Object Oriented Programming with

Java - Advanced Course

**Project Documentation**

Samuel Negash, Sebastian Aybar, Bilal Dardour

**Contents**

**1** **Introduction** **1**

**2** **Group Members** **1**

**3** **Project Documentation** **2**

**4** **Milestones** **3**

4.1 Milestone 1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

4.2 Milestone 2 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4

**1**

**Introduction**

In this document we want to give an overview about our project in our Object Oriented Programming with Java study module. The project is about reading communication network information from a network model file into appropriate Java data structures. Processing the network model data and acquisition of the necessary information and outputting the results of the processing.

**2** **Group Members**

|  |  |
| --- | --- |
| **Name** | **Main Responsibilities** |
| Samuel Negash | Reading file/Output of Information |
| Sebastian Aybar | Object-oriented Design/Algorithms |
| Bilal Dardour | Object-oriented Design/Algorithms |

**3**

**Project Documentation**

**Technical Description:**

We read the communication network information from a network model file with the DOM parser, and create GraphNode and GrapEdge objects.

With these two complex datatypse we create a Graph in the Graph class.

Also in the Graph Class we implemented algorithms for shortest path connectivity, centrality measure and diameter.

**Already Implemented:**

-ReadXML class (Parser) // here we used the DOM Parser

-GraphNode class // which is the class for a Node with his attributes and methods

-GraphEdge class // which is the class for an Edge with his attributes and methods

-Main class //

-Triplet//we crate the datatype “Triplet” to use a Array List with “Triplet” as generics.

**Mostly Implemented:**

- Graph class //Shortest Path and Connectivity are already implemented and the Diameter and Centrality Measure Algorithm are missing.

**Description:**

**-ReadXML class:**

It has following attributes ->

public static String filepath; // it’s the filepath of the xml file which will be analysed public static ArrayList<GraphNode> Nodes //ArrayList of the GraphNodes

public static int edges\_counter = 0; // needed to count the number of edges

It has following methods ->

public static Document inputFile(); // xml-File will be converted to a DOM Document public static void getNodes(); // uses the DOM Document to get all Nodes of it public static void getEdges(); // uses the DOM Document to get all Edges of it

// we will just try to catch all Neighbours of the Nodes

-**GraphNode class(mostly changed):**

It has following attributes->

private String name;

private num;

private boolean visited;

private LinkedList<GraphEdge> edges;

It has following methods->

-Getter and Setter for Atrubutes

-Overloaded Standardconstructor

**GraphEdge class (completely new):**

It has the following attribute->

private GraphNode source;

private GraphNode destination;

private double weight;

It has the following methods->

-Getter and Setter for Atrubutes

-compareto method for deciding which of two edges has less weight

-**Graph class(completely new):**

It has following attribute->

-private Set<GraphNode> nodes // for saving all nodes in a set

It has following methods till now ->

-Standardconstructor

-get NodesandEdgesfromXML

-intoGraphNode

-addAllEdges

-addEdge

-addEdgeHelper

-printEdges

-hasEdge

-resetNodesvisited

-shortestpath

-checkConnectivity

-closetrechableunvisited

-**Main class:**

-outputstream

-output of Number of Nodes

-output of Number of Edges

-output of connectivity

-switch case for shortest path

**Milestone 1:**

**-**We completed the reading of the xml file and the data structures of our programm which fits for our planned algorithm.

-Samuel Negash did the reading of the XML-file, while Sebastian Aybar and Bilal Dardour worked on the GraphNode class according to the needs of algorithms and designed the UML-diagram.

-It is planned for the next milestone to complete all three algorithms and the output

Problemes:

- First Problem what we then solved:

- First Samuel tried to parse the code on his own, which somehow did not work

out like planned. Then he tried 2 external parser libraries, the SAX and the DOM

Parser. Both worked but choosing the right Parser was the problem.

- He choosed the DOM parser because it just parses the whole document first and

afterwards we can work with dat.

- In contrast the SAX parser would imply us to handle the parsed events directly

with each event, which is a good way to parse the xml file too, but Samuel just preferred using the DOM Parser.

- Especially in our project we are not working with very big files, that’s why using

the DOM Parser is no Problem.

- If we would work with big files, it would be way smarter to use the SAX parser

- Second Problem which is unsolved:

- to get the Neighbours (with their weight in a tupel) of every Node out of the

Edges -> Neighbours will be saved as null which isn’t planned

**Milestone 2:**

* We revised our datatstructure for the algorithms which are completely impleted except the centrality measure and the diameter of the graph. The reason was that we met some difficulties with the previous build of datastructure while contstructing our algorithms. The new datastructur is more customized for the algorithms. This also solved the second problem from the first milestone. Our program is able to be started from the command line and also the filename of the graphml file can be inputed from the user in the command line. The next step is to implement a logging class and we have to surround our sourcecode with errorhandling in every case we need. Also we need Diameter and Centrality Measure and the Outputfile. Changes in each Class are mentioned in brackets at the class names.