DOCUMENTATION

Assignment 2

STUDENT NAME: Arraj Kamel
GROUP:30421/1

CONTENTS

1. Assignment Objective

1.1. the main objective :

Design and implement a queues management application which assigns clients to queues such that the waiting time is minimized.

1.2. the sub-objectives :

1.2.1. generating tasks:

generate a specific number of tasks, where each task represents a client, to simulate the queue.

1.2.2. create and start threads :

We are given a number of servers, which represent the queue, for each server we create a Thread and start it.

2. Problem Analysis, Modeling, Scenarios, Use Cases

2.1. Problem analysis :

Queues are commonly used to model real world domains. The main objective of a queue is to provide a place for a "client" to wait before receiving a "service". The management of queue-based systems is interested in minimizing the time amount their "clients" are waiting in queues before they are served. One way to minimize the waiting time is to add more servers, i.e., more queues in the system (each queue is considered as having an associated processor) but this approach increases the costs of the service supplier.

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), $t_{arrival}$ (simulation time when they are ready to enter the queue) and $t_{service}$ (time interval or duration needed to serve the client; i.e. waiting time when the client is in front of the queue).

3. Design

3.1. OOP design :

3.1.1. Classes :

- ***Server*** *: represents the queue.*
- ***Task*** *: represents the client.*
- ***Scheduler*** *: manages the process of pushing the tasks to the suitable servers .*
- ***SimulationManager*** *: it generates the random tasks and manages the whole process and it has the main thread after that one which existed in main method.*
- ***concreteStrategyTime*** *: add the task to the suitable server in term of time.*
- ***concreteStrategyQueue*** *: add the task to the suitable server in term of queue content.*

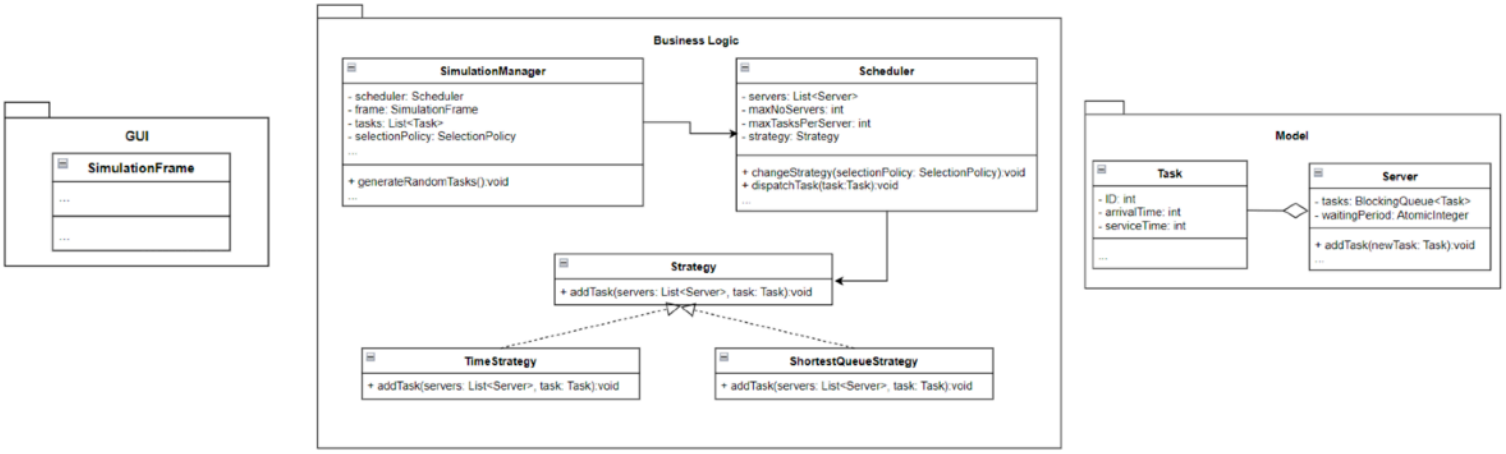
3.1.2. Interfaces :

- ***Strategy*** *: represents the selection policy.*

3.1.3. Enumeration :

•**SelectionPolicy :** has two elements, **“SHORTEST_TIME , SHORTEST_QUEUE” .**

4. Implementation



Implementation diagram

5. Results

you can find the testing files in the repo

6. Conclusions

in conclusion we see how the threads divide the cpu in an efficient way to do more that one mission at the same time as we see in our assignment how can we manage more that one queue in the same time by increase the number of servers.

I have learned how to use threads and how to employee them efficiently.

7. Bibliography

I did not use any book as a reference for this project, I use some YouTube videos some times to get simple information and for simple coding issues I rarely use AI.