yFiles Class Library Overview

https://www.yworks.com/

Panagiotopoulos Dionisis

[October 2020]

The y-Files library

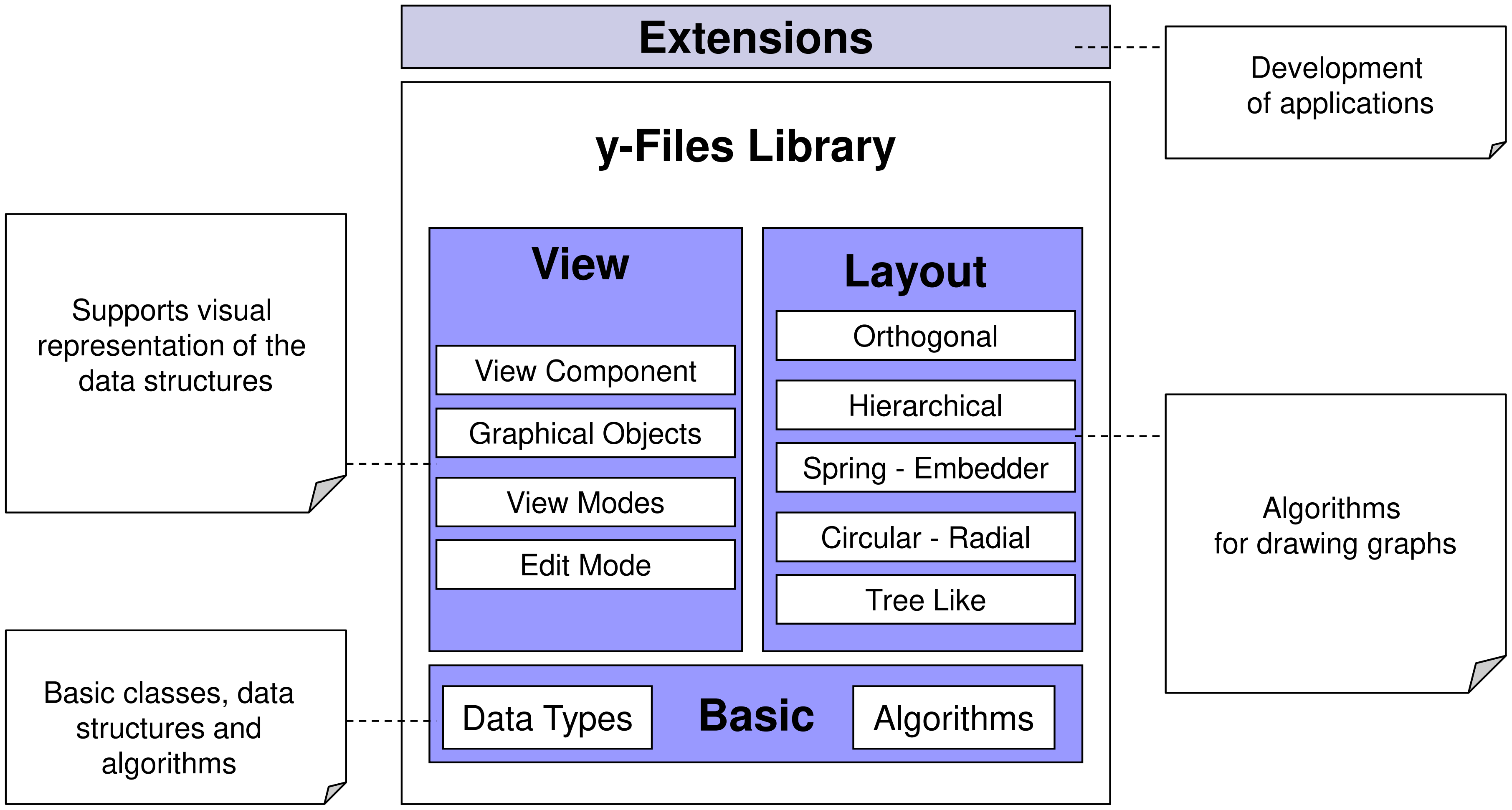
- The y-Files library is designed and developed entirely in Java and may be used by any Java application independently of the operating system:
  - Linux
  - Solaris
  - MacOS X
  - Microsoft Windows

- The latest version of the y-Files library is 3.5 and requires Java 8 (or higher version).
- This is a commercial product distributed by yWorks GmbH.

Useful links

- Initializing IDE (https://docs.yworks.com/yfilesjava/doc/api/#/dguide/getting_started-ide)
  - **IntelliJ IDEA 2021**
  - **Eclipse 2021**
  - **NetBeans 12**
- Documentation of the library (https://docs.yworks.com/yfilesjava/doc/api/#/home)

Structure of y-Files library



The package yfiles.algorithms

- Contains basic classes:
  - `Node – Edge`
  - `Graph`
  - `YList – NodeList – EdgeList`

The class yfiles.algorithms.Node

- Constructor:
  - `protected Node(Graph g)`
- Methods:
  - `int degree(), outDegree(), inDegree()`
  - `IEdgeCursor getEdgeCursor()`
  - `IEdgeCursor getOutEdgeCursor(), getInEdgeCursor()`
  - `INodeCursor getNeighborCursor()`
  - `Edge getEdgeFrom(Node source)`
  - `int index()`

The class yfiles.algorithms.Edge

- Constructor:
  - `protected Edge(Graph g, Node v, Edge e1, Node w, Edge e2, GEI d1, GEI d2)`
- Methods:
  - `boolean isSelfLoop()`
  - `Node source()`
  - `Node target()`
  - `Node opposite(Node v)`
  - `int index()`

`GEI=GraphElementInsertion: "BEFORE"/ "AFTER"`

The class yfiles.algorithms.Graph

- Constructor:
  - `Graph()`
  - `Graph(Graph graph)`
- Methods:
  - `boolean isEmpty()`
  - `boolean contains(Edge e), contains(Node v)`
  - `boolean containsEdge(Node v1, Node v2)`

The class yfiles.algorithms.Graph

- Methods:
  - `Edge createEdge(Node v, Node w)`
  - `Node createNode()`
  - `IEdgeCursor getEdgeCursor()`
  - `INodeCursor getNodeCursor()`
  - `Edge[] getEdgeArray()`
  - `Node[] getNodeArray()`
  - `void removeEdge(Edge e), removeNode(Node v)`
  - `int nodeCount(), edgeCount()`

Demo

```
Graph graph = new Graph();
Node tmpNodes[] = new Node[5];
for(int i = 0; i < 5; i++) {
    tmpNodes[i] = graph.createNode();
}
for(int i = 0; i < 5; i++) {
    for(int j = i+1; j < 5; j++) {
        graph.createEdge(tmpNodes[i],tmpNodes[j]);
    }
}
```

The classes INodeCursor & IEdgeCursor

```
for(INodeCursor nc = graph.getNodeCursor(); nc.ok(); nc.next())
{
   Node node = nc.node();
   System.out.println(node);
}
```

```
for(IEdgeCursor ec = graph. getEdgeCursor(); ec.ok(); c.next())
{
   Edge edge = ec.edge();
   System.out.println(edge);
}
```

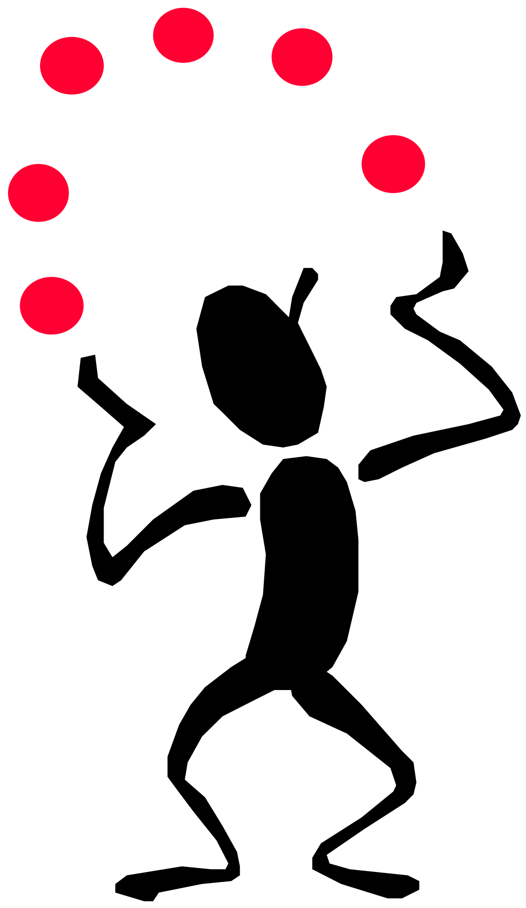
Iterating nodes or edges

```
for(IEdgeCursor ec = graph. getEdgeCursor(); ec.ok(); ec.next())
{
    graph.reverseEdge(ec.edge());
}
```

```
for(INodeCursor nc = graph. getNodeCursor(); nc.ok(); nc.next())
{
    if(nc.node().degree() < 2)
        graph.removeNode(nc.node());
}
```

Demo

- GraphDemo
- RandomTreeGenerator



The classes INodeMap & IEdgeMap

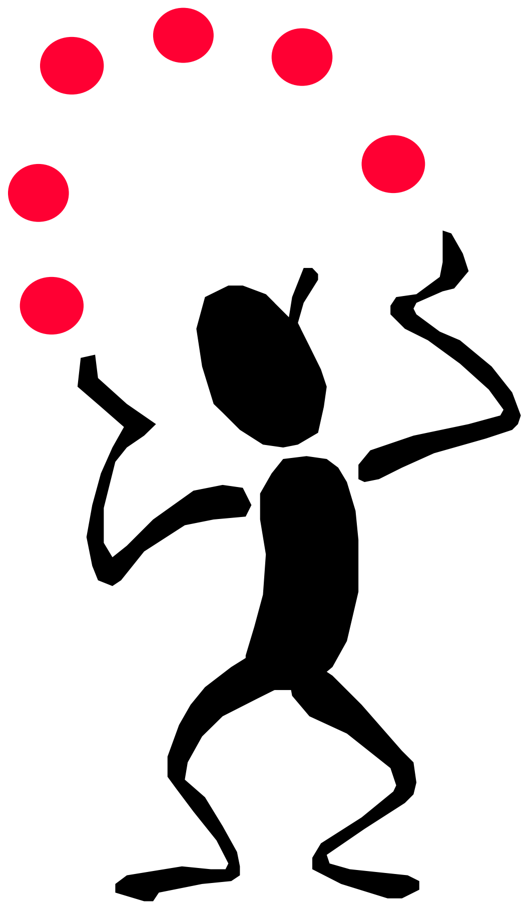
- Give access to custom data related to a node/edge:
  - `INodeMap nodeMap = graph.createNodeMap();`
  - `IEdgeMap edgeMap = graph.createEdgeMap();`
- Methods:
  - `void set(Object element, Object value)`
  - `void setBool(Object  element, boolean value)`
  - `void setDouble(Object  element, double value)`
  - `void setInt(Object  element, boolean value)`
  - `Object get(Object element)`
  - `boolean getBool(Object element)`
  - `int getInt(Object element)`
  - `double getDouble(Object element)`

Example of weighted graph

```
Graph graph = new Graph();
for(int i = 0; i < 10; i++)
{
    graph.createNode();
}
INodeMap map = graph.createNodeMap();
for(INodeCursor nc = graph. getNodeCursor(); nc.ok(); c.next())
{
    map.setInt(nc.node(), nc.node().index());
}
for(INodeCursor nc = graph. getNodeCursor(); nc.ok();nc.next())
{
     Node v = nc.node();
    System.out.println(v+" Weight: "+map.getInt(v));
}
```

Demo

- NodeMapTest
- ExtendedGraph



The package yfiles.algorithms

- Contains classes that implement standard graph algorithms, e.g.
  - Finding cycles in a graph.
  - Finding connected components.
  - Computing Spanning Trees.
  - Etc

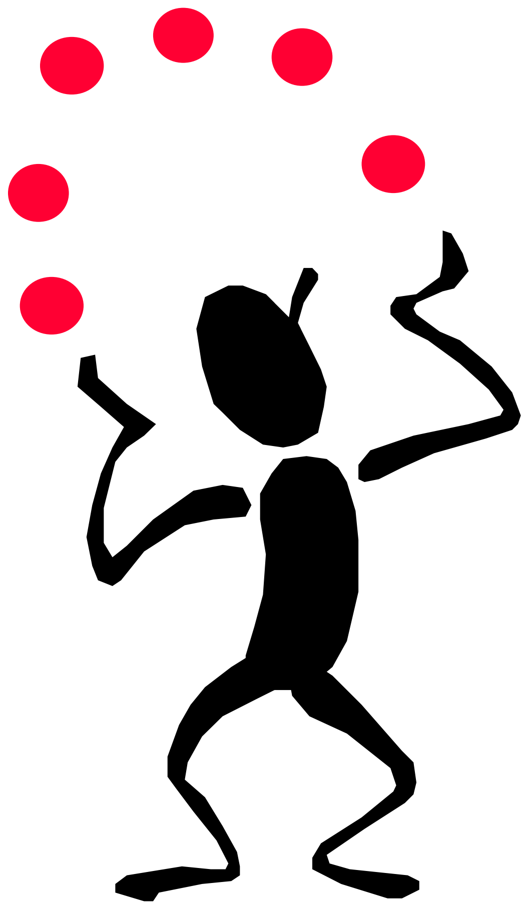
- Static Methods

Basic classes of the package yfiles.algorithms

- Cycles
  `EdgeList findCycle(Graph g, boolean directed)`
- GraphChecker
  `boolean isPlanar(Graph g), isTree(Graph graph)`
- GraphConnectivity
  `boolean isConnected(Graph g), EdgeList makeConnected(Graph g)`
- NetworkFlows
  ```
  int calcMaxFlow(Graph g, Node t, Node s, IDataProvider d,
  IEdgeMap flow)
  ```
- Paths
  `EdgeList findLongestPath(Graph g)`
- Trees
  `boolean isTree(Graph g), isRootedTree(Graph g)`

Demo

- AlgoDemo
- ConnectedDemo



The package yfiles.view

- Important classes:
  - `GraphComponent`
- Uses classes from the package yfiles.graph
  - `IGraph`
  - `IEdge`
  - `INode`

The package yfiles.graphml

- Contains classes:
  - For saving a graph in GraphML - format file.
  - For loading a graph saved in a GraphML file.
- Basic class:
  - `GraphMLIOHandler`

Saving a graph

```
public static void saveGraph(GraphComponent gc,
   String fileName)
{
   if (!fileName.endsWith(".graphml")) {
       fileName = fileName + ".graphml";
   }
   try {
       gc. exportToGraphML(fileName);
   }
   catch (java.io.IOException ioe) {
       //Do something…
   }
}
```

Loading a graph

```
public static void loadGraph(GraphComponent gc,
    String fileName)
{
    if (name.endsWith(".graphml")) {
        try {
            gc. importFromGraphML(fileName);
        }
        catch (java.io.IOException ioe) {
            //Do something…
        }
    }
}
```

Converting IGraph to Graph

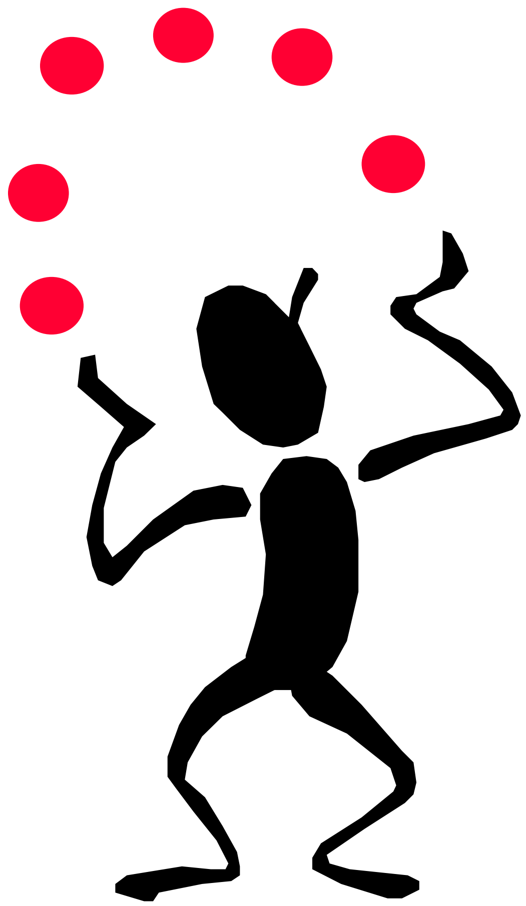
- Class yfiles.layout.YGraphAdapter
  - Constructor: `YGraphAdapter(IGraph)`
  - Methods:
    - `Graph getYGraph()`
    - `IGraph getOriginalGraph()`
    - `INode getOriginalNode(Node v)`
    - `IEdge getOriginalEdge(Edge e)`

The class IGraph

- Methods:
  - `setStyle(IEdge, IEdgeStyle)`
  - `setStyle(INode, INodeStyle)`
  - `setNodeCenter(INode v, PointD position)`
  - `setNodeLayout(INode node, RectD layout)`

Demo

- VisualDemo



The class GraphComponent

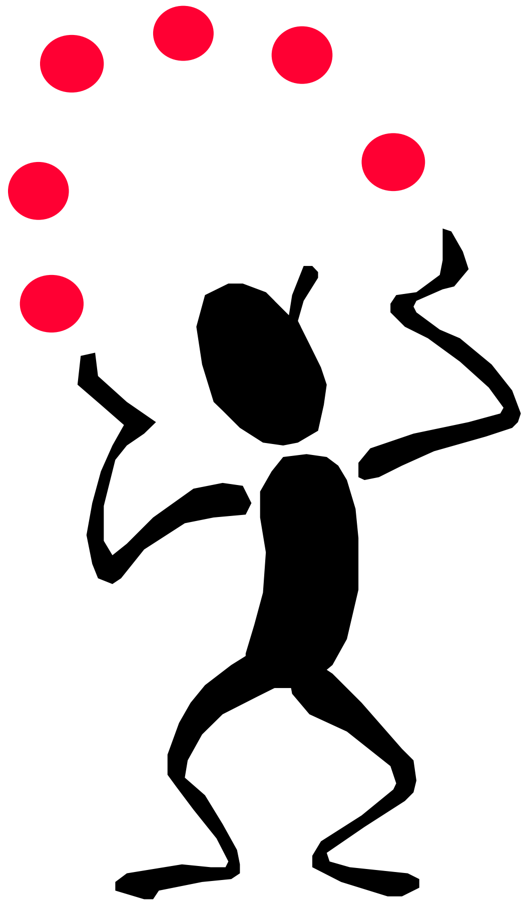
- `JComponent` for drawing graphs

Example

```
public class SimpleFrame extends JFrame
{
  public SimpleFrame()
  {
    super("Simple Frame");
    setSize(600,600);
    setLayout(new BorderLayout());
    GraphComponent gc = new GraphComponent();
    //… read or create a graph
    add(gc, BorderLayout.CENTER);
    setVisible(true);
  }
}
```

Demo

- SimpleDemo



Zoom in / Zoom Out

- The class `GraphComponent` contains methods that can change the Zoom level on this Panel

- Example
  - `gc.setZoom(new PointD(0.0,0.0), 1.2)`
  - `gc.setZoom(0.8*gc.getZoom())`
  - `gc.fitGraphBounds()`

User Interaction

- The class `GraphComponent` allows editing the graph by the user
- Example

```
gc.setInputMode(new GraphEditorInputMode());
```

Demo

- SimpleFrame