Gebze Technical University - Computer Engineering Department
CSE443 - Object Oriented Analysis and Design
Fall 2018-2019
Homework 4

Rule 1: no plagiarism (from colleagues or other sources). Detected cases of plagiarism will lead to a significant penalty of your course grade at the end of the semester.

Rule 2: **no late submissions! Even if it is late by one minute, it will be ignored**. Learning to plan your schedule according to deadlines is part of your education and an invaluable professional asset.

What to submit: a) the source code of your project *fully documented (with javadoc)*, b) a nicely formatted pdf report of your design decision explanations and class diagrams and c) an executable demo that fully illustrates your program's capabilities *whenever code is requested*.

****

Question 1: You are a given a finite state machine in the attached digital image. Use the **state design pattern** to implement it in Java and provide a main method where you test all of its transitions. (**45 points**)

Question 2: Seeing how everybody is using graphs one way or another, you have decided to create a company ("BulutÇizge A.Ş.") that will provide graph oriented services to the public. In more detail, clients will contact your server, send a graph object to it (+ some parameters), and ask for some computation with it, that your server will realize and return its result(s) to the client. For some unknown reason you have chosen **java RMI** to implement the communication layer between the server and the clients. Your server provides two services:
a) Given a graph it returns the minimum spanning tree (assume that the graph always contains such a tree if this method is called)
b) The incidence matrix corresponding to the given graph.

What is required of you, are the following:
1) Design and implement the graph data structure, graph algorithms and nodes. Remember that they can represent anything, e.g. cities, people, etc. You want your service to work with ANY kind of data. (**30 points**)

2) The client, who has some data, will construct her graph using your interface, (so that it'll be compatible with your server), and call the corresponding method and wait for the output, in order to show it on screen (**15 points**) (a GUI is optional, +**20 points**).

3) The server must be waiting at all times for a new method invocation, and have the necessary graph processing methods ready to do the job. Whenever a new request comes it should print on screen which method is called at what time, and how long it took to complete it. (**30 points**)

4) Free trials are over. Now you are going pro, woohoo!! From now on, any client that wants to use your services must pay for it. In order to achieve this, assume that every client has an account on your server containing a credit amount that allows them to make a certain number of remote method calls. For instance client X has a credit of 100, meaning that she can make 100 remote calls (Does every method cost the same? That's up to you.). If her credit is zero, she'll receive an error/exception. Make sure clients authenticate themselves with every remote method call, and decrease their credit accordingly. (**40 points**) (Optional: implement account creation and credit loading also through RMI, (for instance by providing their credit card information) +**20 points**).

Make sure your server is thread-safe when handling credits, as tens (hundreds?) of remote method calls may happen at the same time.

5) Make sure you test both services (if possible, on different systems) and provide screenshots that they work.

Note: assume your graphs are **directed and edge-weighted**, and you are free to implement service (a) and with any algorithm you like, except for brute force. It is up to you to model these graphs. Be careful with RMI as the exact procedure depends on your jdk version. Don’t forget the class diagrams.

Good luck.