DOCUMENTATION

QUEUES MANAGEMENT APPLICATION USING THREADS AND SYNCHRONIZATION MECHANISMS

ASSIGNMENT_2

STUDENT NAME: Maris Radu-Ioan
GROUP: 30421

CONTENTS

1. Assignment Objective

The Main Objective:

Design and implement a queues management application which assigns clients to queues such that the waiting time is minimized. The application should be designed according to the layered architecture pattern. The queues management application should simulate (by defining a simulation time $t_{simulation}$) a series of N clients arriving for service, entering Q queues, waiting, being served and finally leaving the queues. All clients are generated when the simulation is started, and are characterized by three parameters: ID (a number between 1 and N), $tarrival$ (simulation time when they are ready to enter the queue) and $tservice$ (time interval or duration needed to serve the client; i.e. waiting time when the client is in front of the queue). The application tracks the total time spent by every client in the queues and computes the average waiting time. Each client is added to the queue with the minimum waiting time when its $tarrival$ time is greater than or equal to the simulation time ($tarrival \geq tsimulation$).

Sub-Objectives:

- Generating the number of clients
- Generating the number of queues
- Having a simulation interval according to the tMAXSimulationTime
- Generating the min and max arrival time
- Generaring the min and max service time

2. Problem Analysis, Modeling, Scenarios, Use Cases

The application is meant for users that want to synchronize the operations on multiple queues. The input that a user can solve are:

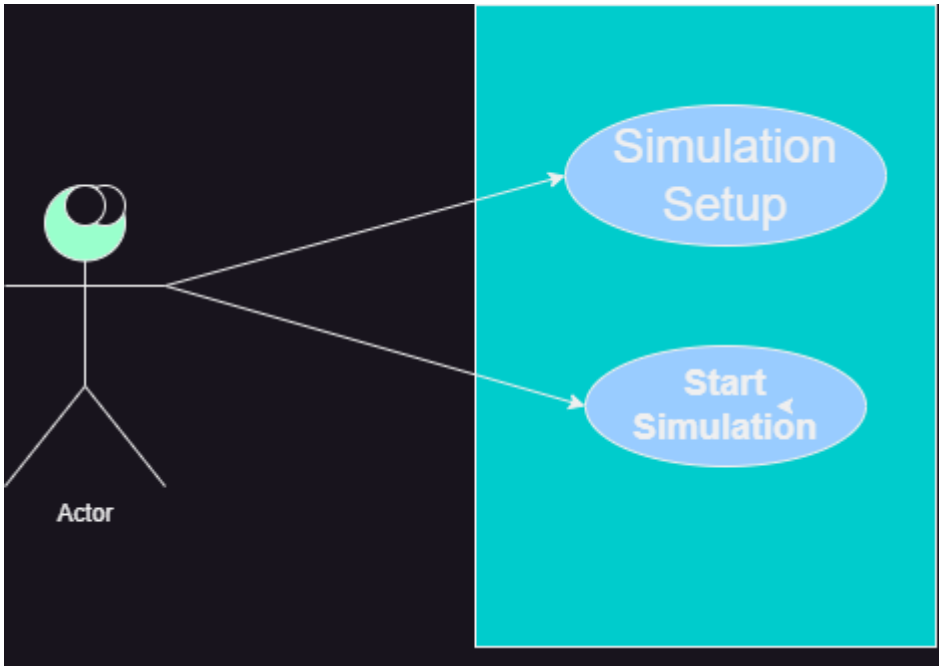
- Number of clients
- Number of queues
- Max and min for arrival time
- Max and min for service time
- Simulation time

Successful scenario:

This case is met when the user's input is valid. This means that the user managed to input the data and the simulation begins when he presses the button. Then the log will be stored in a logfile of type .txt.

Unsuccessful scenario:

When the input is not valid, considering the operations specified before, the imput is not recognized by the interface. Therefore, if the user does not input valid data, then the simulation will not be executed.

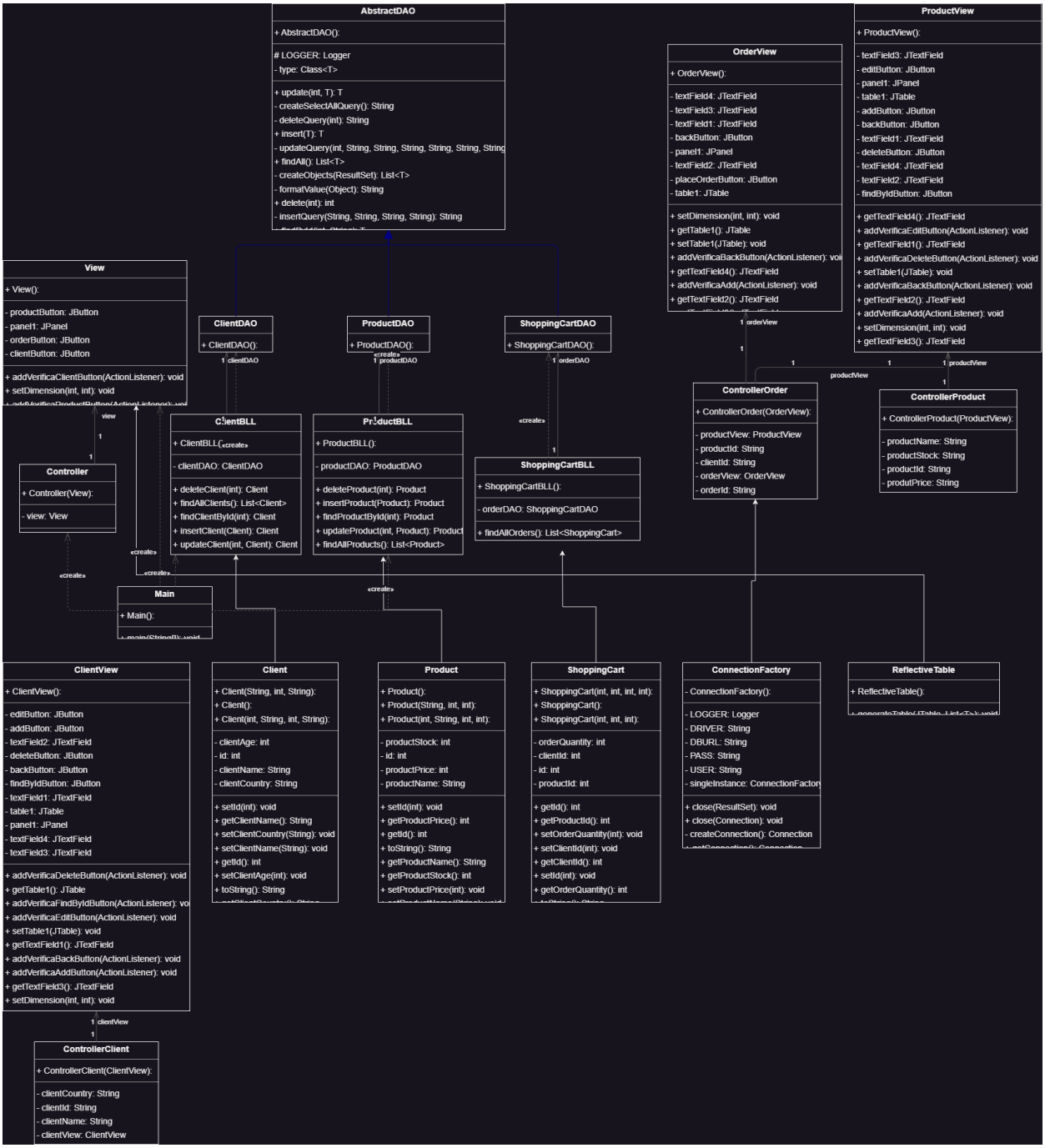


Use case diagram

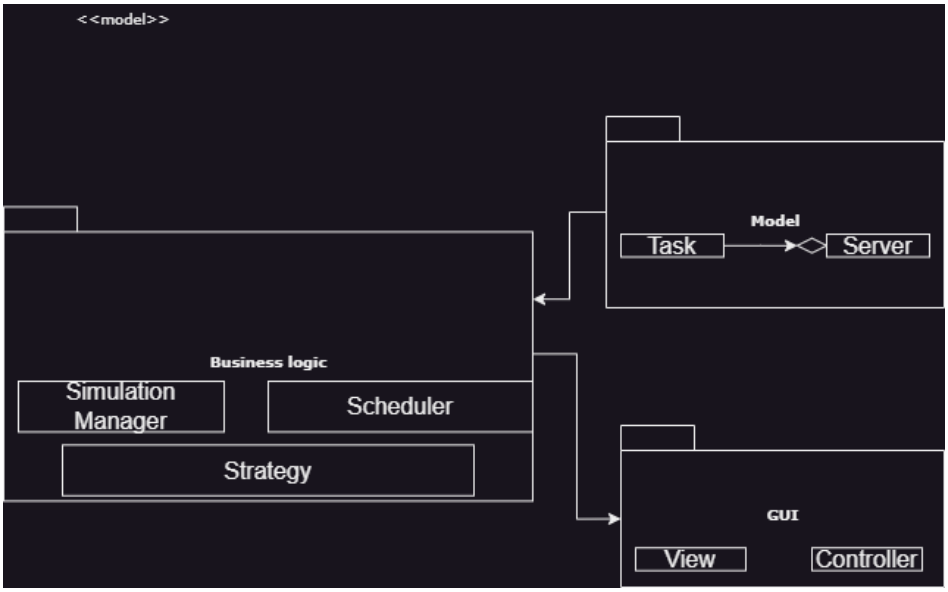
3. Design

I used an OOP design in implementing the solution, by distributing them in classes and methods. For the implementation I used 8 main classes that I distributed in 4 packages.

The Interface is made using Java Swing.
The following are the UML Diagram and Package Diagram.
UML Diagram:



Package Diagram:



4. Implementation

The GUI package contains 2 Java classes for the Interface. The View has 2 buttons and they are meant take the data from the 7 text fields there are linked to the simulation data. Then, there are 2 buttons: an Start button that is used to add the entry data to the simulation and a Show button that is used to dysplay the contents of the logfile.

Also, in the same package, I implemented the controller for the interface. The name of the class is Controller. It implements the functionality for the 2 buttons.

In the Model package there are 2 Java classes: Server and Task. They are meant to implement the models of the objects.

The BusinessLogic package also has 3 classes: Scheduler, SimulationManager and Strategy. The Strategy implements the shortest time strategy for the queue. That means that a generated client will chose a queue with the minimum waiting time. Scheduler is meant to choose the strategy and then appoint a client to a queue. The Simulation manager is responsible for synchronizing the threads for all the queues.

5. Results

The result is that I managed to sincronyze the application for given values and it worked as intended:

Time : 0
Waiting list:[(11,3,2), (17,2,1), (20,2,0), (29,3,3)]
Queue1:
Queue2:
Time : 1
Waiting list:[(11,3,2), (17,2,1), (20,2,0), (29,3,3)]
Queue1:
Queue2:
Time : 2
Waiting list:[(11,3,2), (17,2,1), (20,2,0), (29,3,3)]

Queue1:
Queue2:
Time : 3
Waiting list:[(11,3,2), (17,2,1), (20,2,0), (29,3,3)]
Queue1:
Queue2:
Time : 4
Waiting list:[(11,3,2), (17,2,1), (20,2,0), (29,3,3)]
Queue1:
Queue2:
Time : 5
Waiting list:[(11,3,2), (17,2,1), (20,2,0), (29,3,3)]
Queue1:
Queue2:
Time : 6
Waiting list:[(11,3,2), (17,2,1), (20,2,0), (29,3,3)]
Queue1:
Queue2:
Time : 7
Waiting list:[(11,3,2), (17,2,1), (20,2,0), (29,3,3)]
Queue1:
Queue2:
Time : 8
Waiting list:[(11,3,2), (17,2,1), (20,2,0), (29,3,3)]
Queue1:
Queue2:
Time : 9
Waiting list:[(11,3,2), (17,2,1), (20,2,0), (29,3,3)]
Queue1:
Queue2:
Time : 10
Waiting list:[(11,3,2), (17,2,1), (20,2,0), (29,3,3)]
Queue1:
Queue2:
Time : 11
Waiting list:[(17,2,1), (20,2,0), (29,3,3)]
Queue1: (11,3,2)
Queue2:
Time : 12
Waiting list:[(17,2,1), (20,2,0), (29,3,3)]
Queue1: (11,3,2)
Queue2:
Time : 13
Waiting list:[(17,2,1), (20,2,0), (29,3,3)]
Queue1: (11,3,2)
Queue2:

Time : 14
Waiting list:[(17,2,1), (20,2,0), (29,3,3)]
Queue1:
Queue2:
Time : 15
Waiting list:[(17,2,1), (20,2,0), (29,3,3)]
Queue1:
Queue2:
Time : 16
Waiting list:[(17,2,1), (20,2,0), (29,3,3)]
Queue1:
Queue2:
Time : 17
Waiting list:[(20,2,0), (29,3,3)]
Queue1: (17,2,1)
Queue2:
Time : 18
Waiting list:[(20,2,0), (29,3,3)]
Queue1: (17,2,1)
Queue2:
Time : 19
Waiting list:[(20,2,0), (29,3,3)]
Queue1:
Queue2:
Time : 20
Waiting list:[(29,3,3)]
Queue1: (20,2,0)
Queue2:
Time : 21
Waiting list:[(29,3,3)]
Queue1: (20,2,0)
Queue2:
Time : 22
Waiting list:[(29,3,3)]
Queue1:
Queue2:
Time : 23
Waiting list:[(29,3,3)]
Queue1:
Queue2:
Time : 24
Waiting list:[(29,3,3)]
Queue1:
Queue2:
Time : 25
Waiting list:[(29,3,3)]

Queue1:
Queue2:
Time : 26
Waiting list:[(29,3,3)]
Queue1:
Queue2:
Time : 27
Waiting list:[(29,3,3)]
Queue1:
Queue2:
Time : 28
Waiting list:[(29,3,3)]
Queue1:
Queue2:
Time : 29
Waiting list:[]
Queue1: (29,3,3)
Queue2:
Time : 30
Waiting list:[]
Queue1: (29,3,3)
Queue2:
Time : 31
Waiting list:[]
Queue1: (29,3,3)
Queue2:
Time : 32
Waiting list:[]
Queue1:
Queue2:
Time : 33
Waiting list:[]
Queue1:
Queue2:
Time : 34
Waiting list:[]
Queue1:
Queue2:
Time : 35
Waiting list:[]
Queue1:
Queue2:
Time : 36
Waiting list:[]
Queue1:
Queue2:

Time : 37
Waiting list:[]
Queue1:
Queue2:
Time : 38
Waiting list:[]
Queue1:
Queue2:
Time : 39
Waiting list:[]
Queue1:
Queue2:
Time : 40
Waiting list:[]
Queue1:
Queue2:
Time : 41
Waiting list:[]
Queue1:
Queue2:
Time : 42
Waiting list:[]
Queue1:
Queue2:
Time : 43
Waiting list:[]
Queue1:
Queue2:
Time : 44
Waiting list:[]
Queue1:
Queue2:
Time : 45
Waiting list:[]
Queue1:
Queue2:
Time : 46
Waiting list:[]
Queue1:
Queue2:
Time : 47
Waiting list:[]
Queue1:
Queue2:
Time : 48
Waiting list:[]

Queue1:
Queue2:
Time : 49
Waiting list:[]
Queue1:
Queue2:
Time : 50
Waiting list:[]
Queue1:
Queue2:
Time : 51
Waiting list:[]
Queue1:
Queue2:
Time : 52
Waiting list:[]
Queue1:
Queue2:
Time : 53
Waiting list:[]
Queue1:
Queue2:
Time : 54
Waiting list:[]
Queue1:
Queue2:
Time : 55
Waiting list:[]
Queue1:
Queue2:
Time : 56
Waiting list:[]
Queue1:
Queue2:
Time : 57
Waiting list:[]
Queue1:
Queue2:
Time : 58
Waiting list:[]
Queue1:
Queue2:
Time : 59
Waiting list:[]
Queue1:
Queue2:

6. Conclusions

I conclusion, I learned how to use Atomic variables in my creation of functions in Java and how to work with treads. Also, I found out how to write in a text file in Java.

As a future improvement, the application could have multiple factors in order for a client to choose the strategy. Also, it can have a more friendly user interface.

7. Bibliography

The references that I consulted during the implementation of the homework are the following:

1. https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/atomic/AtomicInteger.html
2. https://www.w3schools.com/java/java_threads.asp
3. https://www.geeksforgeeks.org/java-threads/