DOCUMENTATION

QUEUES MANAGEMENT APPLICATION USING THREADS AND SYNCHRONIZATION MECHANISMS

CONTENTS

1. Assignment Objective

The objective of the assignment was to use threads to make a queue management with tasks and servers.
The application simulates a series of clients arriving for service, entering queues, waiting, being served, and leaving the queues. The goal is to minimize the total time clients spend in the queues. The user can input parameters such as the number of clients, number of queues, simulation interval, minimum and maximum arrival times, and minimum and maximum service times. The application assigns clients to queues based on their arrival times and the queue with the minimum waiting time.

2. Problem Analysis, Modeling, Scenarios, Use Cases

A. Problem Analysis

The problem required that a number of clients to be generated with random arriving and serving time.
After that, the program takes and adds to the queue that has the least waiting time the first task.

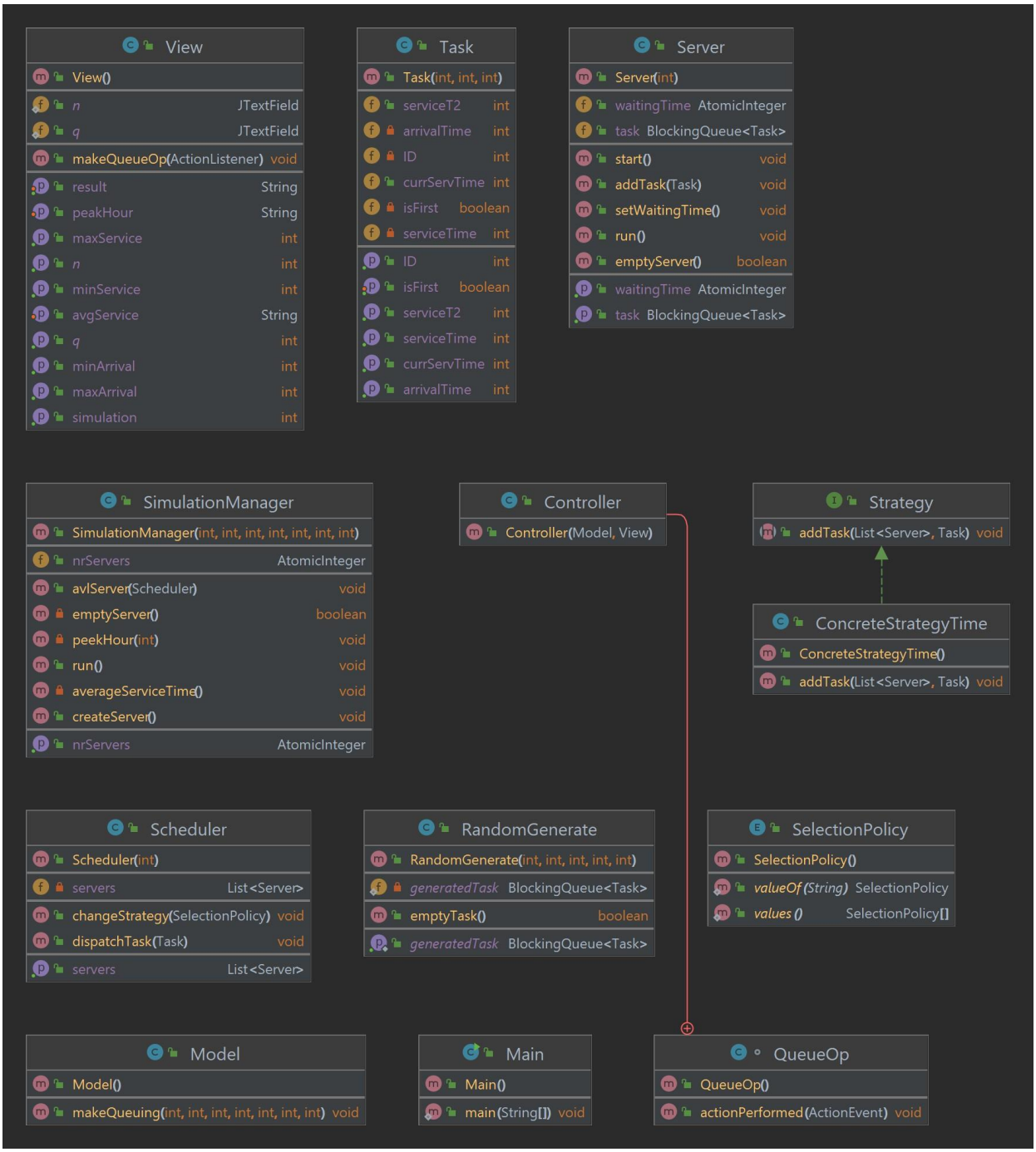
The task stays in the queue as long as the service time is, after that it leaves the queue.

The program also computes the average serving time, meaning the average of the service time that takes place in the queues. Also, the peak hour represents the moment in which the queues are at the maximum capacity

B. Modeling, Scenarios, User Cases

The program has places for the user to write all the needed inputs and after the program runs the peak hour, average service time and also the logs of the queues will be presented on the screen.

3. Design



4. Implementation

A. Task

The class "Task" represents a task to be executed in a simulation system, with properties such as ID, arrival time, service time, current service time, and a flag indicating if it is the first task in the queue.

B. Server

The class "Server" represents a server or processor in a queue management system. It maintains a blocking queue of tasks, keeps track of the number of clients and total service time, and implements the "Runnable" interface for concurrent execution. It has methods to add tasks to the server, calculate waiting time, start the server, check if the server is empty, and run the server by processing tasks in the queue.

C. Scheduler

The class "Scheduler" manages a collection of servers in a queue management system. It initializes the servers and starts their respective threads for concurrent processing. It provides methods to change the selection strategy, dispatch tasks to the servers based on the chosen strategy, and retrieve the list of servers. The class also maintains a time variable and utilizes the Strategy design pattern for task dispatching.

D. RandomGenerate

Generates randomly a task to be added

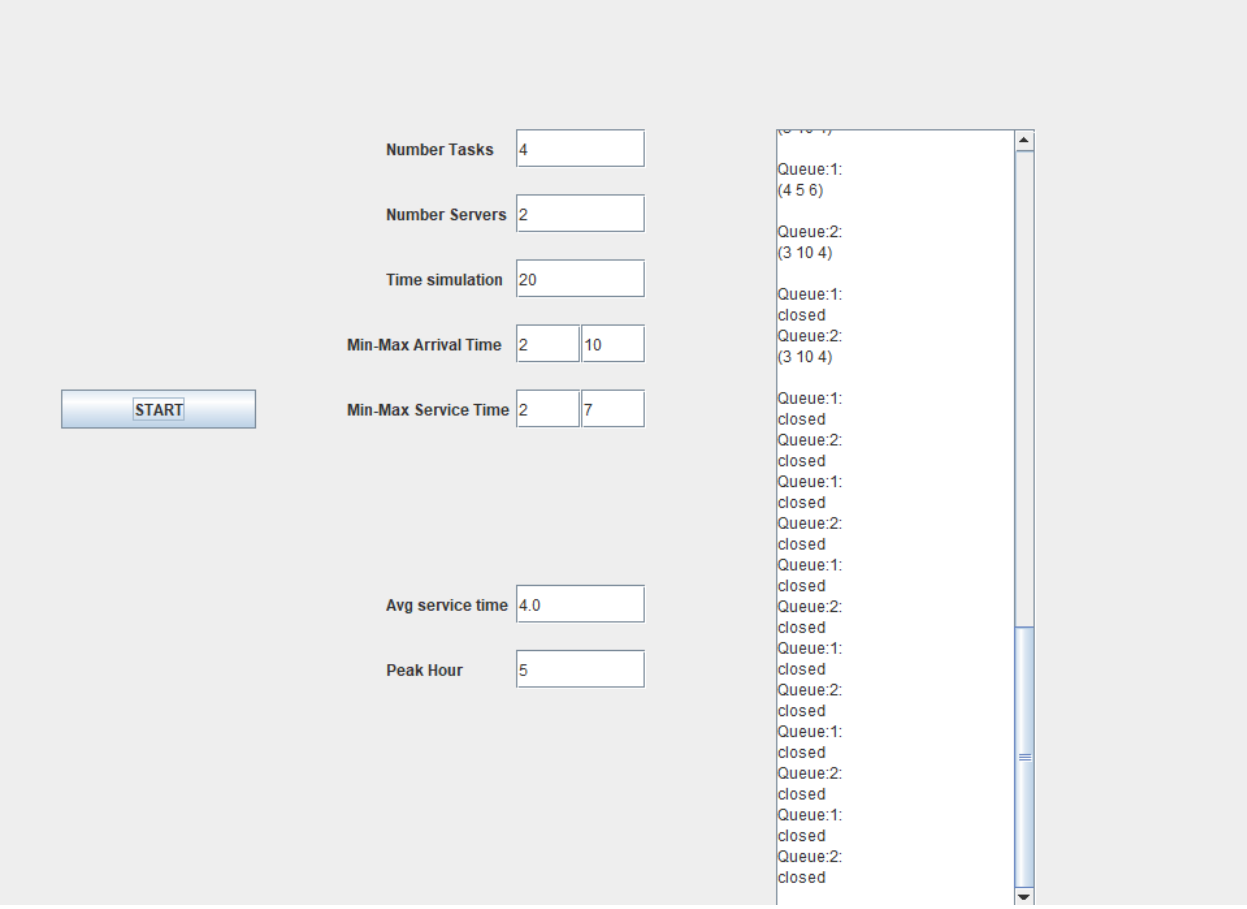
E. ConcreteStrategyTime

The class "ConcreteStrategyTime" implements the "Strategy" interface and represents a strategy for task dispatching based on the shortest waiting time in a queue. It determines the server with the minimum waiting time by iterating through a list of servers, calculates the waiting time for each server, and assigns the task to the server with the shortest waiting time. The class also maintains a total time variable and updates the server's total service time and number of clients accordingly.

F. SimulationManager

The class "SimulationManager" implements the "Runnable" interface and represents the main simulation manager in the system. It manages the simulation process by creating servers, generating tasks, and dispatching tasks to servers based on a scheduler. It also tracks and updates the current time, calculates average service time, and determines the peak hour of client arrivals. The class interacts with other classes such as "Scheduler", "RandomGenerate", and "Server" to perform the simulation. It also writes the simulation results to a file and provides methods for obtaining server status and performing various calculations related to the simulation.

5. Results



6. Conclusions

I learned how to better work with threads, to represent information in a text file.
What I want to do for the future is to arrange better the project and the methods in classes and also implement the other strategy.