DOCUMENTATION

ASSIGNMENT 2

STUDENT NAME: Teodora Huluban
GROUP: 30222

CONTENTS

1.Assignment Objective

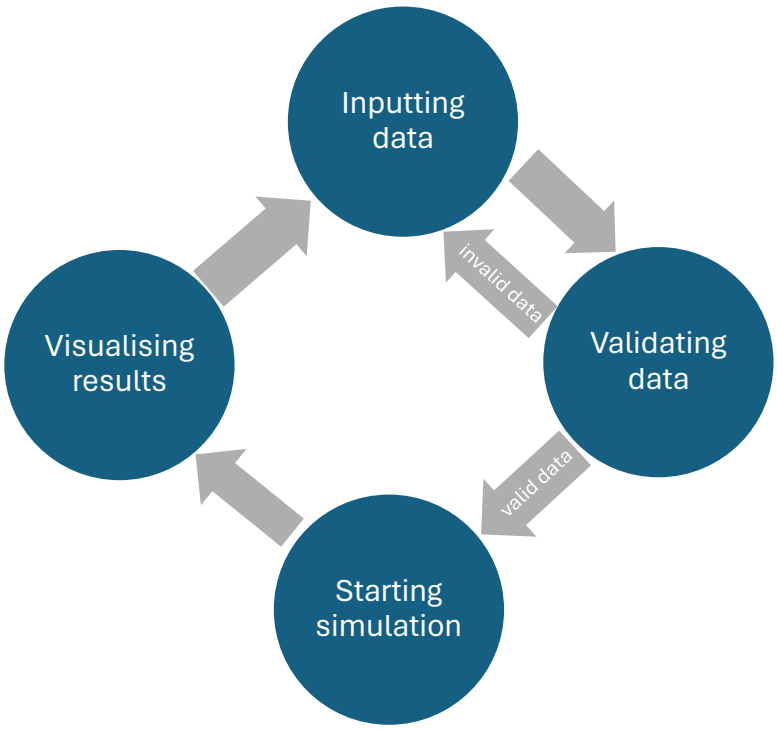
The main objective is to implement a queues management application which assigns clients to queues such that the waiting time is minimized.

The following sub-objectives were acquired to achieve the main objective:

- Analyze the problem and identify requirements
  - an analyze on the problem was done, and requirements were identified (detailed at point "2. Problem Analysis…")
- Design the application
  - a package/class diagram was realized for designing the project (detailed at point "3.Design")
- Implement the application
  - eight classes were implemented: two for modeling the clients and queues, four for the logic behind the simulation, and two for the graphical interface (detailed at point "4.Implementation")
- Test the application
  - several tests were done (detailed at point "5. Results")

2. Problem Analysis, Modeling, Scenarios, Use Cases

The application should be able to simulate clients waiting in line for a service, and should have a system that dispatches the clients to the queues, such that the waiting time is minimized. The clients will have an arrival time and a service time. Two strategies will be addressed: dispatch to the shortest queue as number of clients, or the queue with the shortest waiting time. The application will compute the average waiting time, the average service time, and peek hour.



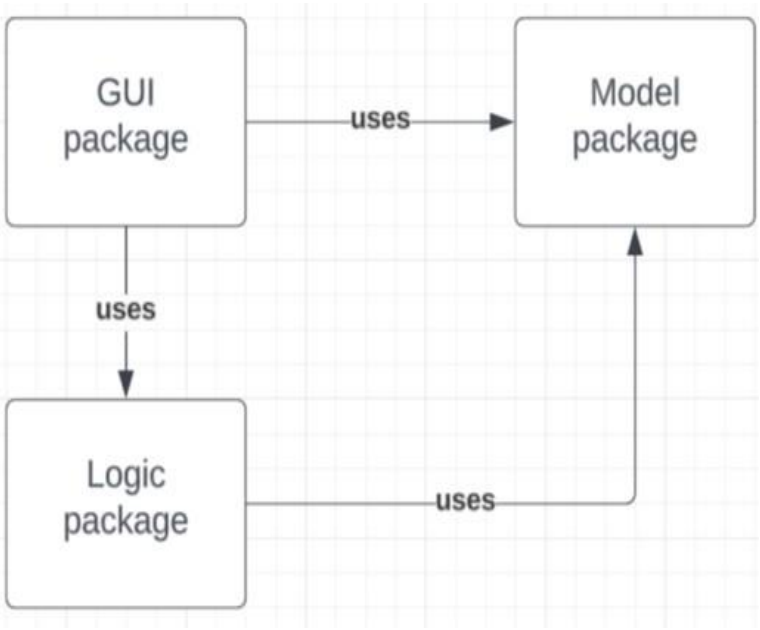
The application requires a graphical interface that lets the user input the following: number of clients, number of queues, simulation interval, minimum and maximum arrival time, minimum and maximum service time, and strategy. The system has to be able to validate the input, and a message of error should be displayed if the input is not valid. Only after the validation, the user should be able to start the simulation. The simulation panel will display the queues and the clients, and the progress made in the simulation. The results will be displayed after the simulation has ended.

3.Design

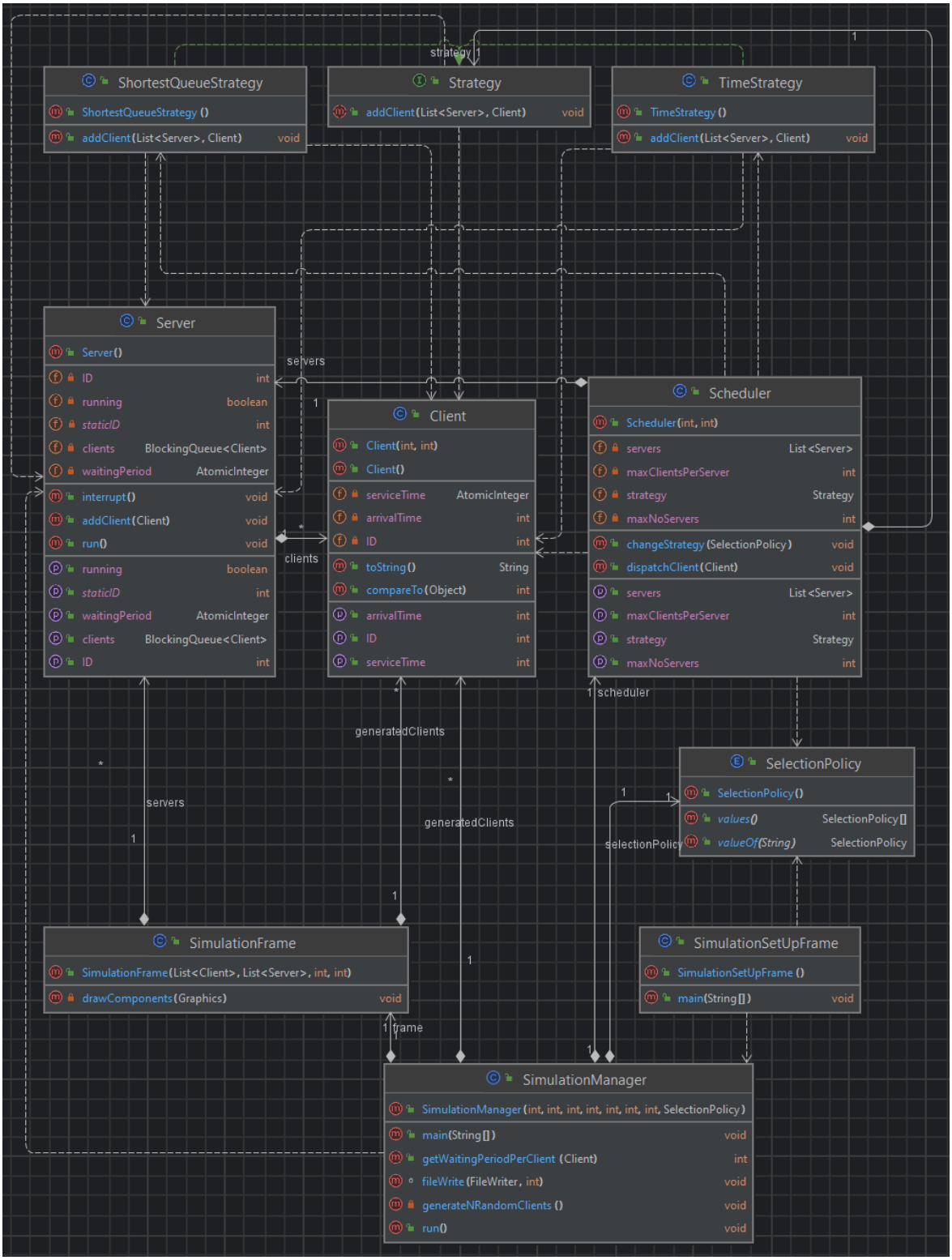
The project was structured in three packages:

- GUI package containing the SimulationSetUpFrame class and SimulationFrame class - first class lets the user input the data, and the second class displays the simulation
- Model package containing the Client class and Server class - manages the modeling of the clients and the queues
- Logic package containing the Strategy interface, SelectionPolicy enumeration, and the following classes: SimulationManager, Scheduler, ShortestQueueStrategy, TimeStrategy

Package Diagram:



Class Diagram:



4.Implementation

SimulationSetUpFrame Class:
- lets the user input data for the simulation; validates the data: in case of error, a message will be displayed, and if the data is valid, the user can press start button to start the simulation

SimulationFrame:
- simulates the clients being dispatch to each queue and the progress made in the queues. After the simulation has ended, the average waiting time, average service time, and peek hour will be displayed

Client Class:
- models the client, which has an ID, arrivalTime, and serviceTime

Methods:
- the toString method is used for displaying the client
- the compareTo method for ordering the clients based on their arrival time

Server Class:
- models the queue, which has an ID, a list of current clients, a waiting period, and a running state

Methods:
- the getWaitingPeriod method computes the waiting period for the queue, based on the current clients' service time
- the addClient method adds a client to the queue's list of clients, and increases the waiting period
- the interrupt method changes the state of the queue, from running to interrupted
- the run method manages the thread on which the queue runs. After the client is "processed" for a time equal with its service time, he is removed from the queue

TimeStrategy Class:
- implements the Strategy interface
- overrides the addClient method, which dispatches the client to the queue with the least waiting time

ShortestQueueStrategy Class:
- implements the Strategy interface
- overrides the addClient method, which dispatches the client to the queue with the least number of clients

Scheduler Class:
- holds the list of the queues, number of queues, and the strategy
- creates and starts the thread for each queue
- changeStrategy method sets the strategy based on the selection policy

SimulationManager Class:
- holds the data inputted through the interface and generates random clients based on that data

Methods:
- fileWrite method writes the progress made in the simulation, in the log file
- getWaitingPeriodPerClient method is used in calculating the average waiting time
- the run method manages the main thread which dispatches the clients to the queues based on their arrival time, and the chosen strategy, updates the simulation frame, increments the time counter, and computes the average waiting/service time and peek hour

5. Results

All the manual-input tests done until now have passed. The validation of the data works for all tested cases of inputting the wrong data. The simulation displays correctly the clients sent to each queue and the average waiting/service time and peek hour are computed correctly.

6. Conclusions

In conclusion, the queue management application successfully achieved its main objective of minimizing client waiting times. The application's structure includes logical packages, an intuitive graphical interface, and efficient client allocation strategies. Testing confirmed the application's accuracy in simulating client behaviour and system performance. From this project I have gain proficiency in working with multiple threads that run simultaneously.

Potential improvements could involve optimizing client allocation algorithms and expanding functionality to handle more complex scenarios.

7. Bibliography

- http://docs.oracle.com/javase/tutorial/essential/concurrency/index.html
- http://www.tutorialspoint.com/java/util/timer_schedule_period.htm
- http://www.javacodegeeks.com/2013/01/java-thread-pool-example-using-executors-andthreadpoolexecutor.html