Technical University of Cluj-Napoca

Laboratory Work – Assignment 3

Queue Simulation

Name: Florea Razvan

Group ID: 30425

Table of Contents

1. Introduction

1.1 Task objective

The objective of this task is rather permissive and is defined as follows: ”Design and implement a simulation application aiming to analyze queuing based systems for determining and minimizing clients’ waiting time”.

Queues are commonly seen both in real world and in the models. The main objective of a queue is to provide a place for a "client" to wait before receiving a "service". The management of queue based systems is interested in minimizing the time amount its "clients" are waiting in queues. One way to minimize the waiting time is to add more servers, i.e. more queues in the system (each queue is considered as having an associated processor) but this approach increases the costs of the supplier. When a new server is added the waiting clients will be evenly distributed to all current available queues.

The application should simulate a series of clients arriving for service, entering queues, waiting, being served and finally leaving the queue. It tracks the time the clients spend waiting in queues and outputs the average waiting time. To calculate waiting time we need to know the arrival time, finish time and service time. The arrival time and the service time depend on the individual clients – when they show up and how much service they need. The finish time depends on the number of queues, the number of other clients in the queue and their service needs.

Due to this reason, the students have full flexibility concerning the implementation and use of resources offered by the Java programming language. It is up to everyone’s decision rather to develop the algorithms in a particular way or another. Same for using a Graphic User Interface or not, or modeling in a specific way or not.

1.2 Personal approach

I personally developed a Java application for processing queues depending on the task’s state. The task can be generated, can arrive at the server, can wait, or be served. It’s waiting time is computed depending on the other tasks service time, on it’s arrival time and therefore we can also have its finish time.

2. Problem Description

2.1 Problem analysis

The problem domain can be divided in smaller problems. In this way we only have to deal with problems of a smaller complexity that together form the main problem, the main objective of our application.

The problem can be seen from different perspectives and different points of view. The modeling of the orders, products and users to be recognized as structures, as independent classes, is very important because it is the core of almost every operation and every function that we want out application to perform. The user must not feel neglected and therefore the graphic user interfaces must not be forgotten and even more, they should be user friendly and efficient. The functionality is the back end "thinking process" of our application and is the most important aspect that we have to take into consideration.

2.2 Modeling

Problem modeling is done by means of object oriented programming and establishing classes, objects and relationships between the components of our problem.

2.3 Scenarios

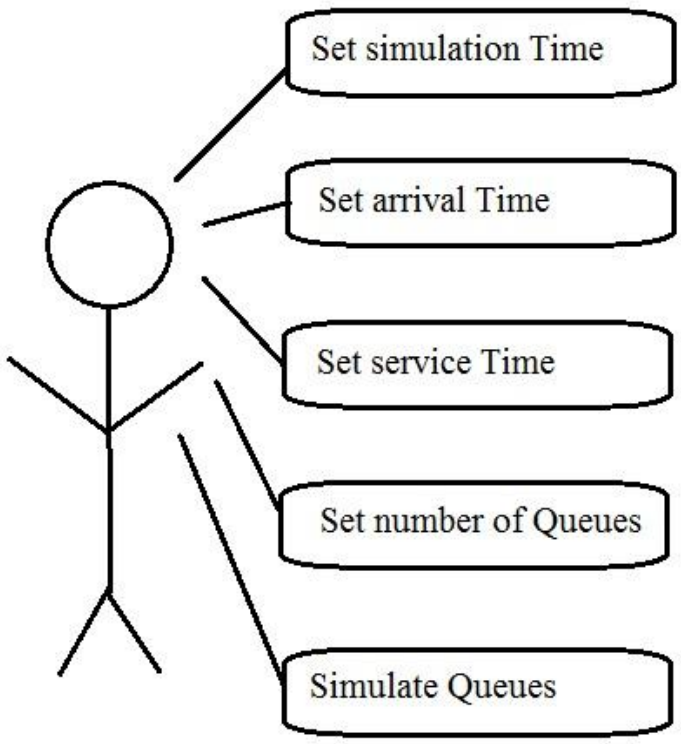
A variety of scenarios have to be taken into consideration. That is, we must be able to predict what the task is ready to be used. That is, the task may be generated, waiting in the queue, or be served by the queue. The User interface lets us see the state of the tasks, their arrival time and the finishing time.

This is why I tried to take into considerations all the possible scenarios that may occur. Therefore I implemented the following functionalities: set the maximum arrival time, as well as the minimum interval, set the maximum service time as well as the maximum, set the simulation interval, display the hour format, set the number of servers, set the simulation speed.

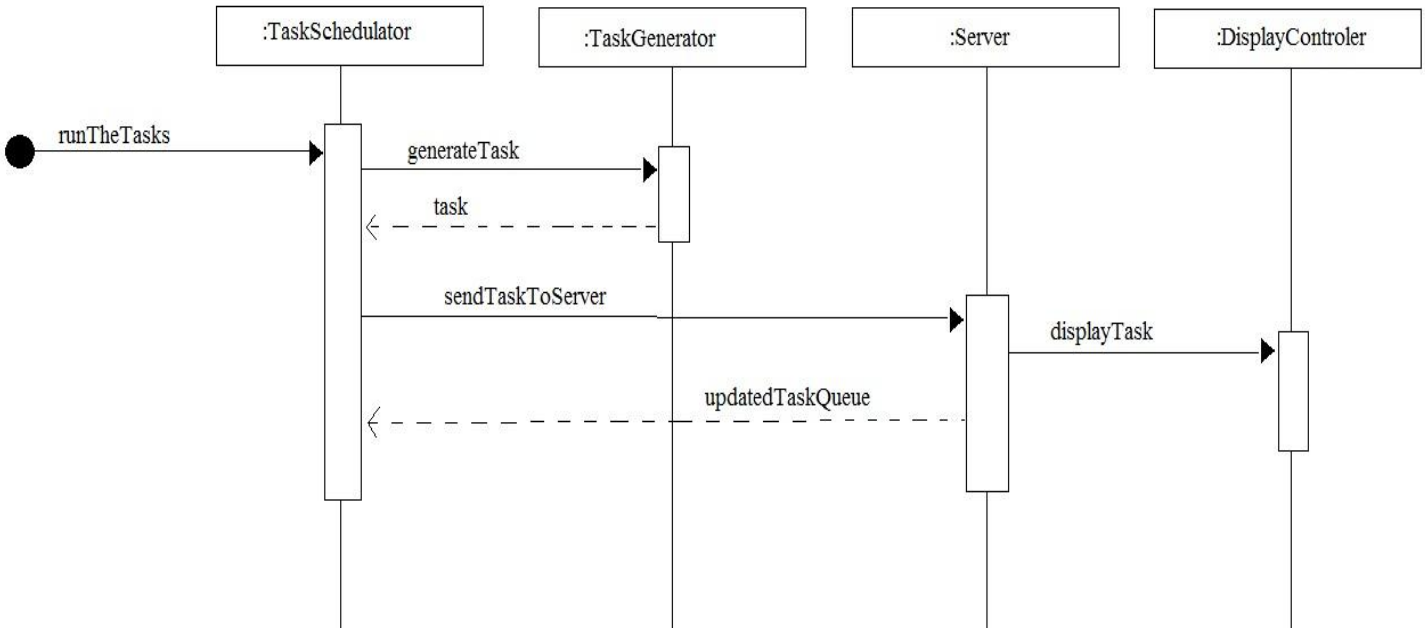
3. Projection

3.1 UML Diagrams

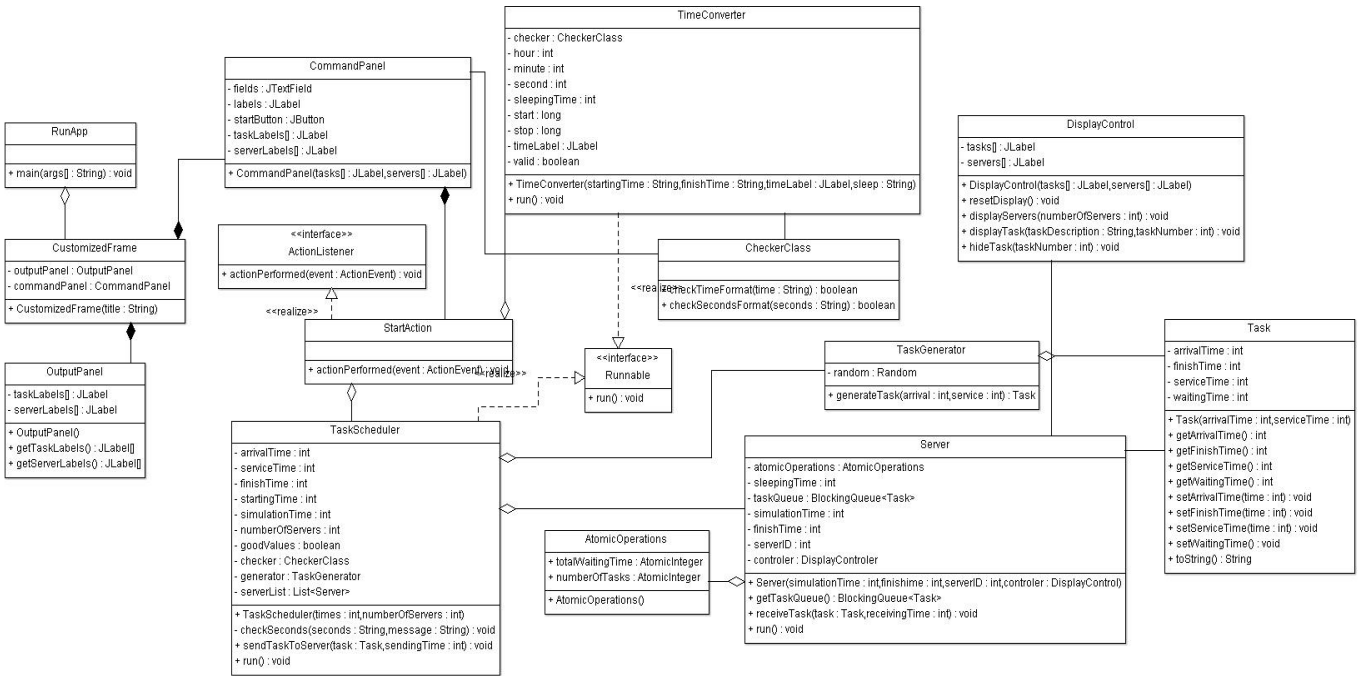
3.1.1 Use Case Diagram



3.1.2 Sequence Diagram



3.1.4 Class Diagram



The diagrams show how the problem is divided in smaller problems and how the modeling was done. The classes communicate between each other and therefore different types of relationships are established.

There are associations between the classes sharing the CheckerClass, Task, Server, aggregations between the classes that use mutual objects, composition for the inner class StartAction which implements ActionListener and is an inner class for the CommandPanel class. The interface Runnable is implemented for the threads in the TimeConverter and TaskScheduler classes. The task Server extends the class Thread so the run method is already created for this one.

For the class diagrams I have used the ArgoUML environment.

3.2 Data Structures

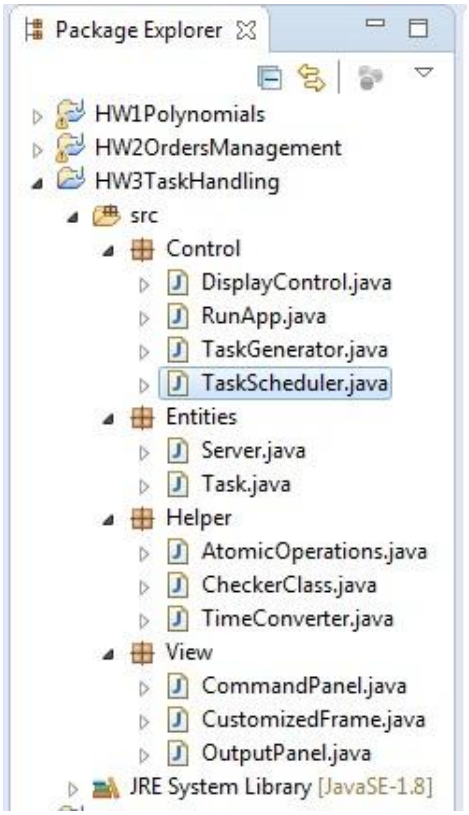
The data structures used are the primitive data types provided by the Java language such as String, integer, double, boolean, font, or other data types such as the customized button, JLabels, JTextFields, JTextAreas, fonts, JFrames.

I also used TreeSet to implement the BinarySearchTrees for sorting and rendering through orders, users or products in the stock.

3.3 Class Projection

The modeling and partitioning of the problem domain is done by means of dividing the problem and data types needed into several classes from which I instantiate objects of that type and use is it in another class. This connection between object of one class and methods of another class is done by means of relationships.

These classes are separated regarding their purpose into different packages. These packages are as follows:



1. Control Package:

The Control package contains the "thinking" done by the applications. That is the functionality of the application and the management of the user input.

It contains the following classes:

- DispalyControl.java;
- TaskGenerator.java;
- TaskScheduler.java;
- RunApp.java

2. Entities:

This package contains the classes responsible with representing the entities. These classes are as follows:

- Server.java;

- Task.java.

3. Helper:

This package contains the classes responsible with checking the user input and the atomic operations needed for thread synchronization. These classes are as follows:

- AtomicOperations.java;

-CheckerClass.java;

- TimeConverter.java

4. View:

This package contains the classes responsible with representing the entities onto the output panel. These classes are as follows:

- CommandPanel.java;

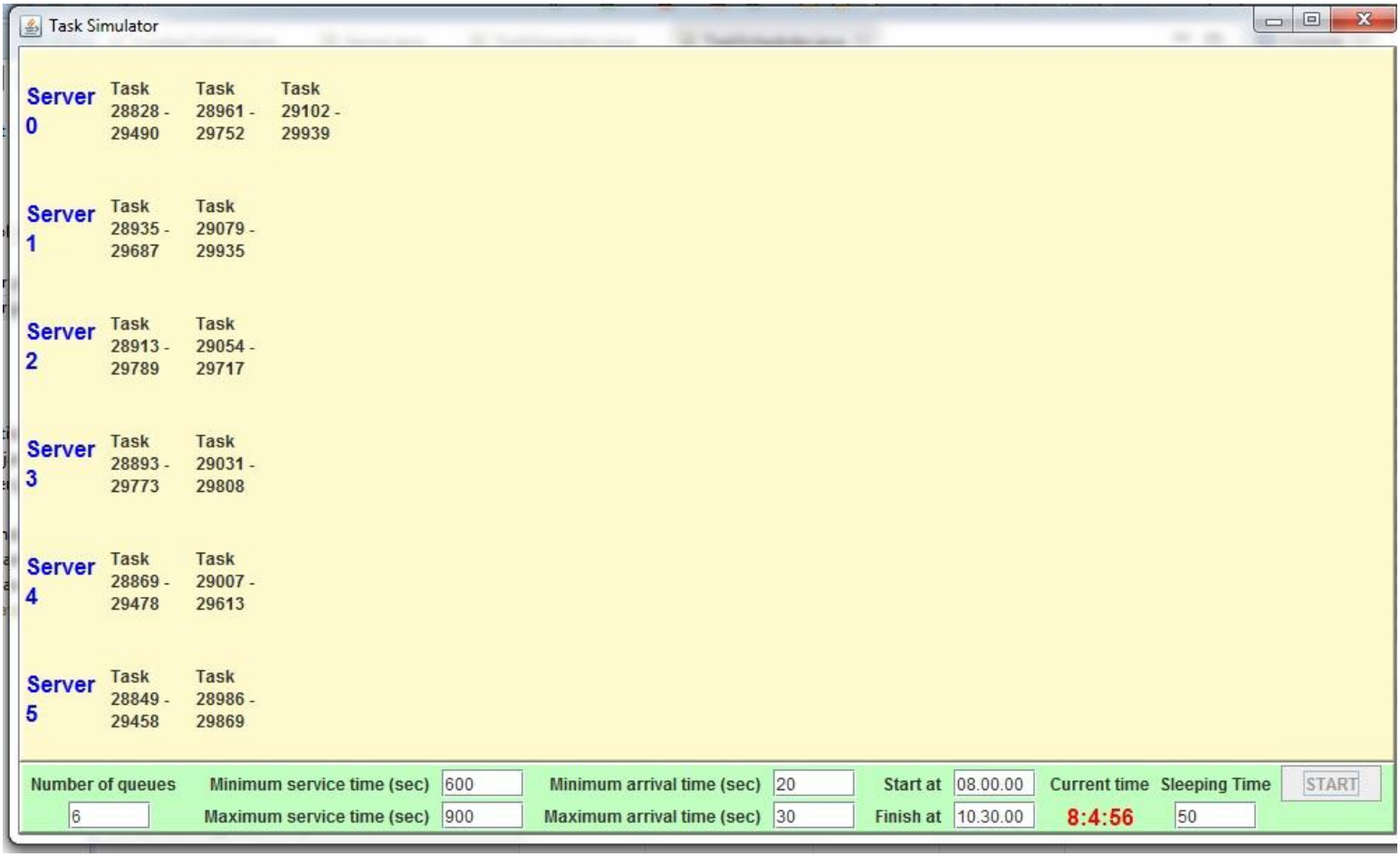
- OutputPanel.java;

- CustomizedFrame.java.

The packages communicate and everything work together in order to satisfy the user and all of his or her requirements. The algorithms are quite simple due to the good managements of resources, data available and modeling the problem domain.

3.4 User Interface

The user interface that I created for this application is the following one:



4. Implementing and Testing

Implementation of the algorithms that I have used and modeling the problem was done by means of the Java programming language and the Integrated Development Environment (IDE) that I used is Eclipse. Even though I have not tested the application on another IDE, it should maintain its portability.

I have covered all the possible cases for user input for the dialogs with the one using the application so that no errors should occur to stop the running time of the applications. For this I used a lot of try - catch blocks to handle different exceptions that may occur.

I have tested all the possible cases that may appear when someone is using the application, and it was proven to be safe for everyone and is easy to understand how to interact with the application and the user interface.

5. Results

The Application is user friendly and very helpful in computing polynomials and different operations on polynomials. Being developed in the Java programming language the final product is highly portable and is able to run on different operating systems as long a java development kit is installed. The application is very straightforward and instructions are always given to the user whenever some wrong data is send as an input and is considered as not being valid.

6. Conclusions

This application meets all of its requirements but it can also be enhanced and developed even more. I have learned some new things and also I increased my java knowledge and experience with this programming language.

7. Bibliography

http://stackoverflow.com/

https://docs.oracle.com/javase/tutorial/uiswing/components/table.html

http://www.color-hex.com/color-names.html

https://docs.oracle.com/javase/7/docs/api/java/io/Serializable.html