Memento API GraphStream

http://litis.univ-lehavre.fr/∼guinand/

1 Graph Elements Creation/Deletion/Access

1.1 Creation

- **SingleGraph Creation**:

```
SingleGraph g = new SingleGraph();
```

- **Node creation**:

```
Node t = g.addNode("idNode");
```

- **Edge creation**:

```
Edge e = g.addEdge("idEdge","id$S_0$","id$S_1$")
```

- **Arc creation**:

```
Edge e = g.addEdge("idEdge","id$S_0$","id$S_1$",true)
```

1.2 Deletion

- **Removing all elements from a graph**:

```
g.clear();
```

- **Node deletion**:

```
g.removeNode("idNode");
```

- **Edge deletion**:

```
g.removeEdge("idEdge")
```

1.3 Access

- **Node Access**:

  `Node t = g.getNode("idNode");`

  `Node t = g.getNode(index);` (where index is an integer value strictly lower than the number of nodes of the graph)

- **Edge Access**:

  `Edge e = g.getEdge("idEdge")`

  `Edge e = g.getEdge(index)`

2 Lists

- **List** of vertices:

  `g.getNodeSet()`

- **List** of edges:

  `g.getEdgeSet()`

- **Number of vertices** of Graph g :

  `g.getNodeCount()`

3 Display

- Displaying the graph with **automatic** layout of vertices:

  `g.display();`

- Without automatic layout:

  `g.display(false);`

4 Attributes of vertices and edges/arcs

You can add as many attributes as needed to the elements (vertices/edges or ars) of your graph.

- for **adding** a new **attribute** to a vertex/edge/arc or to **change its value** :

  `element.setAttribute("attribute name", value);`

- for **verifying** is an element has a specific attribute:

  `element.hasAttribute("attribute name");` returns `true` if the element has an attribute with that name and `false` otherwise.

Some attributes are reserved by the API, like ”x”, ”y” or ”ui.label”, ”ui.style”. Examples:

- for changing the **label** of an element (label is not identifier):

```
element.setAttribute("ui.label","element label");
```

- for changing the **position (coordinates) of a vertex** (integer values):

```
sommet.setAttribute("x",20);
sommet.setAttribute("y",50);
```

5 Common Style Attributes

- For changing the **color** of an element (vertex/link), there exist several possibilities:
  1. `element.setAttribute("ui.style","fill-color:red;")`
  2. `"fill-color:rgb(120,159,204);"` where 120 corresponds to red, 159 to green and 204 to blue (values have to belong to the interval [0:255])
  3. `"fill-color:#a5c8bb;"` where $a5$ corresponds to red, $c8$ to green and bb to blue (hexadecimal values, interval: $[00{:}ff]$)
- for changing the **size** of the element (thickness for a link):

```
element.setAttribute("ui.style","size:5;");
```

- for changing the **shape** of a vertex:

```
element.setAttribute("ui.style","shape:  cross;")
```

→ **don't forget the ";" at the end of the String defining the value of a style attribute.**

6 Degree, Neighbors and Neighborhood

- **Degree of a vertex** v:

```
v.getDegree();
```

- **List of neighbors** of vertex s:

```
Iterator<Node> neighborsOfS = s.getNeighborNodeIterator();
```

- **verify if a link exists** between vertices s and t :

`s.hasEdgeBetween(t);` returns `true` if a link exists `false` otherwise

- for obtaining an iterator on **outgoing arcs** of vertex s

```
s.getLeavingEdgeIterator();
```

- for obtaining the **the other extremity of a link** e knowing the other extremity, s, of this link:

```
Node t = e.getOpposite(s);
```

- for obtaining the **edge** between two vertices u and v

```
if(u.hasEdgeBetween(v)) Edge e = u.getEdgeBetween(v);
```

7 Other Useful Methods

- **random choice** of a vertex of graph g:

```
import org.graphstream.algorithm.Toolkit;

Node randomVertex = Toolkit.randomNode(g);
```

- **random neighbor** of a vertex v:

Edge e = (Edge)v.getEnteringEdgeSet().toArray()[alea.nextInt(v.getDegree())];

```
Node neighbor = e.getOpposite(v);
```