Programming Techniques

Assignment 2

Assignment 2

Student:Horatiu-Stefan Spaniol

Group 30422Contents:

1. Problem specification
2. Analyzing the problem(modelling, scenarios, use cases)
3. Design
4. Implementation and Testing
5. Conclusion and future developments
6. Bibliography
7. Problem specification

Objective

Design and implement a simulation application aiming to analyze queuing based systems for determining and minimizing clients’ waiting time.

Description

Queues are commonly used to model real world domains. 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 their "clients" are waiting in queues before they are served. 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 service supplier. When a new server is added the waiting customers 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 customers 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 clients in the queue and their service needs.

Input data:

- Minimum and maximum interval of arriving time between customers;

- Minimum and maximum service time;

- Number of queues;

- Simulation interval;

- Other information you may consider necessary;

Minimal output:

- The average of waiting time, service time and empty queue time for 1, 2 and 3 queues for the simulation interval and for a specified interval (other useful information may be also considered);

- Log of events and main system data;

- Queue evolution;

- Peak hour for the simulation interval;

2. Analyzing the problem(modelling, scenarios, use cases)

2.1 Analysis of the problem

This application should simulate customers waiting to receive a service (e.g. supermarket, bank, etc.). Like in the real world, they have to wait in queues, each queue processing clients simultaneously. The idea is to analyze how many clients can be served in a certain simulation interval, by entering parameters in an intuitive, user-friendly, application graphical interface.

The clients are generated randomly, each having his own service time and arrival time. How many clients are generated, depends on the input values.

The user can set:

* the maximum number of queues available to process customers. At the beginning, only one queue will be open, and, depending on how many clients are generated and their service time, additional queues will be opened so that the clients are put in the smallest queue
* minimum and maximum arrival interval: the delay between customers arriving to receive a service. When generating clients, the arrival time will be chosen randomly between the 2 values using the data type Random
* minimum and maximum service time: the number of minutes needed for a client to be processed, a value is chosen randomly between the minimum and maximum
* simulation interval: the duration in which the customers are being generated

The user can read:

* How many customers were generated during the simulation interval;
* The average service time of the customers;
* The total service time of all customers;
* Additional details and statistics can be read in the detailed log, in the TextArea to the right of the GUI.

2.2 Modelling

Modelling is the activity to make an abstract concept easier to understand by finding its main characteristics and defining some laws which make the given phenomenon quantifiable. This process also includes the decomposition of a complex problem into smaller and simpler problems which will be easier to implement.

The main part of the problem is the creation of the queues, but the most important part is generating the clients randomly, which is done using by extending Thread class, in the Controller class.

2.3 Scenarios

A success scenario looks like this:

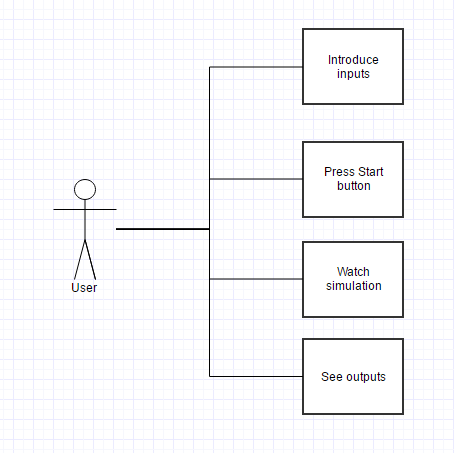
* User successfully inserts the input parameters
* User pushes the "Start" button to start the simulation
* Application displays the output

While a failure scenario looks like this:

* User types wrong data in the input fields
* User pushes the "Start" button attempting to start the simulation
* An error message is displayed
* Another change of inserting/correcting the input data is provided

2.4 Use cases

Use case diagrams are usually referred to as behavior diagrams used to describe a set of actions (use cases) that some system or systems (subject) should or can perform in collaboration with one or more external users of the system (actors).



1. Design

3.1 Design decisions

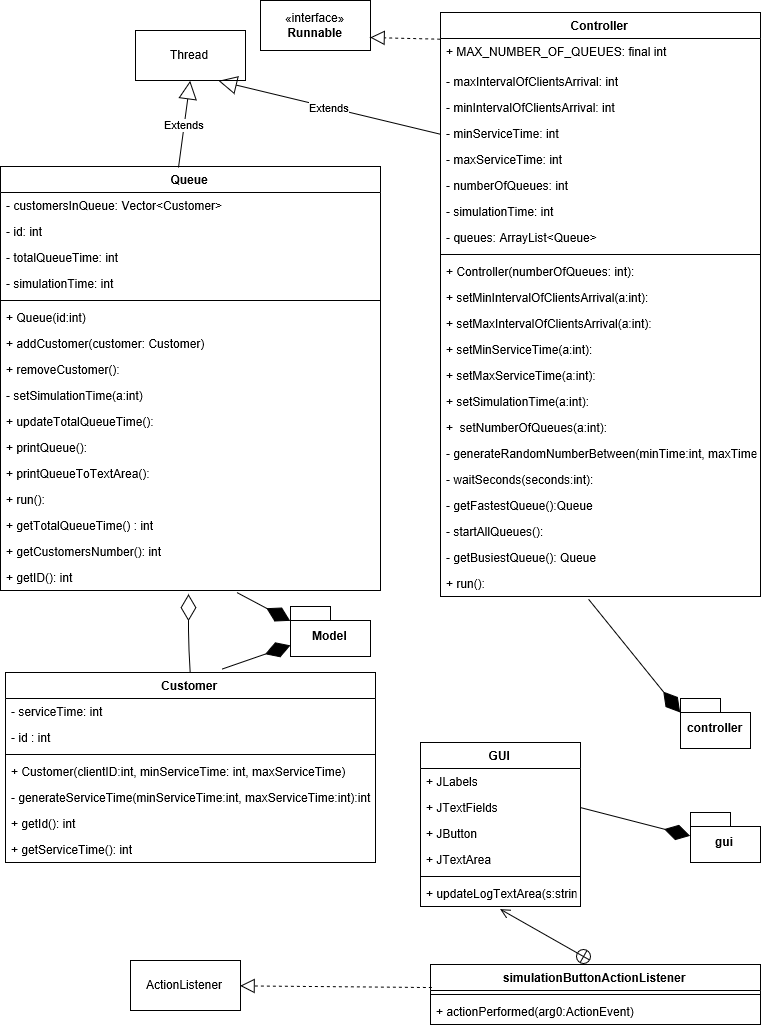
For the design decisions, I have chosen to use an object-oriented approach with a class representing a Customer and one class for representing a Queue. Both classes have been modeled to represent their quote-on-quote visual representation, as in: each Customer has his own service time(how many items need to be scanned at the register) and each Queue has its own list of customers in the actual queue.

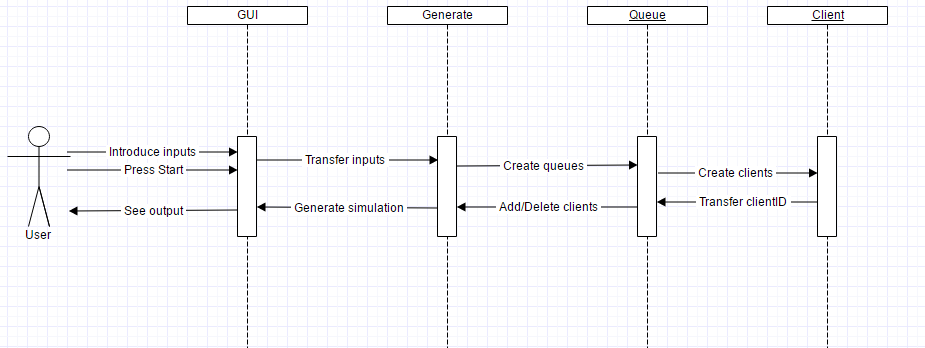
Besides these 2 classes, there is another class called Controller, which is the main class. The Controller class’s purpose is to randomly generate customers and to assign them to the most adequate queue, or the most efficient queue they should be at.

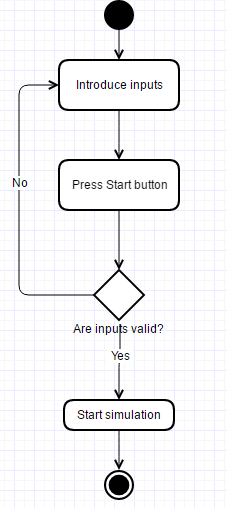
3.2 UML Diagrams

We prepare UML diagrams to understand a system in better and simple way. A single diagram is not enough to cover all aspects of the system. So, UML defines various kinds of diagrams to cover most of the aspects of a system.

Class diagrams are the most common diagrams used in UML. Class diagram consists of classes, interfaces, associations and collaboration. Class diagrams basically represent the object-oriented view of a system which is static in nature.



A sequence diagram is a diagram that shows the interaction between the different objects of the application.



Activity diagram is basically a flow chart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.

3.3 Relationships

For this application, there are some inheritance relationships between the classes Queue and Controller from the class Thread, this is used in the implementation of threads but it is not the only possibility.

Between the class Client and Queue there is a relationship of aggregation because a queue is a vector of clients in our case and clients can exist without a queue.

3.4 Algorithms

Multithreading refers to two or more tasks executing concurrently within a single program. A thread is an independent path of execution within a program. Many threads can run concurrently within a program. Every thread in Java is created and controlled by the Thread class.

Each tread is an instance of class thread. The implementation of threads in a class can be done in 2 ways: the first one by implementing the interface Runnable or by extending Thread (that implements Runnable).

To create a thread, you have to follow the next steps:

* create a class that extends the Thread class
* override the method run() from class Thread
* create an object of type thread using new
* you can start the newly created thread by calling the method start(), also inherited from class Thread

An example of implementing the thread class:

public class Example

{

public static void main(String args[])

{

ThreadClass th = new ThreadClass();

th.start();

System.out.println("Return to main");

}

}

class ThreadClass extends Thread

{

public void run()

{

for(int i=0; i<10; i++)

System.out.println("Step " + i );

System.out.println("Run is done");

}

}

3.5 User Interface

The User Interface was built using Java Swing elements (buttons, panels, frames, labels, textfields,areas). The interface can be called user-friendly because it needs only the most basic of actions and knowledge to make it work, the inputs will be read from textfields that have labels with their individual meaning positioned so that they cannot be mistaken. While starting the application, making it simulate a queue, is done by pressing a button called “start simulation”.

A window will open when running the application. This window will allow the user to introduce the minimum and maximum arriving time, the minimum and maximum service time, the number of queues and the simulation interval in textfields appropriately-named using labels. After introducing the desired data the user will be able to press a button called “Start” to start the simulation of the queue. The Simulation will be able to be visualized in 1 panel below the start button in real time, as there are 5 non-editable text fields that represent customers in each queue, and in the right part of the user interface there is a textarea that prints the log of the whole simulation process, in real-time.

1. Implementation And Testing

4.1 Implementation

The application has been implemented in Java Programming language, using Eclipse. For the GUI, all the buttons, textfields and other components were added by code without any “drag and drop” plugins.

All the GUI components have been added by carefully placing them in adequate panels.

Listeners are placed for specific component to catch events (usually a simple button press), and respond accordingly.

All the implemented classes, including their methods and attributes were documented without comments as I believe the code is clean enough to be understood from their own descriptive name.

4.1 Testing

The testing in this homework is quite different from the previous one where it was required to do the testing using JUnit. There is no mandatory way of doing the tests, so I chose to do them using try-catch structures and by printing errors if any occur during the input or execution part of the application.

Errors appear for leaving certain inputs as their initial values, 0, like the number of queues or the simulation interval (because it would make no sense otherwise). Also, there are some conditions that have to be met, like the maximum should always be bigger than the minimum.

Also my main method of texting was just following the queues visually and keeping tabs on each queue and also printing out the current time each time an operation is done on a queue(adding a customer in the queue, or removing a customer from the queue).

1. Results

Through hard work and intensive testing, I have been able to create an application for simulating for example a supermarket queue-based system.

1. Conclusions and further developments
   1. Conclusions

At the beginning, this assignment looked very hard and hard to approach as our knowledge of threads was practically zero. But as I started to understand how threads work and how multithreading works and to use this concept, it got progressively easier. A big help was the examples given in the laboratory, a simple example that makes you understand the idea of queues and threads and how they work.

I have encountered enough problems with this assignment as at some point I thought the program had the correct behavior, but in fact it had some delays of that were noticeably big. For example, after the program ran for around 15 seconds, a queue should have been added a customer after let’s say 7 seconds and it was only added after 9 seconds. This was quite worrying because the delays should have been not noticeable, in the order of miliseconds. But I have managed to solve this issue.

* 1. Future Developments

There are many improvements that could be made to this application.

There could be introduced a database of products and the number of products to determine the service time.

1. Bibliography

* Example and usage of vectors in java:

<https://docs.oracle.com/javase/7/docs/api/java/util/Vector.html>

* To draw the diagrams:

<https://www.draw.io/>

* Examples of how to create and use threads:

<https://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html>

* Answers to questions regarding JAVA syntax:

<https://stackoverflow.com>

* Great example of a project that shows the implementation of a queue:

<http://users.utcluj.ro/~crisb_pop/TP2017.html>

* Explanations for diagrams:

<https://www.tutorialspoint.com/uml/uml_standard_diagrams.htm>