Logo, company name

Description automatically generated

DOCUMENTATION

**Synchronization of threads**

Programming Techniques

Laboratory - Assignment II

Student: Paul Petruț-Betuel

Group: 30421

# CONTENTS

[1. Assignment Objective 3](file:///C:\Users\paulp\Downloads\Documentatie%20Petrut-%20final.docx#_Toc128043139)

[2. Problem Analysis, Modeling, Scenarios, Use Cases 3](file:///C:\Users\paulp\Downloads\Documentatie%20Petrut-%20final.docx#_Toc128043140)

[3. Design 3](file:///C:\Users\paulp\Downloads\Documentatie%20Petrut-%20final.docx#_Toc128043141)

[4. Implementation 3](file:///C:\Users\paulp\Downloads\Documentatie%20Petrut-%20final.docx#_Toc128043142)

[5. Results 3](file:///C:\Users\paulp\Downloads\Documentatie%20Petrut-%20final.docx#_Toc128043143)

[6. Conclusions 3](file:///C:\Users\paulp\Downloads\Documentatie%20Petrut-%20final.docx#_Toc128043144)

[7. Bibliography 3](file:///C:\Users\paulp\Downloads\Documentatie%20Petrut-%20final.docx#_Toc128043145)

# Assignment Objective

*The topic "Queues Management Application using Threads and Synchronization Mechanisms" refers to the development of an application that simulates the behavior of a queue system, where clients arrive, wait in line, and are served by available servers. The application uses threads to manage the queues and synchronization mechanisms to ensure the correct behavior of the system. The goal is to optimize the use of servers and minimize the waiting time for clients.*

# Problem Analysis, Modeling, Scenarios, Use Cases

*Here we have a simple interface where the user is asked to introduce all the necessary data.* We have given the user a lot of options to choose from, such as: simulation time – how long the simulation will take place, nr. of clients that come along the given schedule etc.

If the user miss to complete a box it will give him a warning and it will not start only after the user has write everything that is needed.

Graphical user interface, application

Description automatically generated

After the users have introduced all the necessary data he can press start and the program should update in real time and let the user take a quick look at the table with the values, but he can see all the steps in the text file named “output.txt”.

*The file should look like this:*

Table

Description automatically generated

# Design

*The following should be presented: OOP design of the application, UML package and class diagrams, used data structures, defined interfaces and used algorithms (if it is the case).*

*In the design phase of this project, we will focus on developing a clear object-oriented design for the application.*

Diagram

Description automatically generated

# Implementation

The Server Java class represents a simulation of a server that processes requests from clients. The class includes several instance variables such as **waitList**, **clientOnThread**, and **queue**, which are all **ArrayBlockingQueue**s. These queues are used to manage clients waiting for service, threads processing those clients, and threads waiting for new clients.

The constructor of the **Server** class takes several parameters that specify the simulation time, the number of queues, the number of clients, and the minimum and maximum arrival and service times for each client. It also takes a **View** parameter that is used to display the output of the simulation.

The **Server** class includes several methods that sort clients based on their arrival time, print the wait list, update a table of clients being served, and write the output of the simulation to a file.

The class includes a nested **MyThread** class that extends **Thread** and is used to simulate the processing of clients by threads. The **Server** class creates a number of **MyThread** objects equal to the number of queues specified in the constructor.

Overall, the Server class provides a framework for simulating the processing of requests by a server and the corresponding wait times experienced by clients.

The MyThread Java class represents a thread that is used in a queue management application. It extends the Thread class and contains several fields and methods for managing a queue of people. The class uses an ArrayBlockingQueue to store people waiting in the queue and an AtomicInteger to keep track of the current time. The run method of the class implements the logic for serving people in the queue and updating the time. The iGetClient method checks if the thread needs to get a new person from the queue and the whoGetClient method determines which thread should get the next person based on the number of clients each thread is currently serving. Overall, the class provides the necessary functionality for managing a queue of people using threads and synchronization mechanisms.

# Conclusions

The Queues Management Application is a Java-based project that provides a solution for managing queues in a multi-threaded environment. The application makes use of threads and synchronization mechanisms to simulate the queuing process and provides a graphical user interface for the user to interact with. The project is designed to handle a large number of clients, with each thread handling a specific queue. The application uses an array blocking queue to manage the waiting clients and ensures that clients are serviced in a first-come, first-served manner. The project demonstrates the use of several programming concepts, including multithreading, synchronization, and graphical user interface design.

# Bibliography

The references that were consulted by the student during the implementation of the homework:

1. Lab teachers explanations
2. <https://dsrl.eu/courses/pt/>
3. <https://www.geeksforgeeks.org/>
4. <https://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html>
5. https://stackoverflow.com/