

Implementing algorithms using GraphBench

```
GraphBench 20030830
[MST.java]
new open reload save saveas

// Java

use variable graphModel to access GraphModel methods
(e.g. graphModel.getVertices())

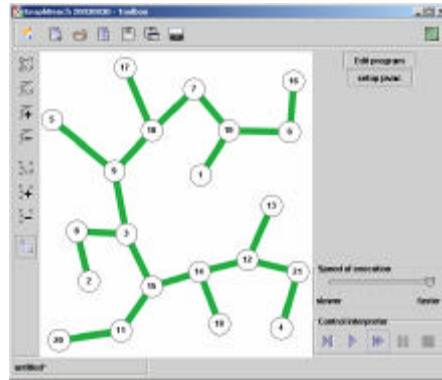
all vertices and edges are instances of AlgorithmVertex and
and can be casted accordingly

Goto http://www.tedu.ethz.ch/brsendla/graphbench
for Java API
*****

import ch.ethz.mab.graph.*;
import ch.ethz.mab.graphbench.toolbox.*;

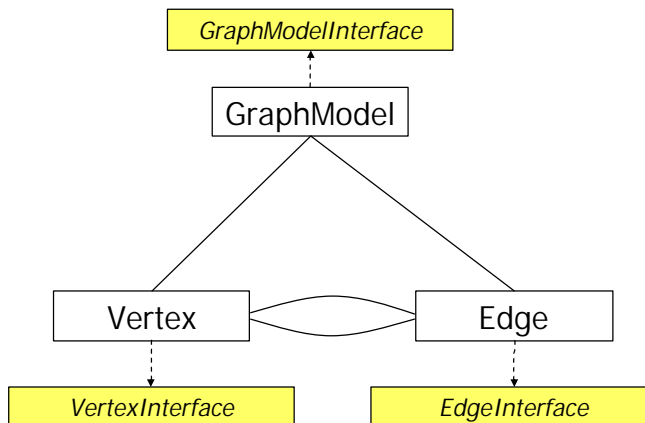
import java.util.ArrayList;

Compile
Executed: javac -classpath "C:\Dokumente und Einstellungen\brsendla\De
```



The graph data structure

- package ch.ethz.mab.graph



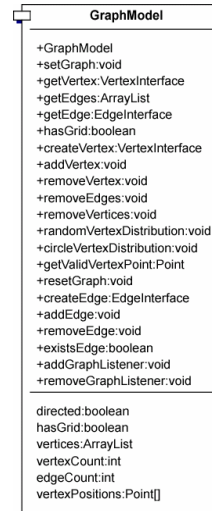
GraphModel

- **getEdgeCount():int**
- **getVertexCount():int**

- **getEdge(int):EdgeInterface**
- **getVertex(int):VertexInterface**

- **createEdge(VertexInterface, VertexInterface):EdgeInterface**
- **createVertex(Point):VertexInterface**

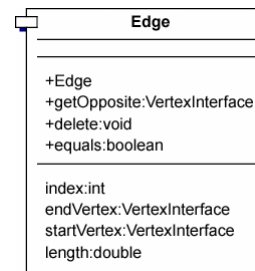
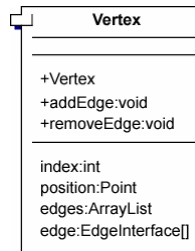
- **addEdge(EdgeInterface):void**
- **addVertex(VertexInterface):void**



Vertex and Edge

- **getEdge(int):EdgeInterface**
- **getEdges():ArrayList**
- **getIndex():int**

- **getLength():int**
- **getStartVertex():VertexInterface**
- **getEndVertex():VertexInterface**
- **getOpposite(VertexInterface):VertexInterface**
- **getIndex():int**



Algorithm execution

Saving algorithm data:

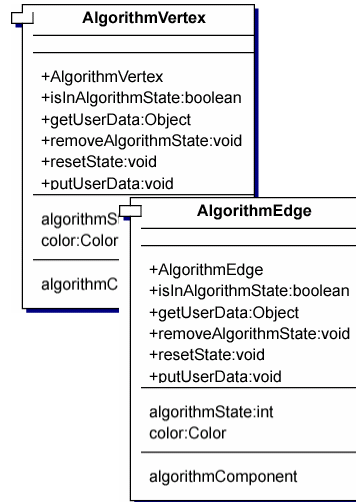
- **putUserData**(Object, Object): void
- **getUserData**(Object): Object

Visualizing algorithm execution:

- **setAlgorithmState**(int): void
- **removeAlgorithmState**(int): void
- **resetState**(): void
- **setColor**(Color): void

AlgorithmConstants:

- QUEUED / ACTIVE / PROCESSED
- VALID / INVALID



The Java template

```
import ch.ethz.mab.graph.*;
import ch.ethz.mab.graphbench.toolbox.*;

public class MyAlgorithm extends GraphAlgorithm {

    public void executeAlgorithm(){

        // insert program code
        // access GraphModel using variable graphModel

    }
}
```

Minimal import statements required

Required superclass of all algorithms

Method called when algorithm is started

Reference to GraphModel uses AlgorithmVertex and AlgorithmEdge *use casting!*

MST: „Simple“ UnionFind

```
class UnionFind{

    protected ArrayList[] sets;

    public UnionFind(ArrayList elements){
        sets = new ArrayList[elements.size()];
        for(int i=0; i < sets.length; i++){
            sets[i] = new ArrayList();
            sets[i].add(elements.get(i));
        }
    }

    public int find(Object element){
        for(int i=0; i < sets.length; i++){
            if(sets[i].contains(element)){
                return i;
            }
        }
        return -1;
    }

    public void union(int first, int second){
        sets[first].addAll(sets[second]);
        sets[second] = new ArrayList();
    }
}
```

MST: Sorting the edges

```
protected void sortEdges(){

    sortedEdges = new ArrayList();

    for(int i=0; i < graphModel.getEdgeCount(); i++){
        AlgorithmEdge edge = (AlgorithmEdge)graphModel.getEdge(i);
        edge.setAlgorithmState(AlgorithmConstants.ACTIVE);
        boolean added = false;
        for(int k=0; (k < sortedEdges.size()) && (!added); k++){
            AlgorithmEdge currentEdge = (AlgorithmEdge)sortedEdges.get(k);
            if(edge.getLength() < currentEdge.getLength()){
                sortedEdges.add(k, edge);
                added = true;
            }
        }
        if(!added){
            sortedEdges.add(edge);
        }
        edge.resetState();
    }
}
```