DOCUMENTATION

ASSIGNMENT 2

STUDENT NAME: GENGIU ROBERT-CONSTANTIN
GROUP: 30421

CONTENTS

1. Assignment Objective

The assignment's main objective is to create an application for implementing a type of simulator dedicated to analyzing a client-queue management system. Specifically, we are talking about several N clients that require obtaining/performing a service (referred to throughout the implementation as a "Task"), clients who line up in Q queues, wait until it's their turn to be served, and then leave the queue. In addition to the dynamics of tasks in queues over a period, we will also study the average waiting time in the queue to be served, the average service duration for a client, and the "peak hour" of the simulation (the time when the queues have the highest number of clients).

2. Problem Analysis, Modeling, Scenarios, Use Cases

The use of the program involves the user inputting the theoretical data of the simulation, followed by analyzing in real-time the transformations that the queues undergo until the simulation is completed, depending on the chosen strategy. The simulation's peculiarities that the user needs to set are the number of arriving clients, the number of open queues, the maximum time interval for the simulation, the interval corresponding to the arrival time of clients, and the limiting interval for the duration of processing and performing the requested service (service time interval).

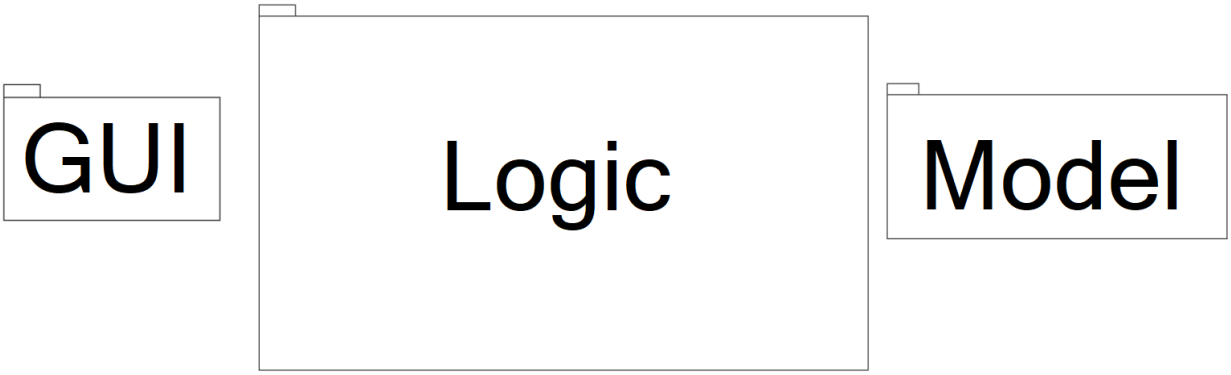
These operations will be performed after pressing the "start" button, with the application being guided by the "shortest time" strategy, meaning that new clients will be automatically placed in the queue with the shortest waiting time. For the correct functioning of the application, the user is responsible for inputting correct initial data.

3. Design

The main data structures used in developing the application were lists, and due to the concurrent nature of the processing mechanism, BlockingQueues. A BlockingQueue is a queue data structure that restricts and blocks the extraction and insertion of elements from it if it is empty or if the maximum limit of elements has been reached. A thread attempting to remove an element from an empty queue will be blocked until another thread inserts an element into that queue. Similarly, the same will happen when inserting an element beyond the maximum limit of possible elements.

Let's imagine we are in a public institution, such as a train station. The use of BlockingQueues takes place through the Server class. An object of type Server represents the virtual representation of a real-life counter, in front of which a queue of clients wishing to be served is formed. Clients are represented by objects of type Task, which are associated with an "arrival time at the institution" and the time required to fulfill the request (e.g., the duration from placing the order to printing the train ticket). These two periods will be randomly generated for each client to translate as realistically as possible the dynamics of requests in a real institution.

UML Diagram of Packages



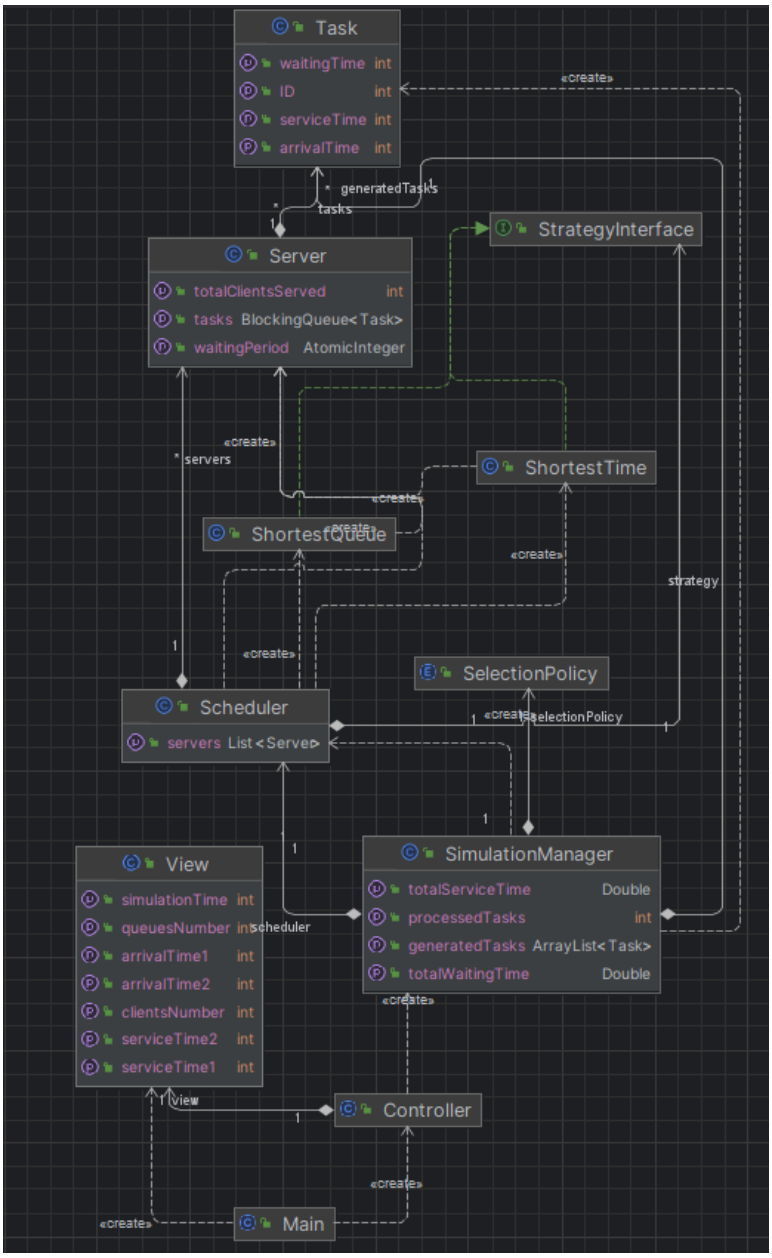
GUI -> Contains the View and Controller class which are representing the interface of the application and how you are controlling the application

LOGIC -> Contains the next classes: Scheduler, Shortest Queue, Shortest Time, Simulation Manager; Selection Policy (Enum), Strategy Interface (Interface)

These classes are responsible with the logic behind the application

MODEL -> Contains the Server and Task class which are responsible for describing the clients and the dynamic simulation of the queues

UML Class Diagram



4. Implementation

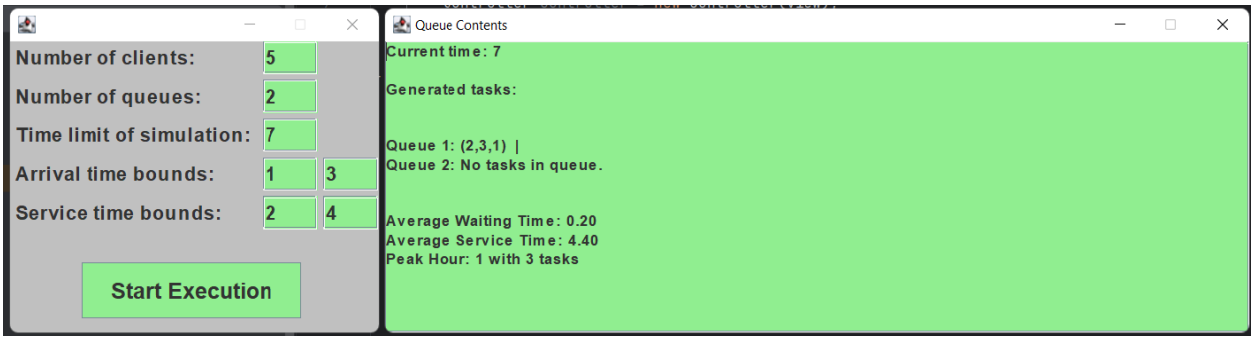
Controller class -> in its constructor has the principal view and adds an ActionListener to the "start button". Also has a method called displayQueueContents which is responsible for creating a new frame where the progression of the queues is visible in real time.
View class -> that is the class where all the principal frame contents are interconnected. Also, in this class we can see some getters to obtain the inputs of the user.
Scheduler class -> is responsible for creating and initializing the threads and servers.
ShortestQueue class -> is implementing the method that selects the server by its smaller size
ShortestTime class -> is implementing the method that selects the server by its smaller time
SimulationManager class -> is the class where all the tasks are generated randomly and where all the tasks are completed. Also, here we can see the methods that are used to generate the data

responsible for every input introduced (Average Waiting Time, Average Service Time, Peak Hour)

Server class -> is the class where all the tasks that belong to one queue are getting iterated
Task class -> is the class that represents the actual task that has an ID, Arrival Time Value, Service Time Value that must expire for the task to be considered finished

5. Results

The result for the logs required to run in the pdf is saved in a file with a suggestive name for each one.
Also, this is how it looks when the user introduces an input



6. Conclusions

After analyzing the code and testing the application, I personally conclude that it is fully functional if the input data follows a configuration like reality. What have I learned from this assignment? I have learned how to use threads and synchronization. For future development, the application could benefit from a richer graphical user interface in terms of appearance. From a logical standpoint, a system for validating user input data and displaying warnings in case of incorrect input should be implemented. Another development could involve finding a more efficient strategy focused on maximizing productivity, such as implementing a scheduling system.

7. Bibliography

The references that were consulted by the student during the implementation of the homework will be added.

1. *https://softwareengineering.stackexchange.com/questions/347498/difference-between-scenario-and-use-case*

2. https://dsrl.eu/courses/pt/materials/A2_Support_Presentation.pdf
3. https://www.javaprogramto.com/2020/04/java-collection-sort-custom-sorting.html
4. https://stackoverflow.com/questions/4818699/practical-uses-for-atomicinteger
5. http://tutorials.jenkov.com/java-concurrency/blocking-queues.html
6. https://howtodoinjava.com/java/multi-threading/atomicinteger-example/