DOCUMENTATION

ASSIGNMENT *2*

STUDENT NAME: Apostol Horia

GROUP: 30227

# CONTENTS

1.Assignment Objective………………………………………………………………………………………………………………………3

2.Problem Analysis,Modeling,Scenarios,Usecase………………………………………………………………………………..3

3.Design………………………………………………………………………………………………………………………………………………4

4.Implementation………………………………………………………………………………………………………………………………..4

5.Results………………………………………………………………………………………………………………………………………………5

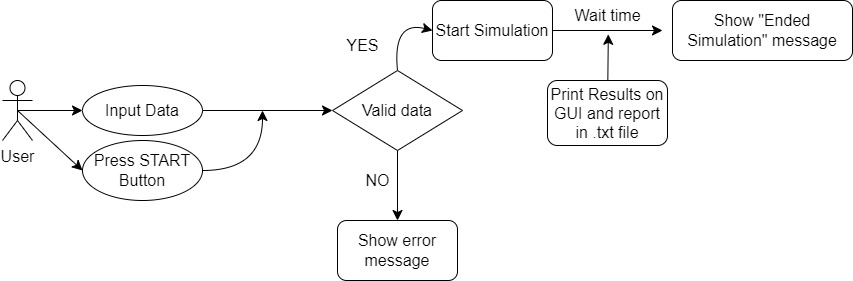
6.Conclusions……………………………………………………………………………………………………………………………………….6

7.Bibliography………………………………………………………………………………………………………………………………………7

1. **Assignment Objective**

The objective of this assignment is to design and implement a queues management application that assigns clients to queues in a way that minimizes their waiting time. The application simulates a series of clients arriving for service, entering queues, waiting, being served, and leaving the queues. The clients are characterized by three parameters: ID, arrival time, and service time. The application tracks the total time spent by every client in the queues and computes the average waiting time. The clients are added to the queue with the minimum waiting time when their arrival time is greater than or equal to the simulation time. The aim is to optimize the queue management system to minimize client waiting time while avoiding the additional cost of adding more servers/queues.

1. **Problem Analysis, Modeling, Scenarios, Use Cases**

****

* 1. *Use case diagram*

The figure represents a use case diagram depicting how a user interacts with the GUI for the queue management application. The user inputs data and presses the start button to initiate the simulation. The application validates the data, and if it passes, the simulation begins. However, if the data does not meet the required format, a new window pops up to inform the user of the issue. The GUI displays the current time and the current state of the queues, but modifications to the queues are limited to 5. If the simulation time exceeds the user-specified limit, a window appears to indicate that the simulation has ended, and a report is generated and saved in the project's file.

1. **Design**

**Diagram, engineering drawing

Description automatically generated**

* 1. **UML Diagram**

# 4.Implementation

The package “Models” contains the classes: Task,Server and Scheduler.

The "Task" class in Java can be thought of as representing a client or request in a queue. The fields "arrivalTime" and "serviceTime" can be used to calculate how long a client has to wait in the queue before being serviced. The "waitTime" field tracks the actual time that the client has to wait before being serviced, and can be updated as the client moves through the queue.

The "Server" class in Java represents a server or queue of clients and provides methods to add tasks to the queue, update the waitingperiod of each task, and execute the tasks one by one. The "run" method is an implementation of the Runnable interface and is used to execute the tasks in the queue. The method runs continuously in a loop, checking for tasks in the queue, and executing them one by one until the "isRunning" flag is set to false. In the "run" method, the server checks for the first task in the queue and then executes it by subtracting 1 from its service time every second until the service time reaches 0, at which point the task is removed from the queue. The waiting period of each task is also updated based on the time it spends waiting in the queue. The method handles exceptions related to thread interruption. Overall, the "Server" class provides a convenient way to model and simulate servers or queues of clients in various programs, making it a useful class in the field of computer science and programming.

The Java "Scheduler" class is used to effectively manage a group of servers that are capable of executing tasks. The class includes a list of "Server" objects that it can manage independently. Upon creation of a new "Scheduler" object, a list of "Server" objects is initialized and a thread is started for each server. To assign a new task to a server, the "dispatchTask" method is used. The Scheduler chooses which server to assign the task to, based on a specific strategy. Currently, the strategy is to assign the task to the server with the shortest waiting period for executing tasks.

The "getQueueSizes" method returns a list of the number of tasks waiting in each server's queue. This can be helpful for monitoring server performance. The "Strategy" interface is an inner interface that provides a blueprint for various strategies that can be used to assign tasks to servers. Currently, the "ConcreteStrategyTime" implementation assigns tasks to the server with the shortest waiting period for executing tasks.

Overall, the "Scheduler" class is a helpful tool for effectively managing and assigning tasks to a group of servers based on a chosen strategy.

The "Controller" class in Java is a part of the graphical user interface (GUI) of a simulation program. It handles user input from the GUI and communicates with the "SimulationManager" to start and manage the simulation.

When a user clicks on the "Start" button in the GUI, the "StartButton" ActionListener in the "Controller" class is triggered. It retrieves the input values for the simulation parameters from the text fields in the GUI and passes them to the "SimulationManager" to start the simulation.

If any of the input values are invalid, such as non-numeric values, the "Controller" class displays an error message in the GUI.

Overall, the "Controller" class plays a crucial role in the user interface of the simulation program, allowing users to start and control the simulation with ease.

The graphical user interface (GUI) contains input fields that are necessary to initiate the simulation. Additionally, there are vacant areas designated for the queues that will be dynamically modified as clients arrive and are served. Furthermore, there is a designated space to display the clients generated during the simulation.

# 5.Results

This simulation models a queuing system that simulates the flow of customers in a store. The system comprises of servers or cashiers that process customers and queues where customers wait for processing. The simulation runs for a specified time, and customers arrive at random intervals and join the shortest queue. Once a customer reaches the front of the queue and a server is available, the customer gets processed for a random time before leaving the system.

We conducted three tests with varying parameters to evaluate the queuing system's performance. In the first test, we simulated a small store with four servers and two queues for 60 seconds. In the second test, we simulated a larger store with 50 servers and five queues for the same time. Finally, in the third test, we simulated a huge store with 1000 servers and 20 queues for 200 seconds.

We recorded the average wait time and service time of customers and the peak hour with the most customers in the system for each test. We saved the results in separate text files, and an overall summary of the results is available in the output.txt file.

The results indicated that as the number of servers and queues increased, the average wait time and service time decreased, and the system could handle more customers. As the system became larger, the peak hour with the most customers shifted towards later hours, indicating a more evenly distributed flow of customers throughout the day.

In summary, the simulation provided us with valuable insights into the performance of queuing systems and their ability to handle different levels of customer demand.

# 6.Conclusions

In conclusion, while the GUI provides a useful view of the dynamic changes within a simulation, creating multiple windows and simulations can greatly enhance the capabilities of the software. By allowing users to run multiple simulations at onceand view them in separate windows, the software can become more versatile and adaptable to a wider range of scenarios. Additionally incorporating features such as the ability to save and load simulation configurations can make the software more convenient and user-friendly. Overall, by continuously striving to improve and expand the functionality of the software, it can become a valuable tool for various fields and applications

# 7.Bibliography

https://dsrl.eu/courses/pt/

# 