows for traversing all node types of the graph.

---

**addAllEdgeTypes**

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

Allows for traversing all edge types of the graph.

**Parameters:**

- `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.
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.

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
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 DEX 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) |
| public | Creates a new instance. |

Method Summary

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

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

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

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

| void | addWeightedEdgeType(int t, EdgesDirection d, int attr) |
| void | Allows for traversing edges of the given type using the given attribute as the weight. |

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

| void | excludeNodes(Objects nodes) |
| void | Set which nodes can't be used. |
```java
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.dex.algorithms.SinglePairShortestPath
addAllEdgeTypes, addAllNodeTypes, addEdgeType, addNodeType, excludeEdges, excludeNodes, exists, getCost, getPathAsEdges, getPathAsNodes, run, setMaximumHops

Methods inherited from class com.sparsity.dex.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, Iong 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**

```java
public boolean exists()
```

Returns TRUE If a path exists or FALSE otherwise.

**addWeightedEdgeType**

```java
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.

**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.

**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.

**addEdgeType**

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

Allows for traversing edges of the given type.

**Parameters:**
- `t` - [in] Edge type.
addNodeType
public void addNodeType(int t)

Allows for traversing nodes of the given type.

Parameters:
  t - null

gGetCost
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.

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.

addAllNodeTypes
public void addAllNodeTypes()

Allows for traversing all node types of the graph.

addAllEdgeTypes
public void addAllEdgeTypes(EdgesDirection d)

Allows for traversing all edge types of the graph.

Parameters:
  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.

---

**run**

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

Executes the algorithm.

---

**setUnweightedEdgeCost**

```java
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

---

**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
# com.sparsity.dex.algorithms.StrongConnectivity

**Class StrongConnectivity**

```java
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 DEX 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 connectivity through 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 connectivity through 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 connectivity through 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 connectivity through nodes of the given type.</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>ConnectedComponents</th>
<th>getConnectedComponents()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Returns the results generated by the execution of the algorithm.</td>
</tr>
</tbody>
</table>
### com.sparsity.dex.algorithms.StrongConnectivity

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>run()</code></td>
<td>Runs the algorithm in order to find the connected components.</td>
</tr>
<tr>
<td><code>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 `com.sparsity.dex.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

**addAllNodeTypes**

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

Allows connectivity through all node types of the graph.

**addAllEdgeTypes**

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

Allows connectivity through all edge types of the graph.

#### Parameters:

- `d` - [in] Edge direction.

**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.

---

### 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.

---

### setMaterializedAttribute

**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.

---

### addNodeType

**public void addNodeType(int t)**

Allows connectivity through nodes of the given type.

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

---

### 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.

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.
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 DEX User Manual for more details on this.

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

Constructor Summary

| public StrongConnectivityGabow(Session s) |
| Creates a new instance of StrongConnectivityGabow |

Method Summary

| void addAllEdgeTypes(EdgesDirection d) |
| Allows connectivity through all edge types of the graph |

| void addAllNodeTypes() |
| Allows connectivity through all node types of the graph |

| void addEdgeType(int t, EdgesDirection d) |
| Allows connectivity through edges of the given type |

| void addNodeType(int t) |
| Allows connectivity through 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 |
null
### addAllEdgeTypes

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

Allows connectivity through all edge types of the graph.

**Parameters:**
- `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.

### 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.

### 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.
**addNodeType**

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

Allows connectivity through nodes of the given type.

**Parameters:**

- `t` - null

**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.

**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.dex.algorithms

Class Traversal

java.lang.Object
--- com.sparsity.dex.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 DEX 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 Traversal 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>int getCurrentDepth()</td>
</tr>
<tr>
<td>Returns the depth of the current node.</td>
</tr>
<tr>
<td>boolean hasNext()</td>
</tr>
<tr>
<td>Gets if there are more objects to be traversed.</td>
</tr>
</tbody>
</table>
boolean 

isClosed() 

Gets if Traversal instance has been closed or not.

long 

next() 

Gets the next object of the traversal.

void 

setMaxHop(int maxhops) 

Sets the maximum hops restriction.

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

hasNext

public boolean hasNext()  

Gets if there are more objects to be traversed.

Returns:
TRUE if there are more objects, 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.

addNodeType

public void addNodeType(int t)  

Allows for traversing nodes of the given type.

Parameters:
  t - null
**getCurrentDepth**

```java
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.

**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.

**close**

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

Closes the Traversal instance.

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

**addAllNodeTypes**

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

Allows for traversing all node types of the graph.

**addAllEdgeTypes**

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

Allows for traversing all edge types of the graph.

**Parameters:**

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.

next

public long next()

    Gets the next object of the traversal.

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

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.

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()
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 DEX User Manual for more details on this.

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

Constructor Summary

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

Method Summary

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

publicTraversalBFS(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

addAllNodeTypes

public voidaddAllNodeTypes()

Allows for traversing all node types of the graph.

addAllEdgeTypes

public voidaddAllEdgeTypes(EdgesDirection d)

Allows for traversing all edge types of the graph.

Parameters:
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

```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.

### next

```java
public long next()
```

Gets the next object of the traversal.

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

### 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.

### addNodeType

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

Allows for traversing nodes of the given type.

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

### getCurrentDepth

```java
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.

---

**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.dex.algorithms
Class TraversalDFS

java.lang.Object
com.sparsity.dex.algorithms.Traversal

com.sparsity.dex.algorithms.TraversalDFS

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 DEX User Manual for more details on this.

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

Constructor Summary

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

Method Summary

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

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

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

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

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

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

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

<table>
<thead>
<tr>
<th>boolean</th>
<th>hasNext()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gets if there are more objects to be traversed.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>long</th>
<th>next()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gets the next object of the traversal.</td>
<td></td>
</tr>
</tbody>
</table>
### void setMaximumHops(int maxhops)
Sets the maximum hops restriction.

### Methods inherited from class com.sparsity.dex.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

#### addAllNodeTypes

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

Allows for traversing all node types of the graph.

#### addAllEdgeTypes

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

Allows for traversing all edge types of the graph.

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

#### hasNext

```java
public boolean hasNext()
```
Gets if there are more objects to be traversed.

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

### 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.

### next

```java
public long next()
```

Gets the next object of the traversal.

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

### 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.

### addNodeType

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

Allows for traversing nodes of the given type.

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

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

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 DEX 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()</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>
<tr>
<td>void excludeNodes(Objects nodes)</td>
</tr>
<tr>
<td>Set which nodes can't be used.</td>
</tr>
<tr>
<td>ConnectedComponents getConnectedComponents()</td>
</tr>
<tr>
<td>Returns the results generated by the execution of the algorithm.</td>
</tr>
<tr>
<td>Method</td>
</tr>
<tr>
<td>--------------------------------</td>
</tr>
<tr>
<td><strong>run()</strong></td>
</tr>
<tr>
<td><strong>setMaterializedAttribute()</strong></td>
</tr>
</tbody>
</table>

**Methods inherited from class** com.sparsity.dex.algorithms.Connectivity

- addAllNodeTypes
- addNodeType
- close
- excludeEdges
- excludeNodes
- getConnectedComponents
- isClosed

**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
default void addAllNodeTypes ()
```

Allows connectivity through all node types of the graph.

#### addAllEdgeTypes

```java
default 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.

#### run

```java
default void run ()
```

Runs the algorithm in order to find the connected components.

This method can be called only once.

#### excludeNodes

```java
default void excludeNodes (Object 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.

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[in] Edge direction.

Parameters:

  t - [in] Edge type.

setMaterializedAttribute

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.

addNodeType

public void addNodeType(int t)

Allows connectivity through nodes of the given type.

Parameters:

  t - null

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.

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.
com.sparsity.dex.algorithms

Class WeakConnectivityDFS

java.lang.Object
   +-com.sparsity.dex.algorithms_CONNECTIVITY
      +-com.sparsity.dex.algorithms.WeakConnectivity
         +-com.sparsity.dex.algorithms.WeakConnectivityDFS

All Implemented Interfaces:
   Closeable

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 DEX User Manual for more details on this.

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

Constructor Summary

| public WeakConnectivityDFS(Session s) |
| Creates a new instance of WeakConnectivityDFS. |

Method Summary

| void addAllEdgeTypes() |
| Allows connectivity through all edge types of the graph. |

| void addAllNodeTypes() |
| Allows connectivity through all node types of the graph. |

| void addEdgeType(int t) |
| Allows connectivity through edges of the given type. |

| void addNodeType(int t) |
| Allows connectivity through 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.

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.dex.algorithms.WeakConnectivity
addAllEdgeTypes, addAllNodeTypes, addEdgeType, addNodeType, excludeEdges, excludeNodes, getConnectedComponents, run, setMaterializedAttribute

Methods inherited from class com.sparsity.dex.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

Methods inherited from interface java.io.Closeable
close

Constructors

WeakConnectivityDFS
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

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[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.

run

public void run()

Executes the algorithm.

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[in] Edge direction.

Parameters:
t - [in] Edge type.

setMaterializedAttribute

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.
addNodeType

public void addNodeType(int t)

Allows connectivity through nodes of the given type.

Parameters:

t - null

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.

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.
Package
com.sparsity.dex.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

| public static | InvalidAttribute | Invalid attribute identifier constant. |

Method Summary

<table>
<thead>
<tr>
<th>long</th>
<th>getCount()</th>
<th>Gets the number of non-NULL values.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataType</td>
<td>getDataType()</td>
<td>Gets the data type.</td>
</tr>
<tr>
<td>int</td>
<td>getId()</td>
<td>Gets the Dex attribute identifier.</td>
</tr>
<tr>
<td>AttributeKind</td>
<td>getKind()</td>
<td>Gets the attribute kind.</td>
</tr>
<tr>
<td>String</td>
<td>getName()</td>
<td>Gets the unique attribute name.</td>
</tr>
<tr>
<td>long</td>
<td>getSize()</td>
<td>Gets the number of different values.</td>
</tr>
<tr>
<td>int</td>
<td>getType()</td>
<td>Gets the Dex type identifier.</td>
</tr>
<tr>
<td>boolean</td>
<td>isSessionAttribute()</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

Fields
InvalidAttribute

```java
public static int InvalidAttribute
```

Invalid attribute identifier constant.

## Methods

### getKind

```java
public AttributeKind getKind()
```

Gets the attribute kind.

**Returns:**
The AttributeKind.

### getCount

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

Gets the number of non-NULL values.

**Returns:**
The number of non-NULL values.

### isSessionAttribute

```java
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

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

Gets the number of different values.

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

### getTypeID

```java
public int getTypeID()
```
Gets the Dex type identifier.

**Returns:**
The Dex type identifier.

---

**getDataType**

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

Gets the data type.

**Returns:**
The DataType.

---

**getId**

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

Gets the Dex attribute identifier.

**Returns:**
The Dex 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

Field Summary

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

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static</td>
<td>AttributeKind.valueOf(String name)</td>
</tr>
<tr>
<td>static</td>
<td>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.dex.gdb.AttributeKind Basic

Basic attribute (non indexed attribute).

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

Indexed attribute.

Unique
public static final com.sparsity.dex.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)
com.sparsity.dex.gdb
Class AttributeList

java.lang.Object
- com.sparsity.dex.gdb.AttributeList

All Implemented Interfaces:
   Iterable

public class AttributeList
extends Object
implements Iterable

Dex attribute identifier list.

It stores a Dex 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>public</th>
<th>AttributeList &lt;br&gt;(Collection col)</th>
<th>Creates a new instance from an integer collection.</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
<td>AttributeList &lt;br&gt;()</td>
<td>Constructor.</td>
</tr>
<tr>
<td>public</td>
<td>AttributeList &lt;br&gt;(int[] list)</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 attr)</th>
<th>Adds a Dex attribute identifier at the end of the list.</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>clear()</td>
<td>Clears the list.</td>
</tr>
<tr>
<td>int</td>
<td>count()</td>
<td>Number of elements in the list.</td>
</tr>
<tr>
<td></td>
<td>AttributeListIterator &lt;br&gt;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

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 Dex attribute identifier at the end of the list.

**Parameters:**
- `attr` - [in] Dex attribute identifier.
public class AttributeListIterator extends Object implements Iterator

AttributeList iterator class.

Iterator to traverse all the Dex attribute identifier into a AttributeList 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 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**

```java
public void remove()
```

Operation not supported.

**next**

```java
public Integer next()
```

See `nextAttribute()`.

**nextAttribute**

```java
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 method Graph::getAttributeStatistics or check out the DEX User Manual for more details on this.

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

### Method Summary

<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
<th>Description</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>int</td>
<td>getMax()</td>
<td>Gets the maximum existing value (BASIC statistics).</td>
</tr>
<tr>
<td>double</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>
</tbody>
</table>
long getTotal()

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

double getVariance()

Gets the variance.

Methods inherited from class java.lang.Object

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

Methods

getMin

public Value getMin()

Gets the minimum existing value (BASIC statistics).

Returns:
The minimum existing value.

getMinLengthString

public int getMinLengthString()

Gets the minimum length.

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

Returns:
The minimum length.

getVariance

public double getVariance()

Gets the variance.

It is computed just for numerical attributes.

Returns:
The variance.

getMode

public Value getMode()

Gets the mode.

Mode: Most frequent Value.

Returns:
The mode.
getPage

public long getPage()

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

Returns:
The number of objects NULL a Value.

getDistinct

public long getDistinct()

Gets the number of distinct values (BASIC statistics).

Returns:
The number of distinct values.

getMean

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

public Value getMax()

Gets the maximum existing value (BASIC statistics).

Returns:
The maximum existing value.

getMedian

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**

```java
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**

```java
public int getMaxLengthString()
```

Gets the maximum length.

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

**Returns:**
The maximum length.

**getAvgLengthString**

```java
public double getAvgLengthString()
```

Gets the average length.

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

**Returns:**
The average length.

**getModeCount**

```java
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.dex.gdb
Class BooleanList

java.lang.Object
        -com.sparsity.dex.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>public</th>
<th>BooleanList(Collection col)</th>
<th>Creates a new instance from a boolean collection.</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
<td>BooleanList(boolean[] list)</td>
<td>Creates a new instance from a boolean array.</td>
</tr>
<tr>
<td>public</td>
<td>BooleanList()</td>
<td>Constructor.</td>
</tr>
</tbody>
</table>

Method Summary

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

Methods inherited from interface java.lang.Iterable

iterator
Constructors

**BooleanList**

```java
public BooleanList(Collection col)
```

Creates a new instance from a boolean collection.

**Parameters:**
- `col` - Collection to initialize the instance.

**BooleanList**

```java
public BooleanList(boolean[] list)
```

Creates a new instance from a boolean array.

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

**BooleanList**

```java
public BooleanList()
```

Constructor.

This creates an empty list.

Methods

**add**

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

Adds a Boolean at the end of the list.

**Parameters:**
- `b` - [in] Boolean.

**clear**

```java
public void clear()
```

Clears the list.

**iterator**

```java
public BooleanListIterator iterator()
```
Gets a new BooleanListIterator.

**Returns:**
BooleanListIterator instance.

count

```java
public int count()
```

Number of elements in the list.

**Returns:**
Number of elements in the list.
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>hasNext()</td>
<td>Gets if there are more elements.</td>
</tr>
<tr>
<td>next()</td>
<td>See nextBoolean().</td>
</tr>
<tr>
<td>nextBoolean()</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 Boolean next()

    See nextBoolean().

nextBoolean
public boolean nextBoolean()

    Gets the next element.
### Field Summary

<table>
<thead>
<tr>
<th>Field Summary</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>public static final</strong> Between</td>
<td>In range operator condition ([x,y]).</td>
</tr>
<tr>
<td><strong>public static final</strong> Equal</td>
<td>Equal condition (==).</td>
</tr>
<tr>
<td><strong>public static final</strong> GreaterEqual</td>
<td>Greater or equal condition (&gt;=).</td>
</tr>
<tr>
<td><strong>public static final</strong> GreaterThan</td>
<td>Greater than condition (&gt;).</td>
</tr>
<tr>
<td><strong>public static final</strong> LessEqual</td>
<td>Less or equal condition (&lt;=).</td>
</tr>
<tr>
<td><strong>public static final</strong> LessThan</td>
<td>Less than condition (&lt;).</td>
</tr>
<tr>
<td><strong>public static final</strong> Like</td>
<td>Substring condition.</td>
</tr>
<tr>
<td><strong>public static final</strong> LikeNoCase</td>
<td>Substring (no case sensitive) condition.</td>
</tr>
<tr>
<td><strong>public static final</strong> NotEqual</td>
<td>Not equal condition (!=).</td>
</tr>
<tr>
<td><strong>public static final</strong> RegExp</td>
<td>Regular expression condition.</td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Method Summary</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static Condition</td>
<td>valueOf(String name)</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

Equal
public static final com.sparsity.dex.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.dex.gdb.Condition GreaterEqual

Greater or equal condition (>=).
Null values cannot be used together with this condition.

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

Greater than condition (>).
Null values cannot be used together with this condition.

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

Less or equal condition (<=).
Null values cannot be used together with this condition.

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

Less than condition (<).
Null values cannot be used together with this condition.
NotEqual

public static final com.sparsity.dex.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.dex.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.dex.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.dex.gdb.Condition Between

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

Null values cannot be used together with this condition.

RegExp

public static final com.sparsity.dex.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 'DEX 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 Dex 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 DEXConfig.

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

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>void close()</td>
</tr>
<tr>
<td>close the Database instance.</td>
</tr>
<tr>
<td>String getAlias()</td>
</tr>
<tr>
<td>get the alias of the Database.</td>
</tr>
<tr>
<td>String getPath()</td>
</tr>
<tr>
<td>gets the path of the Database.</td>
</tr>
<tr>
<td>void getStatistics(DatabaseStatistics stats)</td>
</tr>
<tr>
<td>gets Database statistics.</td>
</tr>
<tr>
<td>boolean isClosed()</td>
</tr>
<tr>
<td>checks if Database instance has been closed or not.</td>
</tr>
<tr>
<td>Session 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
public String getAlias()

    Gets the alias of the Database.

    Returns:
    The alias of the Database.

getPath
public String getPath()

    Gets the path of the Database.

    Returns:
    The path of the Database.

newSession
public Session newSession()

    Creates a new Session.

isClosed
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
public void close()

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

getStatistics
public void getStatistics(DatabaseStatistics stats)
Gets Database statistics.

Parameters:

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>Type</th>
<th>Method</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

getWrite

public long getWrite()

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

getData
public long getData()

    Gets database size in KBytes.

    Returns:
    Database size in KBytes.

tTemp
public long getTemp()

    Gets temporary storage file size in KBytes.

    Returns:
    Temporary storage file size in KBytes.

tRead
public long getRead()

    Gets total read data in KBytes.

    Returns:
    Total read data in KBytes.

tCache
public long getCache()

    Gets cache size in KBytes.

    Returns:
    Cache size in KBytes.
### com.sparsity.dex.gdb

#### Class DataType

```
java.lang.Object
   ^-java.lang.Enum
      ^-com.sparsity.dex.gdb.DataType
```

All Implemented Interfaces:
   Serializable, Comparable

Public final class **DataType**

extends Enum

Data type (domain) enumeration.

**Author:**
Sparsity Technologies [http://www.sparsity-technologies.com](http://www.sparsity-technologies.com)

---

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

---

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

**Boolean**

```java
class Boolean {
public static final com.sparsity.dex.gdb.DataType BOOLEAN = ...
}
```

Boolean data type.

**Integer**

```java
class Integer {
public static final com.sparsity.dex.gdb.DataType INTEGER = ...
}
```

32-bit signed integer data type.

**Long**

```java
class Long {
public static final com.sparsity.dex.gdb.DataType LONG = ...
}
```

64-bit signed integer data type.

**Double**

```java
class Double {
public static final com.sparsity.dex.gdb.DataType DOUBLE = ...
}
```

64-bit signed double data type.

**Timestamp**

```java
class Timestamp {
public static final com.sparsity.dex.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.dex.gdb.DataType String

  Unicode string data type.
  2048 characters maximum length.

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

  Large unicode character object (CLOB) data type.

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

  Object identifier (OID) data type.

Methods

values
public static DataType[] values()
public class DefaultExport
extends ExportManager

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>Constructor</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
<td>DefaultExport ()</td>
</tr>
<tr>
<td></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>boolean enableType (int type)</td>
<td>ExportManager::EnableType.</td>
</tr>
<tr>
<td>boolean getEdge (long edge, EdgeExport edgeExport)</td>
<td>ExportManager::GetEdge.</td>
</tr>
<tr>
<td>boolean getEdgeType (int type, EdgeExport edgeExport)</td>
<td>ExportManager::GetEdgeType.</td>
</tr>
<tr>
<td>boolean getGraph (GraphExport graphExport)</td>
<td>ExportManager::GetGraph.</td>
</tr>
<tr>
<td>boolean getNode (long node, NodeExport nodeExport)</td>
<td>ExportManager::GetNode.</td>
</tr>
<tr>
<td>boolean getNodeType (int type, NodeExport nodeExport)</td>
<td>ExportManager::GetNodeType.</td>
</tr>
<tr>
<td>void prepare (Graph graph)</td>
<td>ExportManager::Prepare.</td>
</tr>
<tr>
<td>void release ()</td>
<td>ExportManager::Release.</td>
</tr>
</tbody>
</table>

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

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

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

Constructors

DefaultExport

public DefaultExport ()

Creates a new instance.

Methods

enableType

public boolean enableType(int type)

ExportManager::EnableType.

This enables all node and edge types to be exported.

Parameters:
  type - null

Returns:
  TRUE.

getEdge

public boolean getEdge(long edge,
  EdgeExport edgeExport)

ExportManager::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)

ExportManager::GetGraph.

This sets the default GraphExport values and "Graph" as the label.

Parameters:
  graphExport - null
**getEdgeType**

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

ExportManager::GetEdgeType.

This sets the default EdgeExport values.

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

---

**getNodeType**

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

ExportManager::GetNodeType.

This sets the default NodeExport values.

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

---

**release**

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

ExportManager::Release.

---

**getNode**

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

ExportManager::GetNode.

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

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

**Returns:**
- TRUE.
graph - null
com.sparsity.dex.gdb
Class Dex

java.lang.Object
  +-com.sparsity.dex.gdb.Dex

All Implemented Interfaces:
  Closeable

public class Dex
  extends Object
  implements Closeable

Dex class.

All Dex programs must have one single Dex 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>public static</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Dex version.</td>
</tr>
</tbody>
</table>

Constructor Summary

<table>
<thead>
<tr>
<th>public</th>
<th>Dex(DexConfig config)</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>close()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Closes the Dex instance.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Database</th>
<th>create(String path, String alias)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Creates a new Database instance.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>isClosed()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets if Dex instance has been closed or not.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Database</th>
<th>open(String path, boolean read)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Opens an existing Database instance.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Database</th>
<th>restore(String path, String backupFile)</th>
</tr>
</thead>
<tbody>
<tr>
<td></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

Dex version.

Constructors

Dex
public Dex(DexConfig config)

Creates a new instance.

Parameters:
config - [in] Dex 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 Graph::Backup.

**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 exist.
- `java.lang.RuntimeException` - null

---

### isClosed

```java
class Dex
{
    public boolean isClosed()
    {
        return true; // Changed return value to TRUE
    }
}
```

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

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

---

### close

```java
class Dex
{
    public void close()
    {
        // Close the Dex instance
    }
}
```

**Closes the Dex instance.**

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

---

### open

```java
class Database
{
    public Database open(String path, boolean read)
    {
        // Open 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 DexConfig
extends Object

Dex 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 DexProperties 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 ('dex.storage.extentsize' at DexProperties).
Pages per extent: 1 page ('dex.storage.extentpages' at DexProperties).
Pool frame size: 1 extent ('dex.io.pool.frame.size' at DexProperties).
Minimum size for the persistent pool: 64 frames ('dex.io.pool.persistent.minsize' at DexProperties).
Maximum size for the persistent pool: 0 frames ('dex.io.pool.persistent.maxsize' at DexProperties).
Minimum size for the temporary pool: 16 frames ('dex.io.pool.temporal.minsize' at DexProperties).
Maximum size for the temporary pool: 0 frames ('dex.io.pool.temporal.maxsize' at DexProperties).
Maximum size for the cache (all pools): 0 MB ('dex.io.cache.maxsize' at DexProperties).
License code: "" ('dex.license' at DexProperties). No license code means evaluation license.
Log level: LogLevel::Info ('dex.log.level' at DexProperties).
Log file: "dex.log" ('dex.log.file' at DexProperties).
Recovery enabled: false ('dex.io.recovery' at DexProperties).
Recovery log file: "" ('dex.io.recovery.logfile' at DexProperties).
Recovery cache max size: 1MB ('dex.io.recovery.cachesize' at DexProperties).
Recovery checkpoint time: 60 seconds [TimeUnit] ('dex.io.recovery.checkpointTime' at DexProperties).
High-availability: false (disabled) ('dex.ha' at DexProperties).
High-availability coordinators: "" ('dex.ha.coordinators' at DexProperties).
High-availability IP: "" (‘dex.ha.ip’ at DexProperties).

High-availability sync polling: 0 (disabled) [TimeUnit] (‘dex.ha.sync’ at DexProperties).

High-availability master history: 1D (1 day) [TimeUnit] (‘dex.ha.master.history’ at DexProperties).

Use of TimeUnit:

Those variables using TimeUnit allow for:

[D|H|M|S|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>Type</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
<td>DexConfig()</td>
</tr>
<tr>
<td></td>
<td>Creates a new instance.</td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Return Type</th>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>getCacheMaxSize()</td>
<td>Gets the maximum size for the cache (all pools) in MB.</td>
</tr>
<tr>
<td>boolean</td>
<td>getCacheStatisticsEnabled()</td>
<td>Gets whether cache statistics are enabled or disabled.</td>
</tr>
<tr>
<td>long</td>
<td>getCacheStatisticsSnapshotTime()</td>
<td>Gets the cache statistics snapshot time in microseconds.</td>
</tr>
<tr>
<td>int</td>
<td>getExtentPages()</td>
<td>Gets the number of pages per extent.</td>
</tr>
<tr>
<td>int</td>
<td>getExtentSize()</td>
<td>Gets the size of a extent.</td>
</tr>
<tr>
<td>String</td>
<td>getHighAvailabilityCoordinators()</td>
<td>Gets the coordinators address and port list.</td>
</tr>
<tr>
<td>boolean</td>
<td>getHighAvailabilityEnabled()</td>
<td>Gets whether high availability mode is enabled or disabled.</td>
</tr>
<tr>
<td>String</td>
<td>getHighAvailabilityIP()</td>
<td>Gets the IP address and port of the instance.</td>
</tr>
<tr>
<td>long</td>
<td>getHighAvailabilityMasterHistory()</td>
<td>Gets the master's history log.</td>
</tr>
<tr>
<td>long</td>
<td>getHighAvailabilitySynchronization()</td>
<td>Gets the synchronization polling time.</td>
</tr>
<tr>
<td>String</td>
<td>getLicense()</td>
<td>Gets the license code.</td>
</tr>
</tbody>
</table>
String getLogFile()
    Gets the log file.

LogLevel getLogLevel()
    Gets the log level.

int getPoolFrameSize()
    Gets the size of a pool frame in number of extents.

int getPoolPersistentMaxSize()
    Gets the maximum size for the persistent pool in number of frames.

int getPoolPersistentMinSize()
    Gets the minimum size for the persistent pool in number of frames.

int getPoolTemporaryMaxSize()
    Gets the maximum size for the temporary pool in number of frames.

int getPoolTemporaryMinSize()
    Gets the minimum size for the temporary pool in number of frames.

int getRecoveryCacheMaxSize()
    Gets the maximum size for the recovery log cache in extents.

long getRecoveryCheckpointTime()
    Gets the delay time (in microseconds) between automatic checkpoints.

boolean getRecoveryEnabled()
    Gets whether the recovery is enabled or disabled.

String getRecoveryLogFile()
    Gets the recovery log file.

void setCacheMaxSize(int v)
    Sets the maximum size for the cache (all pools) in MB.

void setCacheStatisticsEnabled(boolean v)
    Enables or disables cache statistics.

void setCacheStatisticsFile(String v)
    Sets the cache statistics log file.

void setCacheStatisticsSnapshotTime(long v)
    Sets the cache statistics snapshot time.

void setExtentPages(int v)
    Sets the number of pages per extent.

void setExtentSize(int v)
    Sets the size of a pool frame in number of extents.

void setHighAvailabilityCoordinators(String v)
    Sets the coordinators address and port list.

void setHighAvailabilityEnabled(boolean v)
    Enables or disables high availability mode.

void setHighAvailabilityIP(String v)
    Sets the IP address and port of the instance.
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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

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

### Constructors

**DexConfig**

```java
public DexConfig()
```

Creates a new instance.

Values are set with default values.
Methods

getHighAvailabilitySynchronization

```java
public long getHighAvailabilitySynchronization()
```

Gets the synchronization polling time.

**Returns:**
The Synchronization polling time.

setRecoveryCacheMaxSize

```java
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

```java
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

```java
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

```java
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:**

v - [in] The LogLevel.

---

**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**

```java
public String getHighAvailabilityCoordinators()
```

Gets the coordinators address and port list.

**Returns:**

The coordinators address and port list.

**setLicense**

```java
public void setLicense(String v)
```

Sets the license code.

**Parameters:**

- `v` - [in] The license code.

**setRecoveryEnabled**

```java
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**

```java
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**

```java
public void setHighAvailabilitySynchronization(long v)
```

Sets the synchronization polling time.

**Parameters:**

- `v` - [in] The synchronization polling time.

**setPoolFrameSize**

```java
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**

```java
public String getCacheStatisticsFile()
```

Gets the cache statistics log file.

Useless if cache statistics are disabled.

**Returns:**

The cache statistics log file.

**setCacheStatisticsFile**

```java
public void setCacheStatisticsFile(String v)
```

Sets the cache statistics log file.

Useless if cache statistics are disabled.

**Parameters:**


**getPoolFrameSize**

```java
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**

```java
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**

```java
public long getCacheStatisticsSnapshotTime()
```

Gets the cache statistics snapshot time in microseconds.

Useless if cache statistics are disabled.

**Returns:**
getPoolTemporaryMaxSize

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

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

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

public String getLicense()

Gets the license code.

Returns:  
The license code.

getPoolTemporaryMinSize

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**

```java
public void setHighAvailabilityMasterHistory(long v)
```

Sets the master's history log.

**Parameters:**
- `v` - [in] The master's history log.

---

**setCacheStatisticsEnabled**

```java
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**

```java
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**

```java
public long getHighAvailabilityMasterHistory()
```

Gets the master's history log.

**Returns:**
- The master's history log.

---

**getHighAvailabilityIP**

```java
public String getHighAvailabilityIP()
```

Gets the IP address and port of the instance.

**Returns:**
- The IP address and port of the instance.

---

**getLogLevel**

```java
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.
Dex 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 './dex.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>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static String get(String key, String def)</td>
<td>Gets a property.</td>
</tr>
<tr>
<td>static boolean getBoolean(String key, boolean def)</td>
<td>Gets a property as a boolean.</td>
</tr>
<tr>
<td>static int getInteger(String key, int def)</td>
<td>Gets a property as an integer.</td>
</tr>
<tr>
<td>static long getTimeUnit(String key, long def)</td>
<td>Gets a property as a time unit.</td>
</tr>
<tr>
<td>static void 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, wait

### Methods

**get**

```java
public static String get(String key, String def)
```

Gets a property.
**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' or '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**

```java
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.
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

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>long</td>
<td>getEdge()</td>
</tr>
<tr>
<td></td>
<td>Gets the edge identifier.</td>
</tr>
<tr>
<td>long</td>
<td>getHead()</td>
</tr>
<tr>
<td></td>
<td>Gets the head of the edge.</td>
</tr>
<tr>
<td>long</td>
<td>getTail()</td>
</tr>
<tr>
<td></td>
<td>Gets the tail of the edge.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

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

Methods

getHead

public long getHead()

Gets the head of the edge.

Returns:
The Dex edge identifier of the head of the edge.

getTail

public long getTail()

Gets the tail of the edge.
Returns:
The Dex edge identifier of the tail of the edge.

getEdge

class public long getEdge()

Gets the edge identifier.

Returns:
The Dex edge identifier.
com.sparsity.dex.gdb

Class EdgeExport

defined in java.lang.Object

|--.com.sparsity.dex.gdb.EdgeExport

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>Method</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
<td>EdgeExport()</td>
</tr>
</tbody>
</table>

**EdgeExport()**

Creates a new instance.

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>asDirected()</td>
</tr>
<tr>
<td></td>
<td>Gets if the edge should be managed as directed.</td>
</tr>
<tr>
<td>java.awt.Color</td>
<td>getColor()</td>
</tr>
<tr>
<td></td>
<td>Gets the color of the edge.</td>
</tr>
<tr>
<td>int</td>
<td>getColorRGB()</td>
</tr>
<tr>
<td></td>
<td>Gets the edge color.</td>
</tr>
<tr>
<td>int</td>
<td>getFontSize()</td>
</tr>
<tr>
<td></td>
<td>Gets the edge label font size.</td>
</tr>
<tr>
<td>String</td>
<td>getLabel()</td>
</tr>
<tr>
<td></td>
<td>Gets the edge label.</td>
</tr>
<tr>
<td>java.awt.Color</td>
<td>getLabelColor()</td>
</tr>
<tr>
<td></td>
<td>Gets the color of the label.</td>
</tr>
</tbody>
</table>
### Methods

<table>
<thead>
<tr>
<th>int</th>
<th>getLabelColorRGB()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the edge label color.</td>
</tr>
<tr>
<td>int</td>
<td>getWidth()</td>
</tr>
<tr>
<td></td>
<td>Gets the edge width.</td>
</tr>
<tr>
<td>void</td>
<td>setAsDirected(boolean b)</td>
</tr>
<tr>
<td></td>
<td>Sets if the edge should be managed as directed.</td>
</tr>
<tr>
<td>void</td>
<td>setColor(java.awt.Color c)</td>
</tr>
<tr>
<td></td>
<td>Sets the color of the edge.</td>
</tr>
<tr>
<td>void</td>
<td>setColorRGB(int c)</td>
</tr>
<tr>
<td></td>
<td>Sets the edge 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>setFontSize(int s)</td>
</tr>
<tr>
<td></td>
<td>Sets the edge label font size.</td>
</tr>
<tr>
<td>void</td>
<td>setLabel(String l)</td>
</tr>
<tr>
<td></td>
<td>Sets the edge 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 edge label color.</td>
</tr>
<tr>
<td>void</td>
<td>setWidth(int w)</td>
</tr>
<tr>
<td></td>
<td>Sets the edge width.</td>
</tr>
</tbody>
</table>

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

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

### Constructors

**EdgeExport**

```java
public EdgeExport()

Creates a new instance.
```

### Methods

**getColor**

```java
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.

getColorRGB

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
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.dex.gdb
Class EdgesDirection

java.lang.Object
   +--java.lang.Enum
   |   +--com.sparsity.dex.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

| public static final | Any             |
|                     | In-going or out-going edges. |
| public static final | Ingoing         |
|                     | In-going edges.  |
| public static final | Outgoing        |
|                     | Out-going edges.|

Method Summary

| static EdgesDirection | valuesOf(String name) |
| static EdgesDirection[] | 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
Ingoing
public static final com.sparsity.dex.gdb.EdgesDirection Ingoing

In-going edges.

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

Out-going edges.

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

In-going or out-going edges.

Methods

values
public static EdgesDirection[] values()

valueOf
public static EdgesDirection valueOf(String name)
com.sparsity.dex.gdb
Class ExportManager

java.lang.Object
    +com.sparsity.dex.gdb.ExportManager

Direct Known Subclasses:
    DefaultExport

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 process, 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

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean enableType(int type)</td>
</tr>
<tr>
<td>boolean getEdge(long edge, EdgeExport edgeExport)</td>
</tr>
<tr>
<td>boolean getEdgeType(int type, EdgeExport edgeExport)</td>
</tr>
<tr>
<td>boolean getGraph(GraphExport graphExport)</td>
</tr>
<tr>
<td>boolean getNode(long node, NodeExport nodeExport)</td>
</tr>
<tr>
<td>boolean getNodeType(int type, NodeExport nodeExport)</td>
</tr>
<tr>
<td>void prepare(Graph graph)</td>
</tr>
<tr>
<td>void release()</td>
</tr>
</tbody>
</table>

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

Methods
**getNodeType**

```java
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**

```java
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**

```java
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**

```java
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

```java
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

```java
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

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

Ends the export process.

It is called once after the export process.

---

### enableType

```java
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.dex.gdb
Class ExportType

java.lang.Object
   +-java.lang.Enum
   |   +-com.sparsity.dex.gdb.ExportType

All Implemented Interfaces:
   Serializable, Comparable

public final class ExportType
extends Enum

Export type.

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

---

Field Summary

<table>
<thead>
<tr>
<th>public static final</th>
<th>GraphML</th>
</tr>
</thead>
<tbody>
<tr>
<td>GraphML</td>
<td>Export to GraphML format.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public static final</th>
<th>Graphviz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphviz</td>
<td>Export to Graphviz format.</td>
</tr>
</tbody>
</table>

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

---

Method Summary

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

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

```java
public static final com.sparsity.dex.gdb.ExportType Graphviz
```

Export to Graphviz format.

Graphviz home page: http://www.graphviz.org

GraphML

```java
public static final com.sparsity.dex.gdb.ExportType GraphML
```

Export to GraphML format.

GraphML home page: http://graphml.graphdrawing.org/

YGraphML

```java
public static final com.sparsity.dex.gdb.ExportType YGraphML
```

Export to YGRAPHML format.

YGraphML is an 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**

```java
public static ExportType[] values()
```

**valueOf**

```java
public static ExportType valueOf(String name)
```
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 'DEX graph database' sections in the DEX User Manual for more details on the use of this class.

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

---

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>backup(String file)</td>
<td>Dumps all the data to a backup file.</td>
</tr>
<tr>
<td>long</td>
<td>countEdges()</td>
<td>Gets the number of edges.</td>
</tr>
<tr>
<td>long</td>
<td>countNodes()</td>
<td>Gets the number of nodes.</td>
</tr>
<tr>
<td>long</td>
<td>degree(long oid, int etype, EdgesDirection dir)</td>
<td>Gets the number of edges from or to the given node OID and for the given edge type.</td>
</tr>
<tr>
<td>void</td>
<td>drop(long oid)</td>
<td>Drops the given OID.</td>
</tr>
<tr>
<td>void</td>
<td>drop(Objects objs)</td>
<td>Drops all the OIDs from the given collection.</td>
</tr>
<tr>
<td>void</td>
<td>dumpData(String file)</td>
<td>Dumps logical data to a file.</td>
</tr>
<tr>
<td>void</td>
<td>dumpStorage(String file)</td>
<td>Dumps internal storage data to a file.</td>
</tr>
<tr>
<td>Objects</td>
<td>edges(int etype, long tail, long head)</td>
<td>Gets all the edges of the given type between two given nodes (tail and head).</td>
</tr>
<tr>
<td>Objects</td>
<td>explode(long oid, int etype, EdgesDirection dir)</td>
<td>Selects all edges from or to the given node OID and for the given edge type.</td>
</tr>
<tr>
<td>Objects</td>
<td>explode(Objects objs, int etype, EdgesDirection dir)</td>
<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>void</td>
<td>export(String file, ExportType type, ExportManager e)</td>
<td>Exports the Graph.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>int  <code>findAttribute(int type, String name)</code></td>
<td>Gets the Dex attribute identifier for the given type identifier and attribute name.</td>
<td></td>
</tr>
<tr>
<td><code>AttributeList findAttributes(int type)</code></td>
<td>Gets all existing Dex attribute identifiers for the given type identifier.</td>
<td></td>
</tr>
<tr>
<td>long <code>findEdge(int etype, long tail, long head)</code></td>
<td>Gets any of the edges of the given type between two given nodes (tail and head).</td>
<td></td>
</tr>
<tr>
<td><code>TypeList findEdgeTypes()</code></td>
<td>Gets all existing Dex edge type identifiers.</td>
<td></td>
</tr>
<tr>
<td><code>TypeList findNodeTypes()</code></td>
<td>Gets all existing Dex node type identifiers.</td>
<td></td>
</tr>
<tr>
<td>long <code>findObject(int attr, Value v)</code></td>
<td>Finds one object having the given Value for the given attribute.</td>
<td></td>
</tr>
<tr>
<td>int  <code>findType(String name)</code></td>
<td>Gets the Dex type identifier for the given type name.</td>
<td></td>
</tr>
<tr>
<td><code>TypeList findTypes()</code></td>
<td>Gets all existing Dex node and edge type identifiers.</td>
<td></td>
</tr>
<tr>
<td><code>Attribute getAttribute(int attr)</code></td>
<td>Gets information about the given attribute.</td>
<td></td>
</tr>
<tr>
<td><code>Value getAttribute(long oid, int attr)</code></td>
<td>Gets the Value for the given attribute and OID.</td>
<td></td>
</tr>
<tr>
<td><code>void getAttribute(long oid, int attr, Value v)</code></td>
<td>Gets the Value for the given attribute and OID.</td>
<td></td>
</tr>
<tr>
<td>long <code>getAttributeIntervalCount(int attr, Value lower, boolean includeLower, Value higher, boolean includeHigher)</code></td>
<td>Gets how many objects have a value into the given range for the given attribute.</td>
<td></td>
</tr>
<tr>
<td><code>AttributeList getAttributes(long oid)</code></td>
<td>Gets all Dex attribute identifiers with a non-NULL value for the given Dex OID.</td>
<td></td>
</tr>
<tr>
<td><code>AttributeStatistics getAttributeStatistics(int attr, boolean basic)</code></td>
<td>Gets statistics from the given attribute.</td>
<td></td>
</tr>
<tr>
<td><code>TextStream getAttributeText(long oid, int attr)</code></td>
<td>Gets the read-only TextStream for the given text attribute and OID.</td>
<td></td>
</tr>
<tr>
<td><code>EdgeData getEdgeData(long edge)</code></td>
<td>Gets information about an edge.</td>
<td></td>
</tr>
<tr>
<td>long <code>getEdgePeer(long edge, long node)</code></td>
<td>Gets the other end for the given edge.</td>
<td></td>
</tr>
<tr>
<td>int  <code>getObjectType(long oid)</code></td>
<td>Gets the Dex node or edge type identifier for the given OID.</td>
<td></td>
</tr>
<tr>
<td><code>Type getType(int type)</code></td>
<td>Gets information about the given type.</td>
<td></td>
</tr>
<tr>
<td><code>Values getValues(int attr)</code></td>
<td>Gets the Value collection for the given attribute.</td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><code>heads(Objects edges)</code></td>
<td>Gets all the heads from the given edges collection.</td>
<td></td>
</tr>
<tr>
<td><code>indexAttribute(int attr, AttributeKind kind)</code></td>
<td>Updates the index of the given attribute.</td>
<td></td>
</tr>
<tr>
<td><code>neighbors(long oid, int etype, EdgesDirection dir)</code></td>
<td>Selects all neighbor nodes from or to the given node OID and for the given edge type.</td>
<td></td>
</tr>
<tr>
<td><code>neighbors(Objects objs, int etype, EdgesDirection dir)</code></td>
<td>Selects all neighbor nodes from or to each of the node OID in the given collection and for the given edge type.</td>
<td></td>
</tr>
<tr>
<td><code>newAttribute(int type, String name, DataType dt, AttributeKind kind)</code></td>
<td>Creates a new attribute.</td>
<td></td>
</tr>
<tr>
<td><code>newAttribute(int type, String name, DataType dt, AttributeKind kind, Value defaultValue)</code></td>
<td>Creates a new attribute with a default value.</td>
<td></td>
</tr>
<tr>
<td><code>newEdge(int type, int tailAttr, Value tailV, int headAttr, Value headV)</code></td>
<td>Creates a new edge instance.</td>
<td></td>
</tr>
<tr>
<td><code>newEdge(int type, long tail, long head)</code></td>
<td>Creates a new edge instance.</td>
<td></td>
</tr>
<tr>
<td><code>newEdgeType(String name, boolean directed, boolean neighbors)</code></td>
<td>Creates a new edge type.</td>
<td></td>
</tr>
<tr>
<td><code>newNode(int type)</code></td>
<td>Creates a new node instance.</td>
<td></td>
</tr>
<tr>
<td><code>newNodeType(String name)</code></td>
<td>Creates a new node type.</td>
<td></td>
</tr>
<tr>
<td><code>newRestrictedEdgeType(String name, int tail, int head, boolean neighbors)</code></td>
<td>Creates a new restricted edge type.</td>
<td></td>
</tr>
<tr>
<td><code>newSessionAttribute(int type, DataType dt, AttributeKind kind)</code></td>
<td>Creates a new Session attribute.</td>
<td></td>
</tr>
<tr>
<td><code>newSessionAttribute(int type, DataType dt, AttributeKind kind, Value defaultValue)</code></td>
<td>Creates a new Session attribute with a default value.</td>
<td></td>
</tr>
<tr>
<td><code>removeAttribute(int attr)</code></td>
<td>Removes the given attribute.</td>
<td></td>
</tr>
<tr>
<td><code>removeType(int type)</code></td>
<td>Removes the given type.</td>
<td></td>
</tr>
<tr>
<td><code>select(int type)</code></td>
<td>Selects all OIDs belonging to the given type.</td>
<td></td>
</tr>
<tr>
<td><code>select(int attr, Condition cond, Value v)</code></td>
<td>Selects all OIDs satisfying the given condition for the given attribute.</td>
<td></td>
</tr>
<tr>
<td><code>select(int attr, Condition cond, Value lower, Value higher)</code></td>
<td>Selects all OIDs satisfying the given condition for the given attribute.</td>
<td></td>
</tr>
</tbody>
</table>
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

Methods

setAttributeText

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

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

dumpData

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

findTypes

public `TypeList` `findTypes` ()
Gets all existing Dex node and edge type identifiers.

**Returns:**
Dex node and edge type identifier list.

---

### setAttribute

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

Sets the Value for the given attribute and OID.

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

---

### select

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

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

**Parameters:**
- **attr** - [in] Dex attribute identifier.
- **cond** - [in] Condition to be satisfied.
- **v** - [in] Value to be satisfied.

**Returns:**
Objects instance.

---

### 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 AttributeKind::Indexed or AttributeKind::Unique attributes.

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

**Parameters:**
- **attr** - [in] Dex 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
tailsAndHeads(Objects edges,
        Objects tails,
        Objects heads)
```

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

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

Returns:
The number of edges.

dumpStorage

```java
dumpStorage(String file)
```

Dumps internal storage 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

neighbors

```java
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] Dex node OID collection.
- `etype` - [in] Dex edge type identifier.
- `dir` - [in] Direction.

**Returns:**
Objects instance.

---

**export**

```java
public void export(String file,
                   ExportType type,
                   ExportManager e)
```

Exports the Graph.

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

---

**getAttributes**

```java
public AttributeList getAttributes(long oid)
```

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

**Parameters:**
- `oid` - [in] Dex OID.

**Returns:**
Dex attribute identifier list.

---

**getAttributeStatistics**

```java
public AttributeStatistics getAttributeStatistics(int attr,
                                                   boolean basic)
```

Gets statistics from the given attribute.

**Parameters:**
- `attr` - [in] Dex 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 instace.
**newNode**

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

Creates a new node instance.

**Parameters:**

type - [in] Dex 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] Dex OID.
attr - [in] Dex 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.

---

**select**

```java
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 Condition::Between operation, thus it has two Value arguments.

**Parameters:**

attr - [in] Dex attribute identifier.
cond - [in] Condition to be satisfied. It must be Condition::Between.
lower - [in] Lower-bound Value to be satisfied.
higher - [in] Higher-bound Value to be satisfied.

**Returns:**

Objects instance.
findEdgeTypes

```java
public TypeList findEdgeTypes()
```

Gets all existing Dex edge type identifiers.

Returns:
  Dex edge type identifier list.

indexAttribute

```java
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 AttributeKind::Basic and the new one is AttributeKind::Indexed or AttributeKind::Unique.

Parameters:
  attr - [in] Dex attribute identifier.
  kind - [in] Attribute kind.

getTipo

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

Gets information about the given type.

Parameters:
  type - [in] Dex type identifier.

Returns:
  The Type for the given Dex type identifier.

setAttributeDefaultValue

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

findAttribute

```java
public int findAttribute(int type, String name)
```
Gets the Dex attribute identifier for the given type identifier and attribute name.

**Parameters:**
- `type` - [in] Dex type identifier.
- `name` - [in] Unique attribute name.

**Returns:**
The Dex attribute identifier for the given type and attribute name or `Attribute::InvalidAttribute` if there is no attribute with the given name for the given type.

---

newAttribute

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

Creates a new attribute.

**Parameters:**
- `type` - [in] Dex 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 Dex attribute identifier.

---

edges

```java
public Objects 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
- `tail` - [in] Tail node identifier.
- `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] Dex type identifier.
Returns:
Objects instance.

**findNodeTypes**

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

Gets all existing Dex node type identifiers.

**Returns:**
Dex 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] Dex OID.
- `attr` - [in] Dex 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] Dex attribute identifier.

**backup**

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

Throws `FileNotFoundException`, `RuntimeException`

Dumps all the data to a backup file.

See Dex::Restore.

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

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

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] Dex 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 Dex attribute identifier.

findAttributes

public AttributeList findAttributes(int type)

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

Parameters:
   type - [in] Dex type identifier.

Returns:
   Dex attribute identifier list.

countNodes

public long countNodes()

Gets the number of nodes.

Returns:
   The number of nodes.

getEdgeData

public EdgeData getEdgeData(long edge)

Gets information about an edge.

Parameters:
   edge - [in] Dex edge identifier.
Returns:
An EdgeData instance.

neighbors

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] Dex node OID.
etype - [in] Dex edge type identifier.
dir - [in] Direction.

Returns:
Objects instance.

eplode

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:
objs - [in] Dex node OID collection.
etype - [in] Dex 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 Dex type identifier.

newSessionAttribute

public int newSessionAttribute(int type,
                               DataTypes dt,
                               AttributeKind 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] Dex node or edge type identifier.
- dt - [in] Data type for the new attribute.
- kind - [in] Attribute kind.

**Returns:**
Unique Dex attribute identifier.

---

**tails**

```java
public Objects tails(Objects edges)
```

Gets all the tails from the given edges collection.

**Parameters:**

**Returns:**
The tails collection.

---

**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] Dex OID to be removed.

---

**newEdgeType**

```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 Dex type identifier.
heads
public Objects heads(Objects edges)

Gets all the heads from the given edges collection.

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

Returns:
The heads collection.

findEdge
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, Objects::InvalidOID will be returned.

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

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

explode
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] Dex node OID.
  etype - [in] Dex edge type identifier.
  dir - [in] Direction.

Returns:
Objects instance.

drop
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:
**newAttribute**

```java
public int newAttribute(int type,
                      String name,
                      DataType dt,
                      AttributeKind kind,
                      Value defaultValue)
```

Creates a new attribute with a default value.

**Parameters:**
- `type` - [in] Dex 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 Dex attribute identifier.

**getObjectType**

```java
public int getObjectType(long oid)
```

Gets the Dex node or edge type identifier for the given OID.

**Parameters:**
- `oid` - [in] Dex OID.

**Returns:**
Dex node or edge type identifier.

**getAttribute**

```java
public Attribute getAttribute(int attr)
```

Gets information about the given attribute.

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

**Returns:**
The Attribute for the given Dex attribute identifier.

**newEdge**

```java
public long newEdge(int type,
                    int tailAttr,
                    Value tailV,
                    int headAttr,
                    Value headV)
```

Page 143 of 286
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] Dex type identifier.
- `headAttr` - [in] Dex attribute identifier.
- `headV` - [in] Value.

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

---

**getValues**

```java
public Values getValues(int attr)
```

Gets the Value collection for the given attribute.

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

---

**getEdgePeer**

```java
public long getEdgePeer(long edge, long node)
```

Gets the other end for the given edge.

**Parameters:**
- `edge` - [in] Dex edge identifier.
- `node` - [in] Dex node identifier. It must be one of the ends of the edge.

**Returns:**
The other end of the edge.

---

**findObject**

```java
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, Objects::InvalidOID will be returned.

**Parameters:**
- `attr` - [in] Dex attribute identifier.
- `v` - [in] Value.

**Returns:**
Dex OID or Objects::InvalidOID.
newEdge
public long newEdge(int type,
   long tail,
   long head)

Creates a new edge instance.

Parameters:
   type - [in] Dex type identifier.
   tail - [in] Source Dex OID.
   head - [in] Target Dex OID.

Returns:
   Unique OID of the new edge instance.

findType
public int findType(String name)

Gets the Dex type identifier for the given type name.

Parameters:
   name - [in] Unique type name.

Returns:
   The Dex type identifier for the given type name or 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] Dex OID.
   attr - [in] Dex attribute identifier.
   v - [in/out] Value for the given attribute and for the given OID.

removeType
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] Dex type identifier.
newRestrictedEdgeType

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 Dex node type identifier.
    head - [in] Head Dex node type identifier.
    neighbors - [in] If TRUE, this indexes neighbor nodes, otherwise not.

Returns:
    Unique Dex 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>public</th>
<th>GraphExport()</th>
</tr>
</thead>
</table>

Creates a new GraphExport instance.

Method Summary

<table>
<thead>
<tr>
<th>String</th>
<th>getLabel()</th>
</tr>
</thead>
</table>

Gets the graph label.

<table>
<thead>
<tr>
<th>void</th>
<th>setDefaults()</th>
</tr>
</thead>
</table>

Sets to default values.

<table>
<thead>
<tr>
<th>void</th>
<th>setLabel(String l)</th>
</tr>
</thead>
</table>

Sets the graph label.

Methods inherited from class java.lang.Object

close, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait
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.
public class Int32List
extends Object
implements Iterable

Dex 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>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
<td>Int32List(Collection col)</td>
<td>Creates a new instance from an integer collection.</td>
</tr>
<tr>
<td>public</td>
<td>Int32List()</td>
<td>Constructor.</td>
</tr>
<tr>
<td>public</td>
<td>Int32List(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>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>add(int value)</td>
<td>Adds an 32-bit signed integer at the end of the list.</td>
</tr>
<tr>
<td>void</td>
<td>clear()</td>
<td>Clears the list.</td>
</tr>
<tr>
<td>int</td>
<td>count()</td>
<td>Number of elements in the list.</td>
</tr>
<tr>
<td>Int32ListIterator</td>
<td>iterator()</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**

*public Int32List(Collection col)*

Creates a new instance from an integer collection.

**Parameters:**

- `col` - Collection to initialize the instance.

---

**Int32List**

*public Int32List()*

Constructor.

This creates an empty list.

---

**Int32List**

*public Int32List(int[] list)*

Creates a new instance from an integer array.

**Parameters:**

- `list` - Integer array to initialize the instance.

Methods

**add**

*public void add(int value)*

 Adds an 32-bit signed integer at the end of the list.

**Parameters:**

- `value` - null

**clear**

*public void clear()*

Clears the list.

**iterator**

*public Int32ListIterator iterator()*
Gets a new Int32ListIterator.

**Returns:**
Int32ListIterator instance.

count

```java
public int count()
```

Number of elements in the list.

**Returns:**
Number of elements in the list.
com.sparsity.dex.gdb
Class Int32ListIterator

java.lang.Object
  +-com.sparsity.dex.gdb.Int32ListIterator

All Implemented Interfaces:
  Iterator

public class Int32ListIterator
extends Object
implements Iterator

Int32List iterator class.

Iterator to traverse all the integer into a Int32List instance.

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

Method Summary

<table>
<thead>
<tr>
<th>boolean</th>
<th>hasNext()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Integer</th>
<th>next()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Integer</th>
<th>nextInt32()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| void | remove() |
|      |         |
|      | 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

nextInt32

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.dex.gdb
Class LogLevel

java.lang.Object
   +---java.lang.Enum
        +---com.sparsity.dex.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

Field Summary

<table>
<thead>
<tr>
<th>public static final</th>
<th>Config</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config log level.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public static final</th>
<th>Debug</th>
</tr>
</thead>
<tbody>
<tr>
<td>Debug log level.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public static final</th>
<th>Fine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fine log level.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public static final</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>Info log level.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public static final</th>
<th>Off</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disable log.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public static final</th>
<th>Severe</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severe log level.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public static final</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Warning log level.</td>
<td></td>
</tr>
</tbody>
</table>

Method Summary

| static LogLevel        | logLevel.valueOf(String name) |

| static LogLevel[]      | logLevel.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

#### Off

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>

#### Severe

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static final com.sparsity.dex.gdb.LogLevel Severe</td>
<td>Severe log level.</td>
</tr>
<tr>
<td></td>
<td>This is the lower log level, just errors are shown.</td>
</tr>
</tbody>
</table>

#### Warning

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static final com.sparsity.dex.gdb.LogLevel Warning</td>
<td>Warning log level.</td>
</tr>
<tr>
<td></td>
<td>Errors and warnings are shown.</td>
</tr>
</tbody>
</table>

#### Info

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static final com.sparsity.dex.gdb.LogLevel Info</td>
<td>Info log level.</td>
</tr>
<tr>
<td></td>
<td>Errors, warnings and information messages are shown.</td>
</tr>
</tbody>
</table>

#### Config

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static final com.sparsity.dex.gdb.LogLevel Config</td>
<td>Config log level.</td>
</tr>
<tr>
<td></td>
<td>Errors, warnings, information messages and configuration details of the different components are shown.</td>
</tr>
</tbody>
</table>

#### Fine

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
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.dex.gdb.LogLevel Debug

Debug log level.

This is for Dex 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)
```
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>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public NodeExport()</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>java.awt.Color</td>
<td>getColor()</td>
<td>Gets the color of the node.</td>
</tr>
<tr>
<td>int</td>
<td>getColorRGB()</td>
<td>Gets the node color.</td>
</tr>
<tr>
<td>int</td>
<td>getFontSize()</td>
<td>Gets the node label font size.</td>
</tr>
<tr>
<td>int</td>
<td>getHeight()</td>
<td>Gets the node height.</td>
</tr>
<tr>
<td>String</td>
<td>getLabel()</td>
<td>Gets the node label.</td>
</tr>
</tbody>
</table>
**java.awt.Color**

*getLabelColor()*

Gets the color of the label.

*getLabelColorRGB()*

Gets the node label color.

**NodeShape**

*getShape()*

Gets the node shape.

*getWidth()*

Gets the node width.

*isFit()*

Gets whether the node size is fitted to the label or not.

**void**

setColor(java.awt.Color c)

Sets the color of the node.

setColorRGB(int c)

Sets the node color.

setDefaults()

Sets to default values.

setFit(boolean f)

Sets the node fit property.

setFontSize(int s)

Sets the node label font size.

setHeight(int h)

Sets the node height.

setLabel(String l)

Sets the node label.

setLabelColor(java.awt.Color c)

Sets the color of the label.

setLabelColorRGB(int c)

Sets the node label color.

setShape(NodeShape s)

Sets the node shape.

setWidth(int w)

Gets the node width.

---

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

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

g getFont Size

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

public int getLabelColorRGB()

Gets the node label color.

Returns:
The node label color.

getWidth

public int getWidth()

Gets the node width.

Returns:
The node width in pixels.

getLabelColor

public java.awt.Color getLabelColor()

Gets the color of the label.

setColor

public void setColor(java.awt.Color c)

Sets the color of the node.

Parameters:
  c - New value.

setLabel

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.
public final class NodeShape
extends Enum

Node shape.

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

Field Summary

<table>
<thead>
<tr>
<th>public static final</th>
<th>Box</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Box shape.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public static final</th>
<th>Round</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Round shape.</td>
</tr>
</tbody>
</table>

Method Summary

| static NodeShape    | valueOf(String name) |

| static NodeShape[]  | values() |

Methods inherited from class java.lang.Enum

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

Methods inherited from class java.lang.Object

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

Methods inherited from interface java.lang.Comparable

cache

cacheTo

Fields
Box

public static final com.sparsity.dex.gdb.NodeShape Box

    Box shape.

Round

public static final com.sparsity.dex.gdb.NodeShape Round

    Round shape.

Methods

values

public static NodeShape[] values()

valueOf

public static NodeShape valueOf(String name)
com.sparsity.dex.gdb

Class Objects

java.lang.Object
   +com.sparsity.dex.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 Dex 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>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Adds an element into the collection.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>add(Long e)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Adds the specified element to this set if it is not already present (optional operation).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>addAll(Collection clctn)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Adds all of the elements in the specified collection to this set if they're not already present (optional operation).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>long</th>
<th>any()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets an element from the collection.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>clear()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Clears the collection removing all its elements.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>close()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Closes the Objects instance.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>static Objects</strong>&lt;br&gt;<strong>combineDifference</strong>(Objects objs1, Objects objs2)</td>
<td>Creates a new Objects instance which is the difference of the two given.</td>
</tr>
<tr>
<td><strong>static Objects</strong>&lt;br&gt;<strong>combineIntersection</strong>(Objects objs1, Objects objs2)</td>
<td>Creates a new Objects instance which is the intersection of the two given.</td>
</tr>
<tr>
<td><strong>static Objects</strong>&lt;br&gt;<strong>combineUnion</strong>(Objects objs1, Objects objs2)</td>
<td>Creates a new Objects instance which is the union of the two given.</td>
</tr>
<tr>
<td><strong>boolean</strong>&lt;br&gt;<strong>contains</strong>(Object o)</td>
<td>Returns true if this collections contains the specified element or Objects.</td>
</tr>
<tr>
<td><strong>boolean</strong>&lt;br&gt;<strong>contains</strong>(Objects objs)</td>
<td>Checks if this Objects contains the other one.</td>
</tr>
<tr>
<td><strong>boolean</strong>&lt;br&gt;<strong>containsAll</strong>(Collection clctn)</td>
<td>Returns true if this set contains all of the elements of the specified collection.</td>
</tr>
<tr>
<td><strong>Objects</strong>&lt;br&gt;<strong>copy</strong>()</td>
<td>Creates a new Objects instance as a copy of the given one.</td>
</tr>
<tr>
<td><strong>long</strong>&lt;br&gt;<strong>copy</strong>(Objects objs)</td>
<td>Performs the copy operation.</td>
</tr>
<tr>
<td><strong>long</strong>&lt;br&gt;<strong>count</strong>()</td>
<td>Gets the number of elements into the collection.</td>
</tr>
<tr>
<td><strong>long</strong>&lt;br&gt;<strong>difference</strong>(Objects objs)</td>
<td>Performs the difference operation.</td>
</tr>
<tr>
<td><strong>boolean</strong>&lt;br&gt;<strong>equals</strong>(Object o)</td>
<td>Returns true if the collection is equal to the object.</td>
</tr>
<tr>
<td><strong>boolean</strong>&lt;br&gt;<strong>equals</strong>(Objects objs)</td>
<td>Checks if the given Objects contains the same information.</td>
</tr>
<tr>
<td><strong>boolean</strong>&lt;br&gt;<strong>exists</strong>(long e)</td>
<td>Gets if the given element exists into the collection.</td>
</tr>
<tr>
<td><strong>long</strong>&lt;br&gt;<strong>intersection</strong>(Objects objs)</td>
<td>Performs the intersection operation.</td>
</tr>
<tr>
<td><strong>boolean</strong>&lt;br&gt;<strong>isClosed</strong>()</td>
<td>Gets if Objects instance has been closed or not.</td>
</tr>
<tr>
<td><strong>boolean</strong>&lt;br&gt;<strong>isEmpty</strong>()</td>
<td>Returns true if this Objects contains no elements.</td>
</tr>
<tr>
<td><strong>ObjectsIterator</strong>&lt;br&gt;<strong>iterator</strong>()</td>
<td>Gets an ObjectsIterator.</td>
</tr>
<tr>
<td><strong>ObjectsIterator</strong>&lt;br&gt;<strong>iteratorFromElement</strong>(long e)</td>
<td>Gets an ObjectsIterator starting from the given element.</td>
</tr>
<tr>
<td><strong>ObjectsIterator</strong>&lt;br&gt;<strong>iteratorFromIndex</strong>(long index)</td>
<td>Gets an ObjectsIterator skipping index elements.</td>
</tr>
<tr>
<td><strong>boolean</strong>&lt;br&gt;<strong>remove</strong>(long e)</td>
<td>Removes an element from the collection.</td>
</tr>
</tbody>
</table>
boolean remove(Object o)
   Removes the specified element from this set if it is present (optional operation).

boolean removeAll(Collection clctn)
   Removes from this set all of its elements that are contained in the specified collection (optional operation).

boolean retainAll(Collection clctn)
   Retains only the elements in this set that are contained in the specified collection (optional operation).

Objects sample(Object exclude, long samples)
   Creates a new Objects instance which is a sample of the calling one.

int size()
   Gets the size of the collection.

Object[] toArray()
   Returns an array containing all of the object identifiers in this set.

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.

long union(Objects objs)
   Performs the union operation.

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

Methods inherited from interface java.util.Set
   add, addAll, clear, contains, containsAll, equals, hashCode, isEmpty, iterator, remove, removeAll, retainAll, size, toArray, toObject

Methods inherited from interface java.util.Collection
   add, addAll, clear, contains, containsAll, equals, hashCode, isEmpty, iterator, remove, removeAll, retainAll, size, toArray, toObject

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:
   objs - [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:
   objs1 - [in] Objects instance.
   objs2 - [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

```java
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

```java
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

```java
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

```java
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**

**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**

**public void close()**

Closes the Objects instance.

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

**isEmpty**

**public boolean isEmpty()**

Returns true if this Objects contains no elements.

**Returns:**
true if the collection contains no elements.

**contains**

**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**

**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**

```java
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**

```java
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**

```java
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**

```java
public Object[] toArray()
```

Returns an array containing all of the object identifiers in this set.

Obeys the general contract of the Collection.toArray method.
**toArray**

```java
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**

```java
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**

```java
public boolean exists(long e)
```

Gets if the given element exists into the collection.

**Parameters:**

**Returns:**
- TRUE if the element exists into the collection, FALSE otherwise.

---

**combineDifference**

```java
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 <= 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

public Objects copy()

    Creates a new Objects instance as a copy of the given one.

    Returns:
    The new Objects instance.
com.sparsity.dex.gdb
Class ObjectsIterator

java.lang.Object
   +--com.sparsity.dex.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**

```java
public long nextObject()
```

Gets the next element.

**hasNext**

```java
public boolean hasNext()
```

Gets if there are more elements to traverse.

**Returns:**

TRUE if there are more elements to traverse, FALSE otherwise.

**remove**

```java
public void remove()
```

Operation not supported.

**next**

```java
public Long next()
```

See nextObject().

**isClosed**

```java
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**

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

Closes the ObjectsIterator instance.

It must be called to ensure the integrity of all data.
public final class ObjectType
extends Enum

Object type enumeration.

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

Field Summary

<table>
<thead>
<tr>
<th>public static final</th>
<th>Edge</th>
<th>Edge object type.</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static final</td>
<td>Node</td>
<td>Node object type.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>static ObjectType</th>
<th>valueOf(String name)</th>
</tr>
</thead>
<tbody>
<tr>
<td>static ObjectType[]</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
Node
public static final com.sparsity.dex.gdb.ObjectType Node
    
    Node object type.

Edge
public static final com.sparsity.dex.gdb.ObjectType Edge

    Edge object type.

Methods

values
public static ObjectType[] values()

valueOf
public static ObjectType valueOf(String name)
public class OIDList
extends Object
implements Iterable

Dex object identifier list.

It stores a Dex object identifier list.

Use OIDListIterator 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>OIDList(int N)</td>
<td>Constructor.</td>
</tr>
<tr>
<td>OIDList(long[] list)</td>
<td>Creates a new instance from a long array.</td>
</tr>
<tr>
<td>OIDList(Collection col)</td>
<td>Creates a new instance from a long collection.</td>
</tr>
<tr>
<td>OIDList()</td>
<td>Constructor.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void add(long attr)</td>
<td>Adds a Dex object 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>OIDListIterator iterator()</td>
<td>Gets a new OIDListIterator.</td>
</tr>
<tr>
<td>void set(int pos, long oid)</td>
<td>Sets a Dex object identifier at the specified position of the list.</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

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 Dex object identifier at the end of the list.

**Parameters:**
attr - [in] Dex object identifier.

### clear

```java
public void clear()
```

Clears the list.

### set

```java
public void set(int pos, long oid)
```

Sets a Dex object identifier at the specified position of the list.

**Parameters:**
pos - [in] List position [0..Count()-1].
oid - [in] Dex 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.dex.gdb
Class OIDListIterator

java.lang.Object
    +-com.sparsity.dex.gdb.OIDListIterator

All Implemented Interfaces:
    Iterator

public class OIDListIterator extends Object implements Iterator

OIDList iterator class.
Iterator to traverse all the Dex 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>
<tbody>
<tr>
<td></td>
<td>Gets if there are more elements.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Long</th>
<th>next()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>See nextOID().</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>long</th>
<th>nextOID()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the next element.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>remove()</th>
</tr>
</thead>
<tbody>
<tr>
<td></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 Long next()

See nextOId().

close
public long close()

Gets the next element.
com.sparsity.dex.gdb
Class Order

downward
java.lang.Object
   ↓java.lang.Enum
      ↓com.sparsity.dex.gdb.Order

All Implemented Interfaces:
   Serializable, Comparable

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></td>
<td>From lower to higher.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public static final Descendent</th>
<th>Descendent</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>From higher to lower.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>static Order</th>
<th>static Order()</th>
</tr>
</thead>
<tbody>
<tr>
<td>valueOf(String name)</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.dex.gdb.Order Ascendent

   From lower to higher.

Descendent

public static final com.sparsity.dex.gdb.Order Descendent

   From higher to lower.

Methods

values

public static Order[] values()

valueOf

public static Order valueOf(String name)
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></td>
<td>Gets platform data and statistics.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

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

Methods

getStatistics

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>public</th>
<th>PlatformStatistics()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Creates a new instance setting all values to 0.</td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>long</th>
<th>getAvailableMem()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets available (free) memory size in Bytes.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>getNumCPUs()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the number of CPUs.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>long</th>
<th>getRealTime()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets time in microseconds (since epoch).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>long</th>
<th>getSystemTime()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets CPU system time.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>long</th>
<th>getTotalMem()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets physical memory size in Bytes.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>long</th>
<th>getUserTime()</th>
</tr>
</thead>
<tbody>
<tr>
<td></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, wait

### Constructors

**PlatformStatistics**

<table>
<thead>
<tr>
<th>public</th>
<th>PlatformStatistics()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Creates a new instance setting all values to 0.

## Methods

### getRealTime

```java
public long getRealTime()
```

Gets time in microseconds (since epoch).

**Returns:**
Time in microseconds (since epoch).

### getAvailableMem

```java
public long getAvailableMem()
```

Gets available (free) memory size in Bytes.

**Returns:**
Available (free) memory size in Bytes.

### getTotalMem

```java
public long getTotalMem()
```

Gets physical memory size in Bytes.

**Returns:**
Physical memory size in Bytes.

### getSystemTime

```java
public long getSystemTime()
```

Gets CPU system time.

**Returns:**
CPU system time.

### getUserTime

```java
public long getUserTime()
```

Gets CPU user time.

**Returns:**
CPU user time.

getNumCPUs

public int getNumCPUs()

Gets the number of CPUs.

**Returns:**

The number of CPUs.
### com.sparsity.dex.gdb

**Session**

```java
java.lang.Object
   +-com.sparsity.dex.gdb.Session
```

**All Implemented Interfaces:**
- Closeable

---

**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 DEX 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>void</th>
<th>begin()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Begins a transaction.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>void</th>
<th>commit()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Commits a transaction.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Graph</th>
<th>getGraph()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the Graph instance.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>isClosed()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets if Session instance has been closed or not.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Objects</th>
<th>newObjects()</th>
</tr>
</thead>
<tbody>
<tr>
<td></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.
com.sparsity.dex.gdb
Class StringList

java.lang.Object
   +-com.sparsity.dex.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

<table>
<thead>
<tr>
<th>public</th>
<th>StringList(Collection col)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Creates a new instance from an string collection.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public</th>
<th>StringList()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Constructor.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public</th>
<th>StringList(String[] list)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Creates a new instance from an string array.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>void</th>
<th>add(String str)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Adds a String 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>StringListIterator</th>
<th>iterator()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets a new StringListIterator.</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

**StringList**

```java
public StringList(Collection col)
```

Creates a new instance from an string collection.

**Parameters:**
- `col` - Collection to initialize the instance.

**StringList**

```java
public StringList()
```

Constructor.

This creates an empty list.

**StringList**

```java
public StringList(String[] list)
```

Creates a new instance from an string array.

**Parameters:**
- `list` - String array to initialize the instance.

Methods

**clear**

```java
public void clear()
```

Clears the list.

**iterator**

```java
public StringListIterator iterator()
```

Gets a new StringListIterator.

**Returns:**
- StringListIterator instance.

**count**

```java
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.dex.gdb
Class StringListIterator

java.lang.Object
    +-com.sparsity.dex.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>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean hasNext()</td>
<td>Gets if there are more elements.</td>
</tr>
<tr>
<td>String next()</td>
<td>See nextString().</td>
</tr>
<tr>
<td>String nextString()</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.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 DEX User Manual for more details on this.

Author: Sparsity Technologies http://www.sparsity-technologies.com
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait</td>
<td></td>
</tr>
<tr>
<td><strong>Methods inherited from interface</strong> java.io.Closeable</td>
<td>close</td>
</tr>
</tbody>
</table>

### Constructors

**TextStream**

```java
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**

```java
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**

```java
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**

```java
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.
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>Field Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static</td>
<td>GlobalType</td>
</tr>
<tr>
<td></td>
<td>Global type identifier constant.</td>
</tr>
<tr>
<td>public static</td>
<td>InvalidType</td>
</tr>
<tr>
<td></td>
<td>Invalid type identifier constant.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Method Type</th>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>getAreNeighborsIndexed()</td>
<td>Gets if this is an edge type with neighbors index.</td>
</tr>
<tr>
<td>int</td>
<td>getId()</td>
<td>Gets the Dex type identifier.</td>
</tr>
<tr>
<td>boolean</td>
<td>getIsDirected()</td>
<td>Gets if this is a directed edge type.</td>
</tr>
<tr>
<td>boolean</td>
<td>getIsRestricted()</td>
<td>Gets if this is a restricted edge type.</td>
</tr>
<tr>
<td>String</td>
<td>getName()</td>
<td>Gets the unique type name.</td>
</tr>
<tr>
<td>long</td>
<td>getNumObjects()</td>
<td>Gets the number of objects belonging to the type.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>getObjectType()</td>
<td>Gets the object type.</td>
</tr>
<tr>
<td>int</td>
<td>getRestrictedFrom()</td>
<td>Gets the tail or source type identifier for restricted edge types.</td>
</tr>
<tr>
<td>int</td>
<td>getRestrictedTo()</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, 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, 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 Dex type identifier.

Returns:
The Dex 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.dex.gdb

Class TypeList

derived from java.lang.Object

All Implemented Interfaces:
    Iterable

public class TypeList
extends Object
implements Iterable

Dex type identifier list.
It stores a Dex 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 Dex 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

Methods inherited from interface java.lang.Iterable

iterator
Constructors

**TypeList**

```java
public TypeList(int[] list)
```

Creates a new instance from an integer array.

**Parameters:**
- `list` - Integer array to initialize the instance.

```java
public TypeList(Collection col)
```

Creates a new instance from an integer collection.

**Parameters:**
- `col` - Collection to initialize the instance.

```java
public TypeList()
```

Constructor.

This creates an empty list.

Methods

**add**

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

Adds a Dex type identifier at the end of the list.

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

**clear**

```java
public void clear()
```

Clears the list.

**iterator**

```java
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.
com.sparsity.dex.gdb
Class TypeListIterator

java.lang.Object
  +-com.sparsity.dex.gdb.TypeListIterator

All Implemented Interfaces:
  Iterator

public class TypeListIterator
extends Object
implements Iterator

TypeList iterator class.

Iterator to traverse all the Dex 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>boolean hasNext()</td>
<td>Gets if there are more elements.</td>
</tr>
<tr>
<td>Integer next()</td>
<td>See nextType().</td>
</tr>
<tr>
<td>int nextType()</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.util.Iterator

hasNext, next, remove

Methods

hasNext

public boolean hasNext()

  Gets if there are more elements.
store

public void remove()

Operation not supported.

next

public Integer next()

See nextType().

nextType

public int nextType()

Gets the next element.
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>Type</th>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static</td>
<td>MaxLengthString</td>
<td>Maximum number of characters allowed for a String.</td>
</tr>
</tbody>
</table>

Constructor Summary

<table>
<thead>
<tr>
<th>Type</th>
<th>Constructor Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
<td>Value(Value v)</td>
<td>Copy constructor.</td>
</tr>
<tr>
<td>public</td>
<td>Value()</td>
<td>Creates a new instance.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Type</th>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>compare(Value v)</td>
<td>Compares with the given Value.</td>
</tr>
<tr>
<td>int</td>
<td>compareTo(Object v)</td>
<td>See compare().</td>
</tr>
<tr>
<td>int</td>
<td>compareTo(Value v)</td>
<td>See compare().</td>
</tr>
<tr>
<td>boolean</td>
<td>equals(Object other)</td>
<td></td>
</tr>
<tr>
<td>boolean</td>
<td>equals(Value v)</td>
<td>Compares with the given Value.</td>
</tr>
<tr>
<td>boolean</td>
<td>getBoolean()</td>
<td>Gets Boolean Value.</td>
</tr>
<tr>
<td>DataType</td>
<td>getDataType()</td>
<td>Gets the DataType.</td>
</tr>
<tr>
<td>double</td>
<td>getDouble()</td>
<td>Gets Double Value.</td>
</tr>
<tr>
<td>Type</td>
<td>Method Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>------------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>int</td>
<td><code>getInteger()</code></td>
<td>Gets Integer Value.</td>
</tr>
<tr>
<td>long</td>
<td><code>getLong()</code></td>
<td>Gets Long Value.</td>
</tr>
<tr>
<td>long</td>
<td><code>getOID()</code></td>
<td>Gets OID Value.</td>
</tr>
<tr>
<td>String</td>
<td><code>getString()</code></td>
<td>Gets String Value.</td>
</tr>
<tr>
<td>long</td>
<td><code>getTimestamp()</code></td>
<td>Gets Timestamp Value.</td>
</tr>
<tr>
<td>Calendar</td>
<td><code>getTimestampAsCalendar()</code></td>
<td>Gets the Value as a Calendar instance.</td>
</tr>
<tr>
<td>Date</td>
<td><code>getTimestampAsDate()</code></td>
<td>Gets the Value as a Date instance.</td>
</tr>
<tr>
<td>int</td>
<td><code>hashCode()</code></td>
<td></td>
</tr>
<tr>
<td>boolean</td>
<td><code>isNull()</code></td>
<td>Gets if this is a NULL Value.</td>
</tr>
<tr>
<td>Value</td>
<td><code>set(Value v)</code></td>
<td>Sets the Value.</td>
</tr>
<tr>
<td>Value</td>
<td><code>setBoolean(boolean v)</code></td>
<td>Sets the Value.</td>
</tr>
<tr>
<td>void</td>
<td><code>setBooleanVoid(boolean v)</code></td>
<td>Sets the Value.</td>
</tr>
<tr>
<td>Value</td>
<td><code>setDouble(double v)</code></td>
<td>Sets the Value.</td>
</tr>
<tr>
<td>void</td>
<td><code>setDoubleVoid(double v)</code></td>
<td>Sets the Value.</td>
</tr>
<tr>
<td>Value</td>
<td><code>setInteger(int v)</code></td>
<td>Sets the Value.</td>
</tr>
<tr>
<td>void</td>
<td><code>setIntegerVoid(int v)</code></td>
<td>Sets the Value.</td>
</tr>
<tr>
<td>Value</td>
<td><code>setLong(long v)</code></td>
<td>Sets the Value.</td>
</tr>
<tr>
<td>void</td>
<td><code>setLongVoid(long v)</code></td>
<td>Sets the Value.</td>
</tr>
<tr>
<td>Value</td>
<td><code>setNull()</code></td>
<td>Sets the Value to NULL.</td>
</tr>
<tr>
<td>void</td>
<td><code>setNullVoid()</code></td>
<td>Sets the Value to NULL.</td>
</tr>
<tr>
<td>Value</td>
<td>method</td>
<td>description</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------------------------------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>Value</td>
<td>setOID(long v)</td>
<td>Sets the Value.</td>
</tr>
<tr>
<td>void</td>
<td>setOIDVoid(long v)</td>
<td>Sets the OID Value.</td>
</tr>
<tr>
<td>Value</td>
<td>setString(String v)</td>
<td>Sets the Value.</td>
</tr>
<tr>
<td>void</td>
<td>setStringVoid(String v)</td>
<td>Sets the Value.</td>
</tr>
<tr>
<td>Value</td>
<td>setText(TextStream v)</td>
<td>Sets the Value.</td>
</tr>
<tr>
<td>Value</td>
<td>setTimestamp(Calendar v)</td>
<td>Sets the Value.</td>
</tr>
<tr>
<td>Value</td>
<td>setTimestamp(Date v)</td>
<td>Sets the Value.</td>
</tr>
<tr>
<td>Value</td>
<td>setTimestamp(int year, int month, int day, int hour, int minutes, int seconds, int millisec)</td>
<td>Sets the Value.</td>
</tr>
<tr>
<td>void</td>
<td>setTimestampVoid(int year, int month, int day, int hour, int minutes, int seconds, int millisecs)</td>
<td>Sets the Value.</td>
</tr>
<tr>
<td>void</td>
<td>setTimestampVoid(long v)</td>
<td>Sets the Value.</td>
</tr>
<tr>
<td>void</td>
<td>setVoid(Value v)</td>
<td>Sets the Value.</td>
</tr>
<tr>
<td>String</td>
<td>toString()</td>
<td>Gets a String representation of the Value.</td>
</tr>
<tr>
<td>String</td>
<td>toString(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.

getTimestampAsCalendar

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 a 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**

```java
public void setDoubleVoid(double v)
```

Sets the Value.

**Parameters:**

v - [in] New Double value.

---

**setText**

```java
public Value setText(TextStream v)
```

Sets the Value.

**Parameters:**

v - New value.

**Returns:**

The calling instance.

---

**compareTo**

```java
public int compareTo(Value v)
```

See compare().

**Parameters:**

v - null

---

**setBoolean**

```java
public Value setBoolean(boolean v)
```

Sets the Value.

**Parameters:**

v - New value.

**Returns:**

The calling instance.
**setTimestamp**

```java
public Value setTimestamp(Calendar v)
```

Sets the Value.

**Parameters:**
- `v` - New value.

**Returns:**
- The calling instance.

**getBoolean**

```java
public boolean getBoolean()
```

Gets Boolean Value.

This must be a non-NULL Boolean Value.

**Returns:**
- The Boolean Value.

**setTimestampVoid**

```java
public void setTimestampVoid(long v)
```

Sets the Value.

**Parameters:**

**setDouble**

```java
public Value setDouble(double v)
```

Sets the Value.

**Parameters:**
- `v` - New value.

**Returns:**
- The calling instance.

**setNullVoid**

```java
public void setNullVoid()
```

Sets the Value to NULL.
**setOIDVoid**

```java
public void setOIDVoid(long v)
```

Sets the OID Value.

**Parameters:**
- `v` - [in] New OID value.

**getDataType**

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

Gets the DataType.

Value cannot be NULL.

**Returns:**
- The DataType.

**setLong**

```java
public Value setLong(long v)
```

Sets the Value.

**Parameters:**
- `v` - New value.

**Returns:**
- The calling instance.

**getTimestampAsDate**

```java
public Date getTimestampAsDate()
```

Gets the Value as a Date instance.

**Returns:**
- The returning Date instance.

**getString**

```java
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**

public void **setVoid** (Value v)

Sets the Value.

**Parameters:**
- v - [in] New value.

**setNull**

public Value **setNull** ()

Sets the Value to NULL.

**Returns:**
The calling instance.

**setOID**

public Value **setOID** (long v)

Sets the Value.

**Parameters:**
- v - New value.

**Returns:**
The calling instance.

**setStringVoid**

public void **setStringVoid** (String v)

Sets the Value.

**Parameters:**
- v - [in] New String value.

**getLong**

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.dex.gdb
## Class Values

![Class Hierarchy](com.sparsity.dex.gdb.Values.png)

### All Implemented Interfaces:
- Iterable
- Closeable

### public class Values
extends Object
implements Closeable, Iterable

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](http://www.sparsity-technologies.com)

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>close()</td>
<td>void</td>
<td>Closes the Values instance.</td>
</tr>
<tr>
<td>count()</td>
<td>long</td>
<td>Gets the number of elements into the collection.</td>
</tr>
<tr>
<td>isOpened()</td>
<td>boolean</td>
<td>Gets if Values instance has been closed or not.</td>
</tr>
<tr>
<td>iterator()</td>
<td>ValuesIterator</td>
<td>See iterator().</td>
</tr>
<tr>
<td>iterator(Order order)</td>
<td>ValuesIterator</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.io.Closeable

- close

### Methods inherited from interface java.lang.Iterable

- iterator
Methods

iterator
public ValuesIterator iterator()

See iterator().

Creates an Ascendent iterator.

count
public long count()

Gets the number of elements into the collection.

Returns:
The number of elements into the collection.

iterator
public ValuesIterator iterator(Order order)

Gets a ValuesIterator.

Parameters:
order - [in] Ascending or descending order.

Returns:
ValuesIterator instance.

isClosed
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
public void close()

Closes the Values instance.

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

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.util.Iterator

hasNext, next, remove

Methods inherited from interface java.io.Closeable

close
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.dex.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 DEX User Manual for more details on this.

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

---

### Constructor Summary

<table>
<thead>
<tr>
<th>Method</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>
void open(String f)  
    Opens the source file path.

boolean read(StringList row)  
    Reads the next row as a string array.

boolean reset()  
    Moves the reader to the beginning.

void setLocale(String localeStr)  
    Sets the locale that will be used to read the file.

void setMultilines(int numExtralines)  
    Allows the use of fields with more than one line.

void setNumLines(int numLines)  
    Used to limit the number of lines that will be read.

void setQuotes(String quotes)  
    Sets the character used to quote fields.

void setSeparator(String sep)  
    Sets the character used to separate fields in the file.

void setSingleLine()  
    Only allows single line fields.

void setStartLine(int startLine)  
    Sets the number of lines to be skipped from the beginning.

Methods inherited from class com.sparsity.dex.io.RowReader

close, getRow, read, reset

Methods inherited from class java.lang.Object

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

Constructors

**CSVReader**

public **CSVReader**()  

    Constructs CSVReader.

Methods

**setMultilines**

public void **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).

**setNumLines**

```java
public void 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)

**reset**

```java
public boolean reset()
```

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
public void close()
```

Closes the reader.

**setStartLine**

```java
public void setStartLine(int startLine)
```

Sets the number of lines to be skiped from the beginning.

Parameters:
startLine - [in] The line number to skip for start reading

**open**

```java
public void 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.

---

**setSeparator**

```java
public void setSeparator(String sep)
throws RuntimeException
```

Sets the character used to separate fields in the file.

**Parameters:**

- sep - [in] Separator character.

**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.

---

**setLocale**

```java
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.

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.
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 DEX User Manual.

Check out the 'Data export' section in the DEX 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 CSVWriter()</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 close()</td>
<td>Closes the writer.</td>
</tr>
<tr>
<td>void open(String f)</td>
<td>Opens the output file path.</td>
</tr>
<tr>
<td>void setAutoQuotes(boolean autoquotes)</td>
<td>Sets on/off the automatic quote mode.</td>
</tr>
<tr>
<td>void 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 setLocale(String localeStr)</td>
<td>Sets the locale that will be used to write the file.</td>
</tr>
<tr>
<td>void setQuotes(String quotes)</td>
<td>Sets the character used to quote fields.</td>
</tr>
<tr>
<td>void setSeparator(String sep)</td>
<td>Sets the character used to separate fields in the file.</td>
</tr>
</tbody>
</table>
void write(StringList row)
   Writes the next row.

Methods inherited from class com.sparsity.dex.io.RowWriter
   close, write

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

```java
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.

---

setForcedQuotes

```java
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.

---

setAutoQuotes

```java
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.

**Parameters:**

- `autoquotes` - [in] If TRUE it enables the automatic quote mode, if FALSE it disables it.

---

close

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

Closes the writer.

---

write

```java
public void write(StringList row)
```

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
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.dex.io
Class EdgeTypeExporter

java.lang.Object
   +-com.sparsity.dex.io.TypeExporter
     +-com.sparsity.dex.io.EdgeTypeExporter

public class EdgeTypeExporter
extends TypeExporter

EdgeTypeExporter class.

Specific TypeExporter implementation for edge types.

Check out the 'Data export' section in the DEX 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>EdgeTypeExporter(RowWriter rw, Graph g, int t, AttributeList attrs, int hPos, int tPos, int hAttr, int tAttr)</td>
<td>Creates a new instance.</td>
</tr>
<tr>
<td>public</td>
<td>EdgeTypeExporter()</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(TypeExporterListener tel)</td>
<td>Registers a new listener.</td>
</tr>
<tr>
<td>void</td>
<td>run()</td>
<td>See TypeExporter::Run.</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 that will be exported.</td>
</tr>
<tr>
<td>void</td>
<td>setHeadAttribute(int attr)</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>void</td>
<td>setHeader(boolean header)</td>
<td>Sets the presence of a header row.</td>
</tr>
<tr>
<td>void</td>
<td>setHeadPosition(int pos)</td>
<td>Sets the position (index column) of the head attribute in the exported data.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>setRowWriter(RowWriter rw)</td>
<td>Sets the output data destination.</td>
<td></td>
</tr>
<tr>
<td>setTailAttribute(int attr)</td>
<td>Sets the attribute that will be used to get the value to be dumped for the tail of the edge.</td>
<td></td>
</tr>
<tr>
<td>setTailPosition(int pos)</td>
<td>Sets the position (index column) of the tail attribute in the exported data.</td>
<td></td>
</tr>
<tr>
<td>setType(int t)</td>
<td>Sets the type to be exported.</td>
<td></td>
</tr>
</tbody>
</table>

**Methods inherited from class** `com.sparsity.dex.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.
setTailPosition

public void setTailPosition(int pos)

Sets the position (index column) of the tail attribute in the exported data.

Parameters:

pos - [in] Tail position

setAttributes

public void setAttributes(AttributeList attrs)

Sets the list of Attributes.

Parameters:

attrs - [in] Attribute identifiers to be exported

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

setRowWriter

public void setRowWriter(RowWriter rw)

Sets the output data destination.

r - [in] Input RowWriter.

Parameters:

rw - null

register

public void register(TypeExporterListener tel)

Registers a new listener.

Parameters:

tel - [in] TypeExporterListener to be registered.
**run**

public void **run**()
   throws IOException, RuntimeException

   See TypeExporter::Run.

**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.

**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

**setGraph**

public void **setGraph**(Graph g)

   Sets the graph that will be exported.

   **Parameters:**
   g - [in] Graph.

**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

**setType**

public void **setType**(int t)
Sets the type to be exported.

**Parameters:**

- t - [in] Type identifier.

---

**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
com.sparsity.dex.io
Class EdgeTypeLoader

public class EdgeTypeLoader
extends TypeLoader

EdgeTypeLoader class.
Specific TypeLoader implementation for edge types.
Check out the 'Data import' section in the DEX User Manual for more details on this.

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

Constructor Summary

| public | EdgeTypeLoader(RowReader rr, Graph g, int t, AttributeList attrs, Int32List attrsPos, int hPos, int tPos, int hAttr, int tAttr) |
|        | Creates a new instance. |
| public | EdgeTypeLoader() |
|        | Creates a new instance. |

Method Summary

| void | register(TypeLoaderListener tel) |
|      | Registers a new listener. |
| void | run() |
|      | See TypeLoader::Run. |
| void | runNPhases(int partitions) |
|      | See TypeLoader::RunNPhases. |
| void | runTwoPhases() |
|      | See TypeLoader::RunTwoPhases. |
| 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 **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**()
Truns 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.dex.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
default EdgeTypeLoader(RowReader rr, Graph g, Int t,
                           AttributeList attrs,
                           Int32List attrsPos,
                           Int hPos,
                           int tPos,
                           int hAttr,
                           int tAttr)
```

Creates a new instance.
Parameters:
- `r` - [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

```java
public EdgeTypeLoader()
```

Creates a new instance.

### Methods

#### 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.

#### 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

#### 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.
**setAttributes**

public void **setAttributes**([AttributeList] attrs)

Sets the list of Attributes.

**Parameters:**

attrs - [in] Attribute identifiers to be loaded

---

**runTwoPhases**

public void **runTwoPhases**()

throws IOException, RuntimeException

See TypeLoader::RunTwoPhases.

---

**setHeadAttribute**

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

---

**setRowReader**

public void **setRowReader**([RowReader] rr)

Sets the input data source.

**Parameters:**

rr - [in] Input RowReader.

---

**runNPhases**

public void **runNPhases**(int partitions)

throws IOException, RuntimeException

See TypeLoader::RunNPhases.

**Parameters:**

partitions - null

**Throws:**

java.io.IOException - null
java.lang.RuntimeException - 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 TypeLoader::Run.

setAttributePositions

public void setAttributePositions(Int32List attrsPos)

Sets the list of attribute positions.

attrsPos [in] Attribute positions (column index >=0).

Parameters:

attrsPos - 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:

freq - null

setGraph

public void setGraph(Graph g)
Sets the graph where the data will be loaded.

**Parameters:**

\[ g \text{ - [in] Graph.} \]

### setTimestampFormat

```java
public void setTimestampFormat(String timestampFormat)
```

Sets a specific timestamp format.

**Parameters:**

\[ timestampFormat \text{ - null} \]

### setType

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

Sets the type to be loaded.

**Parameters:**

\[ t \text{ - [in] Type identifier.} \]

### setTailAttribute

```java
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:**

\[ attr \text{ - [in] Tail Attribute} \]

### setHeadPosition

```java
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:**

\[ pos \text{ - [in] Head position} \]
public class NodeTypeExporter
extends TypeExporter

NodeTypeExporter class.

Specific TypeExporter implementation for node types.

Check out the 'Data export' section in the DEX User Manual for more details on this.

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

Constructor Summary

<table>
<thead>
<tr>
<th>public</th>
<th>NodeTypeExporter(RowWriter rw, Graph g, int t, AttributeList attrs)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Creates a new instance.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>public</th>
<th>NodeTypeExporter()</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>register(TypeExporterListener tel)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Registers a new listener.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>run()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>See TypeExporter::Run.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>setAttributes(AttributeList attrs)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sets the list of Attributes.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>setFrequency(int freq)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sets the frequency of listener notification.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>setGraph(Graph g)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sets the graph that will be exported.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>setHeader(boolean header)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sets the presence of a header row.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>setRowWriter(RowWriter rw)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sets the output data destination.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>setType(int t)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sets the type to be exported.</td>
</tr>
</tbody>
</table>

Methods inherited from class com.sparsity.dex.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.

Parameters:
    rw - null

register

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 TypeExporter::Run.

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.

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

setAttributes

```java
public void setAttributes(AttributeList attrs)
```

Sets the list of Attributes.

Parameters:

- attrs - [in] Attribute identifiers to be exported

setGraph

```java
public void setGraph(Graph g)
```

Sets the graph that will be exported.

Parameters:

- g - [in] Graph.
**setType**

public void `setType(int t)`

Sets the type to be exported.

**Parameters:**

- `t` - [in] Type identifier.
**com.sparsity.dex.io**  
**Class NodeTypeLoader**

java.lang.Object  
    +-com.sparsity.dex.io.TypeLoader  
    +-com.sparsity.dex.io.NodeTypeLoader

public class NodeTypeLoader  
extends TypeLoader

NodeTypeLoader class.  
Specific TypeLoader implementation for node types.  
Check out the 'Data import' section in the DEX User Manual for more details on this.  
**Author:**  
Sparsity Technologies http://www.sparsity-technologies.com

### Constructor Summary

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

<table>
<thead>
<tr>
<th>public</th>
<th>NodeTypeLoader(RowReader rr, Graph g, int t, AttributeList attrs, Int32List attrsPos)</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>register(TypeLoaderListener tel)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Registers a new listener.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>run()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>See TypeLoader::Run.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>runNPhases(int partitions)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>See TypeLoader::RunNPhases.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>runTwoPhases()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>See TypeLoader::RunTwoPhases.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>setAttributePositions(Int32List attrsPos)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sets the list of attribute positions.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>setAttributes(AttributeList attrs)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sets the list of Attributes.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>setFrequency(int freq)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sets the frequency of listener notification.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>setGraph(Graph g)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sets the graph where the data will be loaded.</td>
</tr>
</tbody>
</table>
void \texttt{setLocale(String localeStr)}
Sets the locale that will be used to read the data.

void \texttt{setLogError(String path)}
Sets a log error file.

void \texttt{setLogOff()}
Truns off all the error reporting.

void \texttt{setRowReader(RowReader rr)}
Sets the input data source.

void \texttt{setTimestampFormat(String timestampFormat)}
Sets a specific timestamp format.

void \texttt{setType(int t)}
Sets the type to be loaded.

Methods inherited from class \texttt{com.sparsity.dex.io.TypeLoader}
\texttt{register, run, runNPhases, runTwoPhases, setAttributePositions, setAttributes, setFrequency, setGraph, setLocale, setLogError, setLogOff, setRowReader, setTimestampFormat, setType}

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

Constructors

\textbf{NodeTypeLoader}

\texttt{public NodeTypeLoader()}  

Creates a new instance.

\textbf{NodeTypeLoader}

\texttt{public NodeTypeLoader(RowReader \texttt{rr},}
\texttt{\hspace{1cm}Graph \texttt{g,}}
\texttt{\hspace{1cm}int \texttt{t,}}
\texttt{\hspace{1cm}AttributeList \texttt{attrs,}}
\texttt{\hspace{1cm}Int32List \texttt{attrsPos})}

Creates a new instance.

Parameters:
\texttt{rr} - [in] Input RowReader.
\texttt{g} - [in] Graph.
\texttt{t} - [in] Type identifier.
\texttt{attrs} - [in] Attribute identifiers to be loaded.
\texttt{attrsPos} - [in] Attribute positions (column index >=0).
Methods

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.

setLogError

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.

setAttributes

public void setAttributes(AttributeList attrs)

Sets the list of Attributes.

Parameters:

attrs - [in] Attribute identifiers to be loaded

runTwoPhases

public void runTwoPhases()
throws IOException, RuntimeException

See TypeLoader::RunTwoPhases.

setRowReader

public void setRowReader(RowReader rr)

Sets the input data source.

Parameters:

rr - [in] Input RowReader.
runNPhases

public void runNPhases(int partitions)
throws IOException, RuntimeException

See TypeLoader::RunNPhases.

Parameters:
partitions - null

Throws:
java.io.IOException - null
java.lang.RuntimeException - 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 TypeLoader::Run.

setAttributePositions

public void setAttributePositions(Int32List attrsPos)

Sets the list of attribute positions.

attrsPos [in] Attribute positions (column index >=0).

Parameters:
attrsPos - 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

**setGraph**

```java
public void setGraph(Graph g)
```

Sets the graph where the data will be loaded.

**Parameters:**
- g - [in] Graph.

**setTimestampFormat**

```java
public void setTimestampFormat(String timestampFormat)
```

Sets a specific timestamp format.

**Parameters:**
- timestampFormat - null

**setType**

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

Sets the type to be loaded.

**Parameters:**
- t - [in] Type identifier.
com.sparsity.dex.io
Class RowReader

java.lang.Object
   +-com.sparsity.dex.io.RowReader

Direct Known Subclasses:
   CSVReader

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 DEX 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 close()</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Closes the reader.</td>
</tr>
<tr>
<td>----------------</td>
</tr>
<tr>
<td>intgetRow()</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>The row number for the current row.</td>
</tr>
<tr>
<td>----------------</td>
</tr>
<tr>
<td>booleanread(StringList row)</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Reads the next row as a string array.</td>
</tr>
<tr>
<td>----------------</td>
</tr>
<tr>
<td>booleanreset()</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<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**

```java
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.

**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.

**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.
com.sparsity.dex.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 DEX User Manual for more details on this.

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>close</td>
<td>void close()</td>
<td>Closes the writer.</td>
</tr>
<tr>
<td>write</td>
<td>void write(StringList 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

Methods

close

public void close()
throws IOException, RuntimeException

Closes the writer.

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
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 DEX 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>register(TypedExporterListener tel)</code></td>
<td>Registers a new listener.</td>
</tr>
<tr>
<td><code>run()</code></td>
<td>Runs export process.</td>
</tr>
<tr>
<td><code>setAttribute(AttributeList attrs)</code></td>
<td>Sets the list of Attributes.</td>
</tr>
<tr>
<td><code>setFrequency(int freq)</code></td>
<td>Sets the frequency of listener notification.</td>
</tr>
<tr>
<td><code>setGraph(Graph g)</code></td>
<td>Sets the graph that will be exported.</td>
</tr>
<tr>
<td><code>setHeader(boolean header)</code></td>
<td>Sets the presence of a header row.</td>
</tr>
<tr>
<td><code>setRowWriter(RowWriter rw)</code></td>
<td>Sets the output data destination.</td>
</tr>
<tr>
<td><code>setType(int t)</code></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, wait
Methods

setRowWriter

```java
public void setRowWriter(RowWriter rw)
```

Sets the output data destination.

\( \text{rw} \) [in] Input RowWriter.

**Parameters:**
- \( \text{rw} \) - null

run

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

Throws IOException,

- RuntimeException

Runs export process.

register

```java
public void register(TypeExporterListener tel)
```

Registers a new listener.

**Parameters:**
- \( \text{tel} \) - [in] TypeExporterListener to be registered.

setHeader

```java
public void setHeader(boolean header)
```

Sets the presence of a header row.

**Parameters:**
- \( \text{header} \) - [in] If TRUE, a header row is dumped with the name of the attributes.

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:**
- \( \text{freq} \) - null
**setAttributes**

public void **setAttributes**([AttributeList] attrs)

Sets the list of Attributes.

**Parameters:**

.attrs - [in] Attribute identifiers to be exported

**setGraph**

public void **setGraph**([Graph] g)

Sets the graph that will be exported.

**Parameters:**

.g - [in] Graph.

**setType**

public void **setType**([int] t)

Sets the type to be exported.

**Parameters:**

.t - [in] Type identifier.
com.sparsity.dex.io
Class TypeExporterEvent

public class TypeExporterEvent extends Object

Provides information about the progress of an TypeExproter instance.

Check out the 'Data export' section in the DEX 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>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getCount</td>
<td>long</td>
<td>Gets the current number of objects exported.</td>
</tr>
<tr>
<td>getTotal</td>
<td>long</td>
<td>Gets the total number of objects exported.</td>
</tr>
<tr>
<td>getTypeId</td>
<td>int</td>
<td>Gets the type identifier.</td>
</tr>
<tr>
<td>isLast</td>
<td>boolean</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.
class TypeExporterListener

public class TypeExporterListener
        extends Object

Interface to be implemented to receive TypeExporterEvent events from a TypeExporter.

Check out the 'Data export' section in the DEX User Manual for more details on this.

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

Method Summary

void | notifyEvent(TypeExporterEvent tee)

    Method to be notified from a TypeExporter.

Methods inherited from class java.lang.Object

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

Methods

notifyEvent

public void notifyEvent(TypeExporterEvent tee)

    Method to be notified from a TypeExporter.

Parameters:
    tee - [in] Notified event.
com.sparsity.dex.io

Class TypeLoader

```java
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 DEX 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 register(TypeLoaderListener tel)</code></td>
<td>Registers a new listener.</td>
</tr>
<tr>
<td><code>void run()</code></td>
<td>Run the loader.</td>
</tr>
<tr>
<td><code>void runNPhases(int partitions)</code></td>
<td>Run the loader for N phases loading.</td>
</tr>
<tr>
<td><code>void runTwoPhases()</code></td>
<td>Run the loader for two phases loading.</td>
</tr>
<tr>
<td><code>void setAttributePositions(Int32List attrsPos)</code></td>
<td>Sets the list of attribute positions.</td>
</tr>
<tr>
<td><code>void setAttributes(AttributeList attrs)</code></td>
<td>Sets the list of Attributes.</td>
</tr>
<tr>
<td><code>void setFrequency(int freq)</code></td>
<td>Sets the frequency of listener notification.</td>
</tr>
<tr>
<td><code>void setGraph(Graph g)</code></td>
<td>Sets the graph where the data will be loaded.</td>
</tr>
<tr>
<td><code>void setLocale(String localeStr)</code></td>
<td>Sets the locale that will be used to read the data.</td>
</tr>
<tr>
<td><code>void setLogError(String path)</code></td>
<td>Sets a log error file.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>void setLogOff()</code></td>
<td>Truns off all the error reporting.</td>
</tr>
<tr>
<td><code>void setRowReader(RowReader rr)</code></td>
<td>Sets the input data source.</td>
</tr>
<tr>
<td><code>void setTimestampFormat(String timestampFormat)</code></td>
<td>Sets a specific timestamp format.</td>
</tr>
<tr>
<td><code>void setType(int t)</code></td>
<td>Sets the type to be loaded.</td>
</tr>
</tbody>
</table>

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

---

### Methods

**runTwoPhases**

```java
public 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.

**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() throws IOException, RuntimeException
```

Run the loader.

**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.

setAttributePositions
public void setAttributePositions(Int32List attrsPos)

Sets the list of attribute positions.

Parameters:
    attrsPos [in] Attribute positions (column index >=0).

setAttributes
public void setAttributes(AttributeList attrs)

Sets the list of Attributes.

Parameters:
    attrs - [in] Attribute identifiers to be loaded

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.

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.

setAttributePositions
public void setAttributePositions(Int32List attrsPos)

Sets the list of attribute positions.

Parameters:
runNPhases

public 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

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

setGraph

public void setGraph(Graph g)

    Sets the graph where the data will be loaded.

    Parameters:
    g - [in] Graph.

setTimestampFormat

public void setTimestampFormat(String timestampFormat)

    Sets a specific timestamp format.

    Parameters:
    timestampFormat - null

setType

public void setType(int t)
Sets the type to be loaded.

**Parameters:**

τ - [in] Type identifier.
com.sparsity.dex.io
Class TypeLoaderEvent

java.lang.Object
   +-com.sparsity.dex.io.TypeLoaderEvent

public class TypeLoaderEvent
extends Object

Provides information about the progress of a TypeLoader instance.

Check out the 'Data import' section in the DEX User Manual for more details on this.

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

Method Summary

<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>long</td>
<td>getCount()</td>
<td>Gets the current number of objects created.</td>
</tr>
<tr>
<td>int</td>
<td>getPartition()</td>
<td>Gets the current partition.</td>
</tr>
<tr>
<td>int</td>
<td>getPhase()</td>
<td>Gets the current phase.</td>
</tr>
<tr>
<td>int</td>
<td>getTotalPartitions()</td>
<td>Gets the total number of partitions.</td>
</tr>
<tr>
<td>int</td>
<td>getTotalPartitionSteps()</td>
<td>Gets the total number of steps in the current partition.</td>
</tr>
<tr>
<td>int</td>
<td>getTotalPhases()</td>
<td>Gets the total number of phases.</td>
</tr>
<tr>
<td>int</td>
<td>getTypeId()</td>
<td>Gets the type identifier.</td>
</tr>
<tr>
<td>boolean</td>
<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, wait

Methods

getTotalPhases

public int getTotalPhases()
Gets the total number of phases.

**Returns:**
- The total number of phases.

---

### getCount

**Description:**
- Gets the current number of objects created.

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

**Returns:**
- The current number of objects created.

---

### getTotalPartitionSteps

**Description:**
- Gets the total number of steps in the current partition.

```java
public int getTotalPartitionSteps()
```

**Returns:**
- The total number steps in the current partition.

---

### isLast

**Description:**
- Gets if this is the last event or not.

```java
public boolean isLast()
```

**Returns:**
- TRUE if this is the last event, FALSE otherwise.

---

### getPartition

**Description:**
- Gets the current partition.

```java
public int getPartition()
```

**Returns:**
- The current partition.

---

### getTypeId

**Description:**
- Gets the type identifier.

```java
public int getTypeId()
```

**Returns:**
- The type identifier.
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.
Class TypeLoaderListener

public class TypeLoaderListener extends Object

Interface to be implemented to receive TypeLoaderEvent events from a TypeLoader.

Check out the 'Data import' section in the DEX 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 notifyEvent(TypeLoaderEvent ev)</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

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.dex.script
public class ScriptParser
extends Object

ScriptParser.

The ScriptParser can create schemas and load data from a set of commands in a dex script.

A DEX 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 DEX schema.

Check out the 'Scripting' chapter in the DEX User Manual for a comprehensive explanation on the grammar of the DEX commands and how they work.

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

### Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
</tr>
</thead>
<tbody>
<tr>
<td>public ScriptParser()</td>
</tr>
<tr>
<td>Constructor.</td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>static void generateSchemaScript(String path, Database db)</td>
</tr>
<tr>
<td>Writes an script with the schema definition for the given database.</td>
</tr>
<tr>
<td>static void main()</td>
</tr>
<tr>
<td>Executes ScriptParser for the given file path.</td>
</tr>
<tr>
<td>boolean parse(String path, boolean execute, String localeStr)</td>
</tr>
<tr>
<td>Parses the given input file.</td>
</tr>
<tr>
<td>void setErrorLog(String path)</td>
</tr>
<tr>
<td>Sets the error log.</td>
</tr>
<tr>
<td>void setOutputLog(String path)</td>
</tr>
<tr>
<td>Sets the output log.</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

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

---

**com.sparsity.dex.script**

**Class ScriptParser**

java.lang.Object

+-com.sparsity.dex.script.ScriptParser
ScriptParser

public ScriptParser()

Constructor.

Methods

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.

parse

public boolean parse(String path,
                    boolean execute,
                    String localeStr)

throws IOException

Parses the given input file.

Parameters:
path - [in] Input file path.
execlute - [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:
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, 150, 170, 173, 182, 197, 208
addAll 174
addAllEdgeTypes 12, 14, 19, 24, 29, 36, 41, 44, 48, 52, 57
addAllNodeTypes 7, 12, 14, 19, 24, 29, 32, 36, 41, 44, 48, 52, 56
addEdgeType 11, 14, 18, 23, 28, 33, 37, 40, 45, 49, 53, 57
addNodeType 8, 11, 15, 18, 23, 29, 33, 38, 40, 45, 49, 53, 57
addWeightedEdgeType 28
Any 122
any 168
Ascendent 186
asDirected 120
AttributeList 66

B

backup 138
Basic 63
begin 193
Between 81
Boolean 89
BooleanList 75
Box 163

C

checkOnlyExistence 24
clear 66, 75, 150, 171, 183, 196, 208
close 5, 8, 12, 15, 41, 84, 97, 171, 178, 193, 202, 225, 227, 231, 236, 258, 260
combineDifference 174
combineIntersection 171
combineUnion 169
commit 193
cmpare 216
cmpareTo 217, 220
cmpute 10, 11
Config 155
ConnectedComponents 4
contains 168, 171
containsAll 169

Context 10
copy 170, 175
count 66, 76, 151, 171, 183, 196, 209, 225
countEdges 135
countNodes 139
create 96
CSVReader 230
CSVWriter 235

D

Debug 156
DefaultExport 92
degree 133
Descendent 187
dex 96
dexConfig 101
difference 169
Double 89
drop 141, 142
dumpData 131
dumpStorage 133

E

Edge 180
EdgeExport 117
edges 137
EdgeTypeExporter 239
EdgeTypeLoader 244, 245
enableType 92, 125
Equal 80
equals 168, 170, 215, 221
excludeEdges 8, 11, 15, 19, 24, 29, 34, 38, 41, 46, 50, 53, 58
excludeNodes 7, 12, 14, 19, 24, 29, 32, 37, 41, 45, 49, 52, 57
exists 18, 22, 28, 174
expode 140, 142
export 134

F

findAttribute 136
findAttributes 139
findEdge 142
findEdgeTypes 136
findNodeTypes 138
findObject 144
findType 145
findTypes 131
Fine 155

G

generateSchemaScript 279
get 111
getAlias 83
getAreNeighborsIndexed 204
getAttribute 138, 143, 145
getAttributeIntervalCount 132
getAttributeStatistics 134
generate schema script 279
newObjects 193
newRestrictedEdgeType 145
newSession 84
newSessionAttribute 139, 140
next 42, 45, 49, 69, 78, 153, 178, 185, 199, 211, 227
nextAttribute 69
nextBoolean 78
nextInt32 152
nextObject 177
nextOID 185
nextString 199
nextType 211
Node 179
NodeExport 158
NodeTypeExporter 250
NodeTypeLoader 254
NotEqual 81
notifyEvent 267, 276
O
Off 155
OID 90
OIDList 182
open 97, 231, 237
Outgoing 122
P
parse 279
PlatformStatistics 189
prepare 93, 125
R
read 201, 233, 259
RegExp 81
register 240, 247, 250, 256, 263, 270
release 93, 125
remove 69, 78, 153, 168, 172, 178, 185, 199, 211, 227
removeAll 173
removeAttribute 138
removeType 145
reset 231, 259
restore 96
retainAll 172
Round 164
run 7, 14, 18, 24, 30, 32, 37, 52, 57, 240, 247, 251, 256, 263, 269
runNPhases 246, 256, 271
runTwoPhases 246, 255, 269
S
sample 175
ScriptParser 278
select 132, 135, 137
set 183, 216
setAsDirected 119
setAttribute 132
setAttributeDefaultValue 136
setAttributePositions 247, 256, 270
setAttributes 240, 245, 251, 255, 263, 270
setAttributeText 131
setAutoQuotes 236
setBoolean 217
setBooleanVoid 216
setCacheMaxSize 105
setCacheStatisticsEnabled 109
setCacheStatisticsFile 107
setCacheStatisticsSnapshotTime 103
setColor 119, 161
setColorRGB 120, 162
setDefaults 118, 148, 162
setDouble 218
setDoubleVoid 217
setErrorLog 279
setExtentPages 108
setExtentSize 104
setFit 160
setFontSize 118, 160
setForcedQuotes 236
setFrequency 241, 247, 251, 257, 263, 271
setGraph 241, 247, 251, 257, 264, 271
setHeadAttribute 240, 246
setHeader 241, 251, 263
setHeadPosition 242, 248
setHeight 160
setHighAvailabilityCoordinators 103
setHighAvailabilityEnabled 102
setHighAvailabilityIP  102
setHighAvailabilityMasterHistory  108
setHighAvailabilitySynchronization  106
setInteger  223
setIntegerVoid  221
setLabel  120, 147, 161
setLabelColor  120, 162
setLabelColorRGB  118, 159
setLicense  106
setLocale  232, 235, 247, 256, 270
setLogError  245, 255, 269
setLogFile  110
setLogLevel  104
setLogOff  245, 254, 269
setLong  219
setLongVoid  215
setMaterializedAttribute  7, 33, 37, 53, 57
setMaximumHops  10, 15, 20, 25, 30, 42, 46, 50
setMultilines  230
setNull  222
setNullVoid  218
setNumLines  231
setOID  222
setOIDVoid  219
setOutputLog  280
setPoolFrameSize  106
setPoolPersistentMaxSize  105
setPoolPersistentMinSize  102
setPoolTemporaryMaxSize  106
setPoolTemporaryMinSize  108
setQuotes  232, 235
setRecoveryCacheMaxSize  102
setRecoveryCheckpointTime  110
setRecoveryEnabled  106
setRecoveryLogFile  105
setRowReader  246, 255, 270
setRowWriter  240, 250, 263
setSeparator  232, 235
setShape  162
setSingleLine  232
setStartLine  231
setString  216
setStringVoid  222
setTailAttribute  241, 248
setTailPosition  239, 245
setText  217
setTimestamp  218, 220, 223
setTimestampFormat  248, 257, 271
setTimestampVoid  218, 220
setType  241, 248, 251, 257, 264, 271
setUnweightedEdgeCost  30
setVoid  222
setWidth  118, 159
Severe  155
SinglePairShortestPathBFS  22
SinglePairShortestPathDijkstra  27
size  175
String  89
StringList  196
StrongConnectivityGabow  36
T
tails  141
tailsAndHeads  133
Text  90
TextStream  201
Timestamp  89
toArray  173, 174
toString  221, 223
TraversalBFS  44
TraversalDFS  48
TypeList  208
U
union  172
Unique  64
V
Value  214, 215
valueOf  64, 82, 90, 122, 127, 156, 164, 180, 187
values  64, 81, 90, 122, 127, 156, 164, 180, 187
Version  96
W
Warning  155
WeakConnectivityDFS  56
write 201, 236, 260

Y

YGraphML 127