



UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

PROGRAMMING TECHNIQUES
ASSIGNMENT 2

QUEUES
MANAGEMENT
APPLICATION

STUDENT: GYARMATHY CSABA

UNIVERSITY: TECHNICAL UNIVERSITY OF CLUJ-NAPOCA

FACULTY: COMPUTER SCIENCE

TEACHER: CRISTINA BIANCA POP

TEACHER ASSISTANT: ANDREEA VALERIA VESA

YEAR: 2

ACADEMIC YEAR: 2021/2022

GROUP: 30424

Contents

Assignment Objective	3
Project analysis, scenario, approach and use cases	3
Analysis	3
Scenario and modelling.....	3
Use case scenarios.....	5
Implementation	6
Diagrams	6
Class diagram.....	6
Use case diagram	7
Packages.....	7
Business Logic Layer (BLL).....	7
Model	8
View	8
Results.....	8
Further development	8
Conclusion	9
Bibliography	9

Assignment Objective

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

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 *tsimulation*) 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 ($tarrival \geq tsimulation$).

The following data should be considered as input data for the application that should be inserted by the user in the application's user interface:

- Number of clients (N);
- Number of queues (Q);
- Simulation interval (*tsimulationMAX*);
- Minimum and maximum arrival time ($tarrivalMIN \leq tarrival \leq tarrivalMAX$);
- Minimum and maximum service time ($tserviceMIN \leq tservice \leq tserviceMAX$);

Project analysis, scenario, approach and use cases

Analysis

As seen in the description, we will have to work with virtual models of real life structures, and so, they are associated with random events, that should not be predictable in any way. Thus, we will use threads and randomly generated data.

A Java Thread is like a virtual CPU that can execute your Java code - inside your Java application. when a Java application is started its main() method is executed by the main thread - a special thread that is created by the Java VM to run your application. From inside your application, you can create and start more threads which can execute parts of your application code in parallel with the main thread. Java threads are objects like any other Java objects. Threads are instances of class `java.lang.Thread`, or instances of subclasses of this class. In addition to being objects, java threads can also execute code.

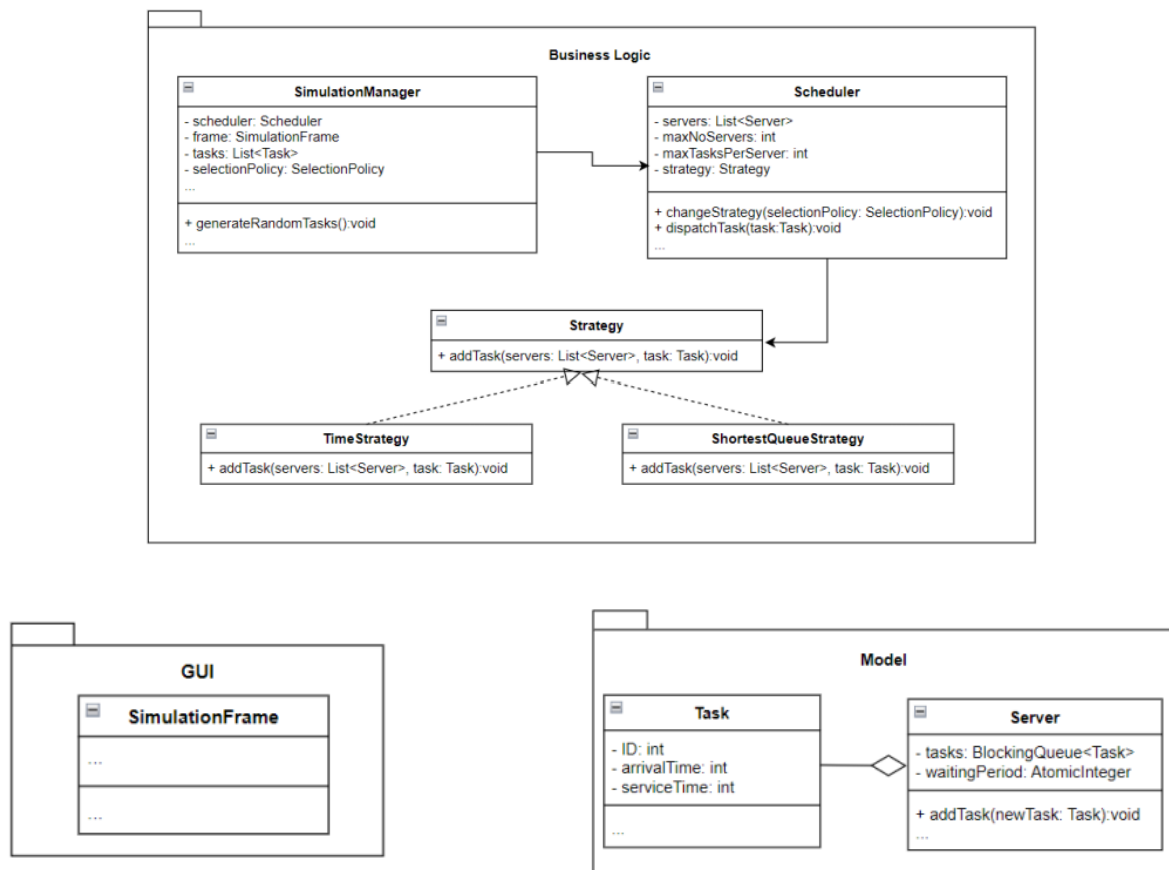
Scenario and modelling

To be able to model a queue-based system we must decompose the problem into the basic building units: client (task), queue (server) and simulation manager. Each queue will run its separate thread.

Customers should retain the information about their arrival and service time, and each customer should be unique, hence we'll also associate them an ID number.

Queues are dynamic, their data constantly changes in time as customers arrive and they are served. Queues should hold an array of customers which is constantly updated throughout the thread's evolution. Besides this, information like the average of waiting time, service time and empty queue time is also stored and constantly updated. Constant data passed to the queues from the user are the minimum and the maximum service time, which define the interval of the randomly generated numbers.

The simulation manager holds an array of queues, and adds customers to them, which arrive randomly. For this, we'll pass the minimum and the maximum required arrival time from the user and generate random numbers within the specified interval. The next arriving customer will always be added to the shortest queue.



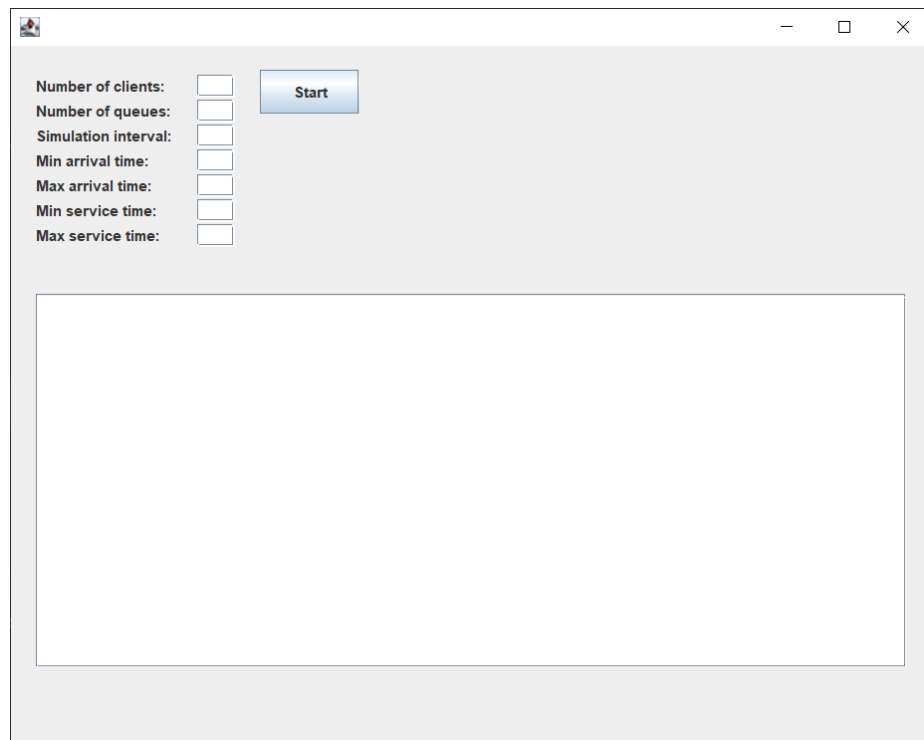
The layered architecture style is one of the most common architectural styles. The idea behind Layered Architecture is that modules or components with similar functionalities are organized into horizontal layers. As a result, each layer performs a specific role within the application.

The layered architecture style does not have a restriction on the number of layers that the application can have, as the purpose is to have layers that promote the concept of separation of concerns. The layered architecture style abstracts the view of the system, while providing enough detail to understand the roles and responsibilities of individual layers and the relationship between them.

The Graphical User Interface (GUI), unlike the previous and future projects, consists of only one window, where everything is displayed.

The inputs mean, for example, that 10 clients will be scheduled in the given 3 queues periodically in an arrival time bigger than 2 seconds and smaller than 5, will be served in the queue (when it is their turn) in a time greater than 3 seconds and smaller than 6. 5 queues will be displayed and simulated for 30 seconds of simulation time.

Whenever these inputs do not conform, for example, non-numeric characters are input or $\text{min} > \text{max}$, an error pop-up window appears:



Use case scenarios

The user of the application must input the required data, the preferred simulation time and press the start button to start the simulation. From this point on, he will follow the evolution of the queues until the simulation interval is over. At the end of the simulation, the average waiting time and average service time will be displayed, as well as the peak time (the moment when the total number of customers in the market was the largest).

Precondition: The user successfully launches the application. Upon successful launch, the following actions will go down:

1. The user introduces the minimum arrival time for the customers
2. The user introduces the maximum arrival time for the customers
3. The user introduces the minimum service time for the customers.
4. The user introduces the maximum service time for the customers.
5. The user introduces the number of queues.
6. The user introduces the preferred simulation time.
7. The user introduces the number of clients
8. The user presses the “Start” button.
9. Input data is parsed, transformed and passed internally to the control of the system.
10. Simulation starts, threads are created and started, queues appear on the interface in their initial state.

11. Customers arrive and are served, evolution of the market can be followed on the screen, events are specified in the log; running time is also displayed.
12. Simulation time is over, customers stop arriving, queues stop serving, and all threads stop.
13. Final data remains on the screen.

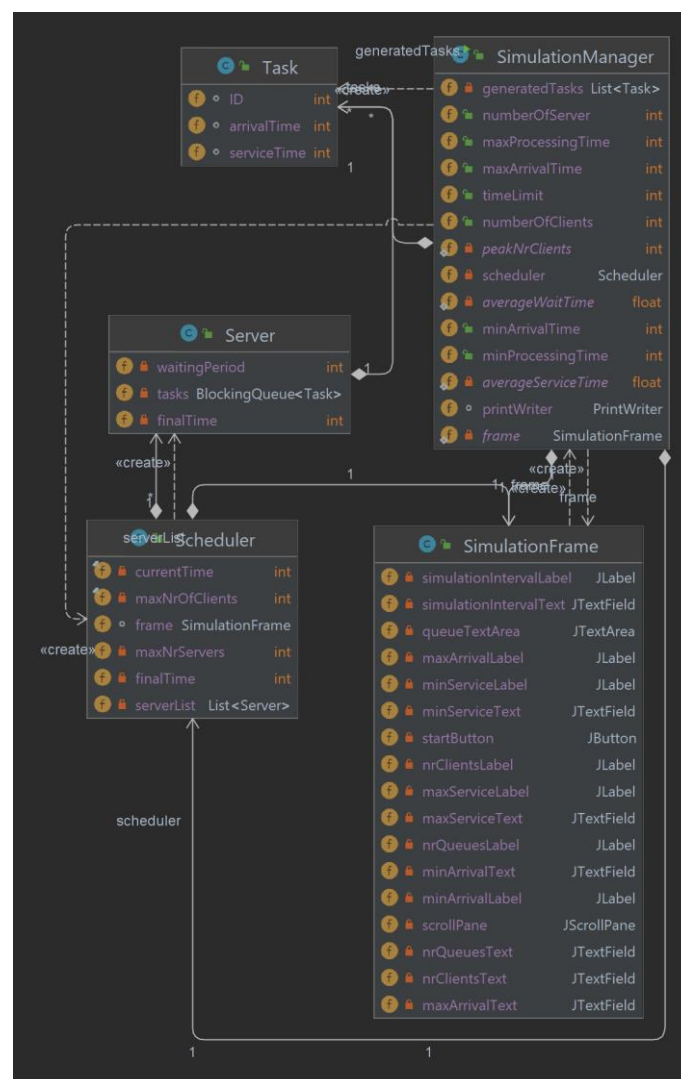
Implementation

Diagrams

Class diagram

In the course of creating the project, several diagrams were made to help with the development.

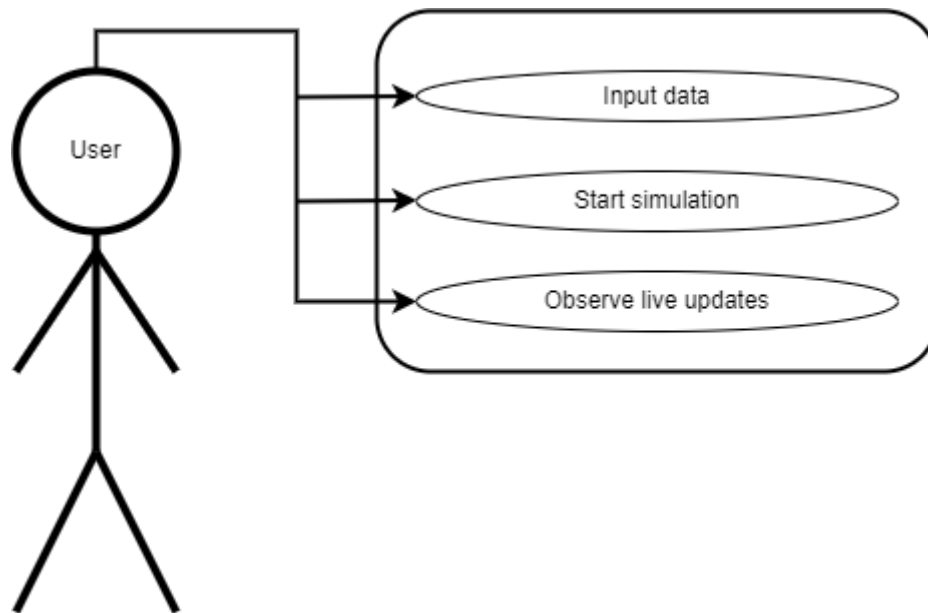
Below you will see a UML (Unified Modelling Language) diagram, used to visualize, specify, and document the software system. This diagram contains all the Java classes used in the implementation of this project, as well as their dependencies and relations with each other. This diagram is known as the class diagram.



Since several fields and components were made in each class, to see a more detailed diagram, please refer to the generated one in the Java IntelliJ IDE.

Use case diagram

The following diagrams are the use case diagram, depicting the different cases a user finds itself in.

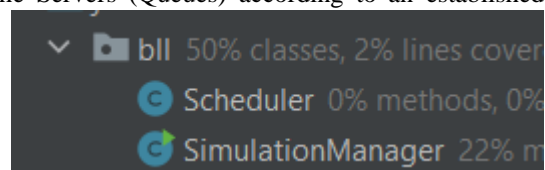


Packages

Business Logic Layer (BLL)

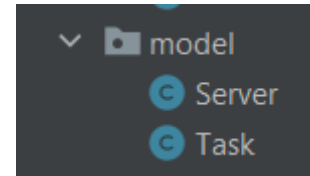
This package contains the logic of the program. It contains two classes, SiumlationManager and Scheduler, which create, start, organize, stop and finally, delete threads from the queues. The package can also correspond as the controller package of the Model-View-Controller (MVC) architecture, as it contains the main controller of the application.

The Scheduler package sends Tasks (Clients) to the Servers (Queues) according to an established strategy. The SimulationManager generates Tasks with random arrival Time and service Time, according to the range of the user input. It also contains a simulation loop, which updates based on the current time, it calls the Scheduler to dispatch tasks, i.e., sort the tasks into servers, and it also updates the GUI in real time.



Model

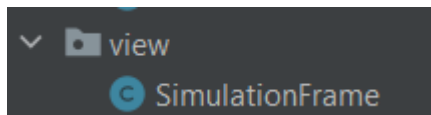
This package, as suggested by its name, models the Server and Task objects accordingly. Each task has an ID, an arrival time and a service time. The server has a list of tasks, a total waiting period and a final time. Both classes contain, beside the constructor, getters and setters, which are not all used because I just auto-generate them in case I need them but, in the end, I don't use them and just cannot be arsed to delete them even though they're additional lines of code that is irrelevant and never used lmao.



View

Finally, we have the view package, consisting of a single class SimulationFrame that contains all the GUI related components of the application's window. The package contains only graphical user interface elements implemented in Java Swing.

Swing is a set of program components for Java programmes that provide the ability to create graphical user interface (GUI) components, such as buttons and scroll bars, that are independent of the windowing system for specific operating system. Swing components are used with the Java Foundation Classes (JFC).



It contains the whole implementation of the graphical user interface, it extends JFrame and it has a list of components from which the constructor 'constructs' the user interface.

As it is constantly changing in time, when the application is opened, not all data is displayed at once. The boxes for the user input data are displayed and below this, the empty log of events. After correct data was input, and the user pressed the start button, the requested number of queues appear, their data and evolution are displayed, while events are listed in the log.

To be able to display queues of preferred number, in the upper panel a vertical scrollbar was added, while the log serves with a vertical one. These but become usable only if the size of the panel exceeds the one of the main frame, for example when many queues were added, or many events occurred already and the log listed more that can fit into its initial height.

Results

The development of this assignment resulted in a decent, user-friendly application that illustrates queue evolution in a very simple manner. Not only can the user configure the input data of this system, but all relevant information is retrieved regarding the simulation, and at the end of the simulation, is saved in a Log.txt file, so it can be read even after the application has been closed.

Further development

I feel like this project can not be possibly developed any further, as far as implementation and user interface goes. Everything works perfectly, all data is displayed and it's easy to use with no errors.

However, this application can firstly serve for tracking and analyzing random data, which can be used to calculate probabilities and such. Further developments may include adding more constraints to the evolution in

time of the system or calculating and displaying more information about the queues. A more sophisticated and colorful animation of the arriving and leaving customers could also be added in the future.

And I just remembered that the application is lacking any sort of warning or exception when wrong data is given as input, therefore, as a possible further development, these can be added.

Conclusion

With this assignment I have certainly learned the usage of threads, which I have failed to understand before. By developing the application, I have also enhanced my skills in Java programming language and in developing a proper application.

I arrived at the conclusion that facing problems with your code and trying to make it work by yourself, through the mean of research, has the benefit of learning new concepts and a better use of the known ones

Bibliography

<https://dsrl.eu/courses/pt/>

<https://stackoverflow.com/>

<https://www.youtube.com/>

<https://www.geeksforgeeks.org/>