# Documentation

**Assignment 1**

# Student name: Georgiu Adrian

Group : 30423

# CONTENTS

[1. Assignment Objective 3](file:///C:\Users\jack\Downloads\PT2023_Documentation_Template_EN.doc#_Toc128043139)

[2. Problem Analysis, Modeling, Scenarios, Use Cases 3](file:///C:\Users\jack\Downloads\PT2023_Documentation_Template_EN.doc#_Toc128043140)

[3. Design 3](file:///C:\Users\jack\Downloads\PT2023_Documentation_Template_EN.doc#_Toc128043141)

[4. Implementation 3](file:///C:\Users\jack\Downloads\PT2023_Documentation_Template_EN.doc#_Toc128043142)

[5. Results 3](file:///C:\Users\jack\Downloads\PT2023_Documentation_Template_EN.doc#_Toc128043143)

[6. Conclusions 3](file:///C:\Users\jack\Downloads\PT2023_Documentation_Template_EN.doc#_Toc128043144)

[7. Bibliography 3](file:///C:\Users\jack\Downloads\PT2023_Documentation_Template_EN.doc#_Toc128043145)

# Assignment Objective

Table

Description automatically generated with low confidence

The objective of the assignment is to simulate a shop that has a number of queues and a number of clients that each enter the shop at different times, require a different amount of service time and the idea is to assign each client to a queue to minimize the amount of time waited for each client.

I also write the output and relevant data of the simulation into a file called “simulation.txt”.

# Problem Analysis, Modeling, Scenarios, Use Cases

Problem analysis involves identifying, defining and comprehending a problem in order to find one or more solutions for it. In this regard, my use of OO language confers an advantage. This type of language enables me to work at a higher, more abstract level, free from the constraints of technical details.

The applications will be used by people who might not be programmers, so I need a friendly user interface for it that is easy and convenient.

We know that the simulation requires some inputs that need to be changed at every run (press of the button “Simulate” in this case). The input can be written in the interface I created and attached a photo below.

The output will be written in a timely manner in the output JavaTextArea on the right.

# Design

Diagram

Description automatically generated

Main:

-contains the ViewFrame which shows the user interface

-has the controller which takes the relevant data from the interface and generates all the simulation data

Client:

-contains the data of a client which is its ID, the arrival time at the shop and its service time (time in seconds that he will spend at that shop)

Scheduler:

-contains the list of Servers (a class that will be presented later) which are threads and will generate and start them upon creation (in the constructor)

View:

-is the interface codeGraphical user interface, application, Word

Description automatically generated

Controller:

-is the class that “controls” the viewframe and collects all the relevant data from it

-it has the SimulationManager(presented later) which is a thread and starts the thread upon creation when the button is pressed)

Server:

-is a thread

-is started in the scheduler class

-has a function that adds a client to its queue

-has a function to stop its run

-the function run will continue running until the thread is stopped and if it has a client in its queue it will process that client and remove it from the queue when the client is no longer necessary

SimulationManager:

-has all the relevant data of the simulation (simulation time, minimum/maximum time of arrival

and service time of each generated client, the list of clients, number of Servers)

-when the class is generated the list of random clients is generated inside using the rand function

- and also write the output data into the file “simulation.txt”and the JavATextArea of the output.

-When started it will simulate each second until the simulation time limit the state of the shop (what clients are waiting/have not arrived; what each queue has in its list; and the current time)

# 4.Implementation

Simulation manager:

Text

Description automatically generated

Simulates each second the state of the shop.

It contains a lock that also all the servers have so that no server can eliminate a client from the queue while the manager write the output.

Server:

Text

Description automatically generated

While the server is running it checks if there is a client in its queue and if so it will process it for its service time and then wait for a signal from the manager in order to remove it from the queue

Graphical user interface, text, application

Description automatically generated

# 5.Results

Testing was made using the user interface which I implemented and also the input which was given in the project requirements.

If the inputs are not numbers and are strings the controller will fail to read the input.

If the input does not permit a real time event where all the queues manage to finish each client then the simulation will end at the simulationLimitTime given in the interface and all the threads will be stopped even if clients are still in their queues.

# 6.Conclusions

The hard part of this project was to maintain a synchronization between the server threads and the manager thread and to not delete or write relevant data that should no longer be accurate.

The threads will sleep for a second to simulate the real time events and to show the outputs accordingly.

# 7.Bibliography:

-stackoverflow.com

-geeksforgeeks.com