Technical University of Cluj-Napoca

Programming Techniques

Assignment 2

Queue Mangament System

Teacher: prof. Ioan Salomie

Teacher Assistant: Dr. Viorica Chifu

Student: Rusu Horia Gabriel

Group : 30421/2

1.Objectives

Main objective: the main objective is to design and implement a queue managment system with a dedicated graphical interface trough which the user can insert the perimeters of the simulation and view the log of the events both in the user interface as well as the .txt document.

Sub-objectives:

* Analyze the problem and identify requirements
* Design queue managment system
* Implement the queue managment system
* Test the queue managment system

2. Problem analysis,modeling,scenarios,use cases

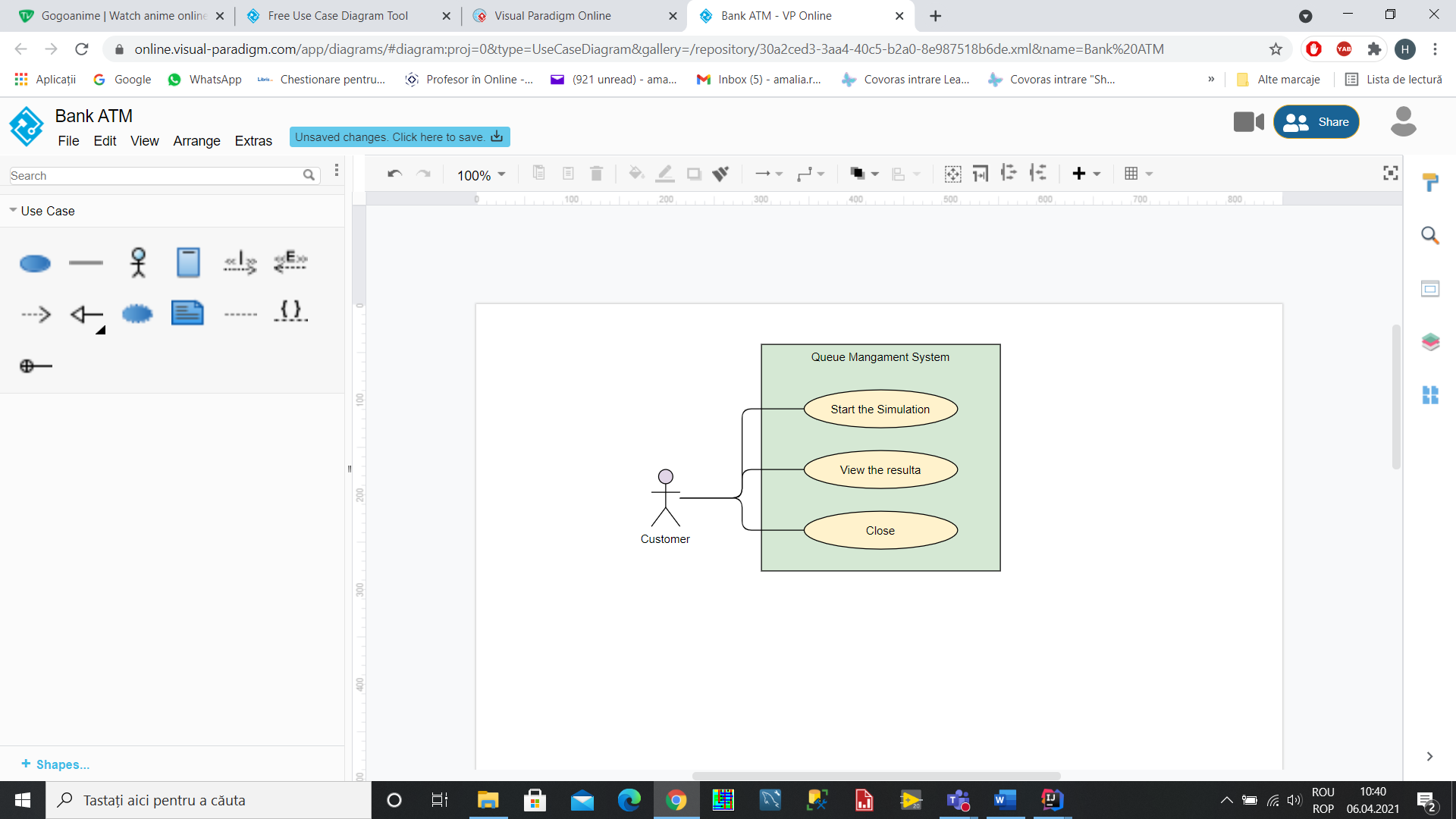
a. Problem analysis

The queue managment system, as the name implies, takes randomly generated clients from a waiting list and inserts them in a queue, where they stay for a certain amount of time. The clients are identified by an Id, an arrival time which represents the time in the overall simulation time in which the certain client arrives at the store or whatever the simulated queue is supposed to represent. The service time on the other hand represents the time the client spends at the queue. This service time represents the overall waiting period of the certain queue. The next client that comes will got to one of the queues based on the waiting period and will of course pick the shortest one based on this criterion. On this note it is important to note that the queue with the smallest number of clients waiting, though it may seem to be the shortest one from the waiting periods point of view, this is not necessarily always the case and there can be queues with more clients and shorter waiting times than queues with fewer clients. The result of the execution, should not only be the log of events, but also the average waiting time, computed as the waiting period for each client when inserted into one of the queues. The average service time si the average of the service times of each of the clients when inserted, and the peak hour is the time at which the queues have per total the greatest number of clients.

b. Modelling of the problem

The user will be able to the implemented user interface in order to write in the required slots the number of clients that will have to be implemented in the queues, the number of queues in the simulation, the simulation time, the maximum and minimum arrival times and the maximum and minimum service times. The individual service and arrival times for each of the clients will be chosen at randomly, but will be values in the given intervals. The result, as in the log of the operation will be displayed in the log of events and at the end the average waiting time, the average service time and the peak hour for the given input. If the proper syntax is not used a message will appear indicating the case and you will have to change your input.

c. Scenarios and Use Cases



Title: Start Simulation

Resume:  The first thing that a user has to do is to insert in the graphical user interface all the needed parameters for the application to work as it should, specifically the needed perimeters are the minimal and the maximum arrival time, the minmum and maximum service time, the simulation interval, the number of queues and the number of clients.

Actors: The user

Main success Scenarios:

1. The user successfully inserts the needed values in the designated slots
2. The user presses the start button in order to start the computation of the thread
3. The user sees the resulted log of events as well as the computed values, calculated during the serving process, such as the average waiting time, the average service time and peak hour.

Alternative Sequence: Incorrect data

1. The user inserts elements that do not comply to the required standard such as using letters instead of numbers or using a greater value for the minimum as compared to the maximum, both when the arrival as well as the service time are chosen.
2. A message will be displayed telling you that your input is wrong and that you should change it.

Use case: Close

Actor: user

Main Success scenario:

1. The user presses the close button and closes the simulations user interface

Alternate Success scenario: None

Functional Requirements:

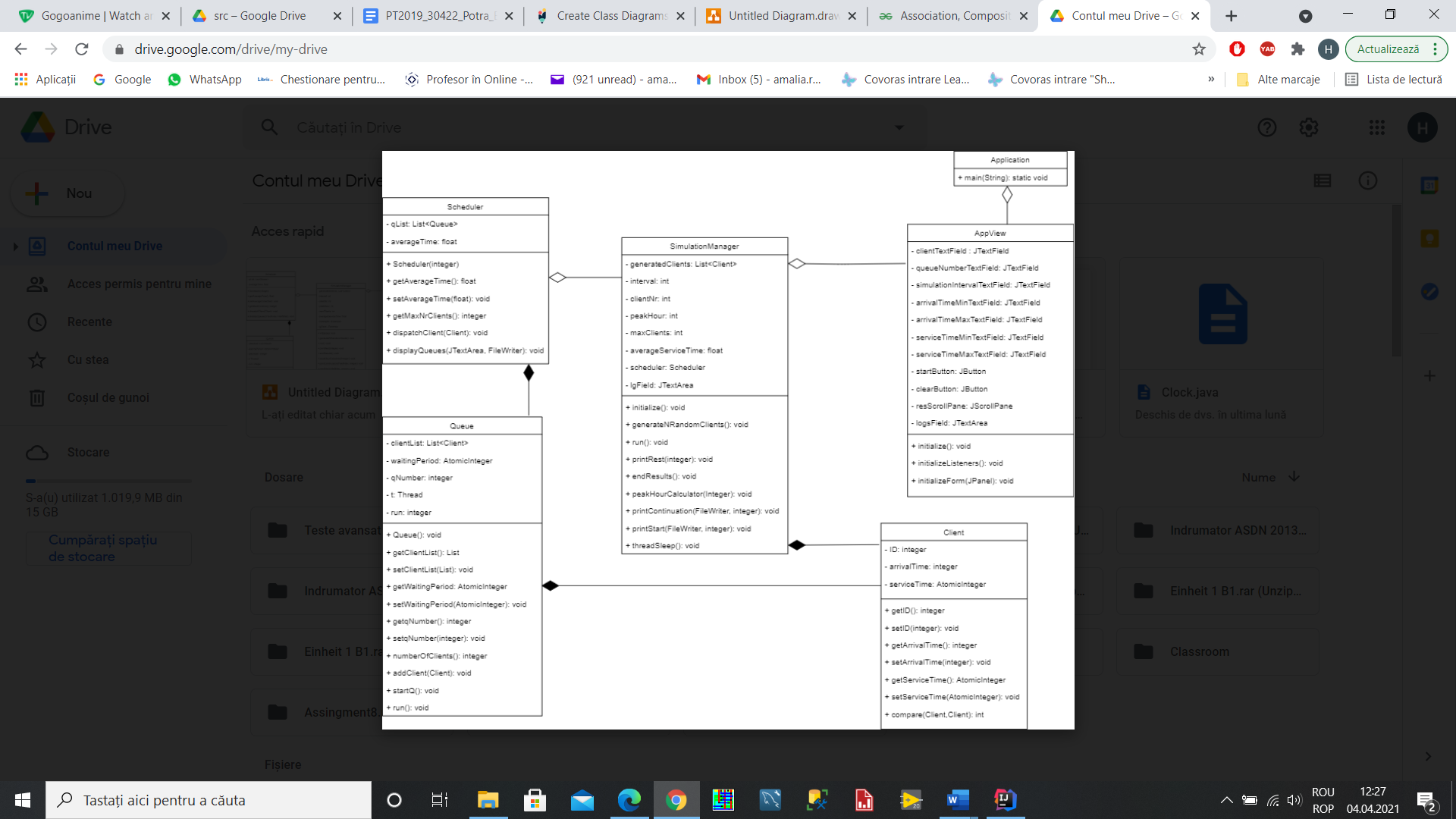
* The queue managment system should allow to insert the number of clients
* The queue managment system should allow to insert the number of queues
* The queue managment system should allow to insert the simulation interval
* The queue managment system should allow to insert the minimal arrival time
* The queue managment system should allow to insert the maximal arrival time
* The queue managment system should allow to insert the minimal service time
* The queue managment system should allow to insert the maximum service time
* The queue managment system should show the log of events as the program dispatches the clients to the queus
* The queue managment system should show the result of the computations

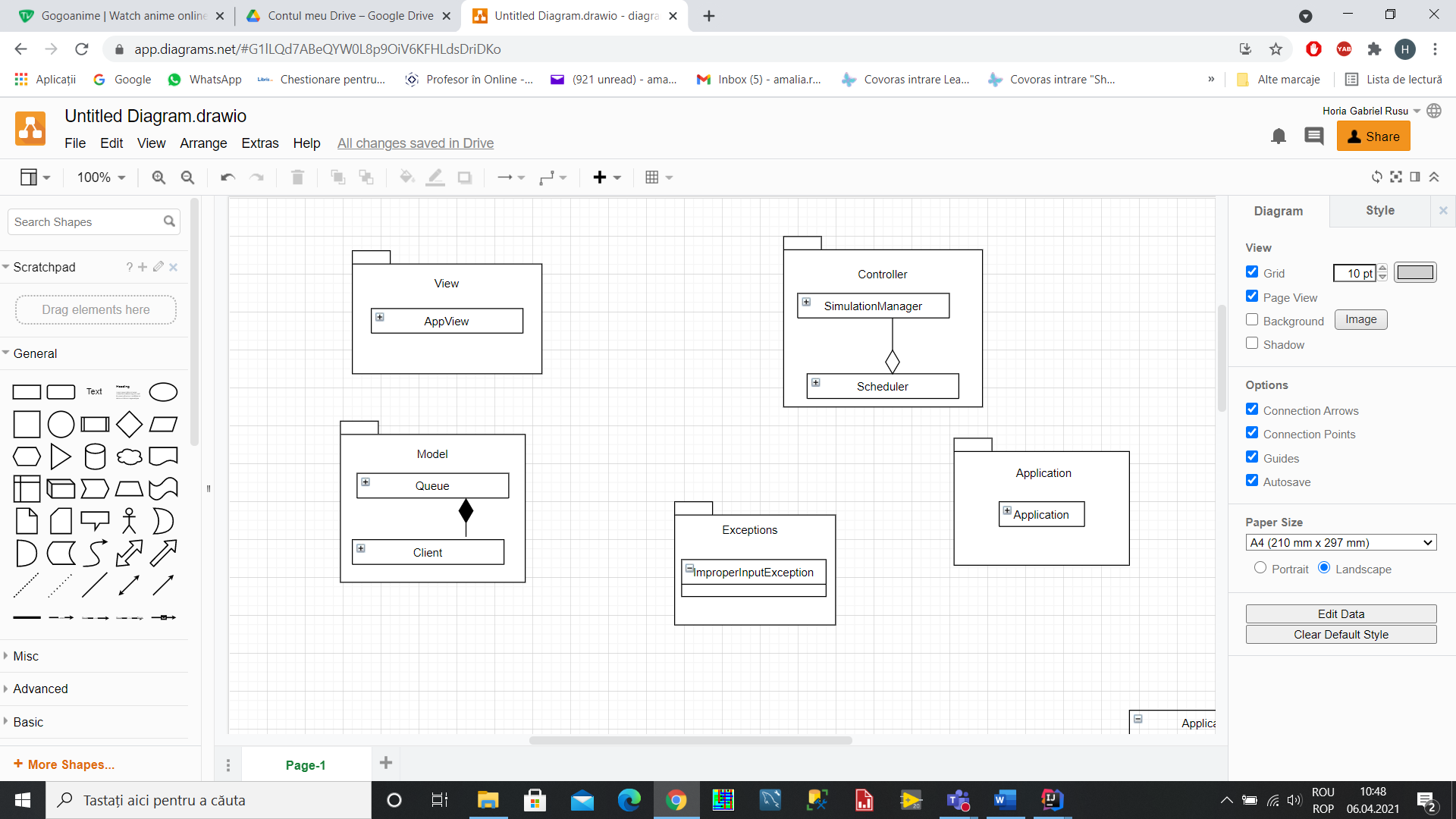
Non-functional Requirements:

* The queue managment system should be intuitive and easy to use by the user
* The queue managment system should display the result in a clear and easy to understand way

3 Design

1. Class Diagram





1. Data Structures

The data structures used range from the simple ones, such as the integers used in order to represent the Id or the arrival time of the client. The service time, on the other hand, also a variable of the client is represented via an atomic integer, a special class of integer used because of the need to assure thread safety when using multithreading to handle the queues. The threads are also special structures used, that implement run. These are used because of the realisation that queues work simultaneously, so they are used to show the working of multiple classes simultaneously without interfering with each other, but having a logical structute to them. Each queue has one or more clients, this represented by a array list, which holds the various clients. At the same time the queues are being held in a list of their own, also represented using the array list structure. ArrayLists are more useful than simple arrays because of the ease with which the new objects can be added. Also they are more efficient when it comes to the memory managmnet.

1. Main Algorithm pseudocode description

The main algorithm used should use the variables given in the input and computes randomely the needed data and orders it properly. Once the main thread starts it should go for as long there are clients that where not assigned to any queue. This clients will be compared with the current time in the simulation interval given and when the two of them are exactly the same, then the client will be dispatched to a queue based on the waiting periods of the other queues. The queue with the shortest waiting period will be chosen and the client will be added to that queue, and at the same time be removed form the waiting list. Afterwards the time will be incremented and the algorithm should wait for 1 second before resuming its activities. At the same time each queue should take each of the clients and wait for a period representing the number of the service time. Afterwards the client will be removed from the list and the queue will go on for as long

as there are people in the queue.

1. Implementation

Class Design: The class design has a divide and conquer method, through which the problem is divided into multiple smaller problems in order to be able to solve more simple, well-known problems instead of one big problem which seems too complicated to solve. The division of the program into classes is one part in the divide and conquer method and due to the pattern used, which is similar to the Model-View-Controller, we divide the problem into the before mentioned multiple packages. Thus the program consists in total of five parts, which are:

1. The Model: contains the logic of the application:
   1. The Client class:

The client class is one of the more important classes of the program. The client is the one that at first is in a waiting list and when his time comes he is inserted into the queue, where it will stay for the indicated time and then be removed, as it is not needed anymore. The variables used are the id, which is given to the client when he is initiated, the arrival time, an integer denoting the time in which the client arrives and must be integrated in a queue and is randomely chosen from the previously established bounds. The third variable, an atomic integer which represents the service time, also chosen at random from a certain interval, is an atomic integer, used so because of the need to decrement the service time without influencing the thread and keeping it from having synchronization problems.

Methods:

* public int getID : returns the id of the client;
* public void setID : sets the Id of the client;
* public int getArrivalTime : gets the arrival time of the client;
* public void setArrivalTime : sets the arrival time of the client;
* public AtomicInteger getServiceTime : gets the service time of the client;
* public void setServiceTime : sets the service time of the client;
* public static Comparator<Client> ArrivComp : compares and reorders the clients from the list according to the arrival time in increasing order.

* 1. The Queue class:

The queue class holds the clients and is the main point of the program. It contains an arraylist, the list of clients which must be evaluated in the main method, the waiting period, an atomicinteger used to see which queue is the shortest one when inserting a new client, and where to insert it. The qNumber is an integer, representing which Queue it is. The thread t is used to process the clients in the queue and the integer run is used to start the execution of the thread.

Methods:

* public Queue : constructor to setup the thread of the queue;
* public List<Client> getClientList : returns the list of clients of the queue;
* public void setClientList : sets the client list of the queue;
* public AtomicInteger getWaitingPeriod : gets the queues waiting period;
* public void setWaitingPeriod : sets the waiting period of the queue;
* public int getqNumber : returns the qNumber of the queue;
* public void setqNumber : sets the qNumber of the queue;
* public int numberOfClients: returns the number of clients of the queue;
* public void addClient : adds a client to the queue
* public void startQ : starts the thread of the queue
* public void run : runs the thread and processes each client of the queue

1. The Controller: creates relations between the View and the Model

1. SimulationManager:

Manages and creates the lists as well as their coordination when going to the queue. The variables are the array list generatedClients, the list of the clients in the waiting list. The integral interval, representing the time interval in which the simulation takes place, the clientNr, an integer showing the number of clients in the list, the integer peakHour, the integer maxClients, representing the maximum number of clients in the queues, the averageServiceTime and the JScroll to display the log.

public void initialize : initializes the SimulationManager by adding all the user induced parameters and using them in further functions;

private void generateNRandomClients : creates the list of clients using the number of clients from the input and randomely choosing the arrival and service time before crreating the client and adding it to the list;

public void run : starts running the thread, goes through the simulation interval and when the apropriate time comes the client is sent to the scheduler to be put in a queue and removed from the client list. At the same time the peak hour, average waiting time and average service time are computed through other functions;

public void printStart : prints the time and the list of clients;

public void peakHourCalculator : computes the peak hour of the simualtion;

public void threadSleep : puts the thread to sleep for a second;

public void printContinuation : prints the list of the clients and calls a function in the scheduler to display the queues

public void printRest : prints the queues after the client lis tis empty;

public void endResults : prints the service time, peak hour and medium waiting time;

1. Scheduler

The scheduler has a qList, a list of queues and an averageTime, which is used to compute the medium waiting time;

Scheduler : constructor used to set up each of the queues and add them to the qList;

public float getAverageTime : returns the average time of the scheduler;

public void setAverageTime : sets the average time of the scheduler;

public int getMaxNrClients : returns the number of clients in all the queues;

public void dispatchClient : dispatches the client to the queue depending on the waiting the time of each of the queue and choosing the smallest one;

public void displayQueues : displays the queues and all the clients in each of them;

1. The View : contains the graphical user interface
2. AppView

The class extends JFrame and has text fields for the number of clients, the number of queues, the minimum and maximum arrival time, the minimum and maximum service time and the simulations interval. There are also two buttons, the one for starting the simulation and the clear, for the clearing of the JTextArea.

Methods:

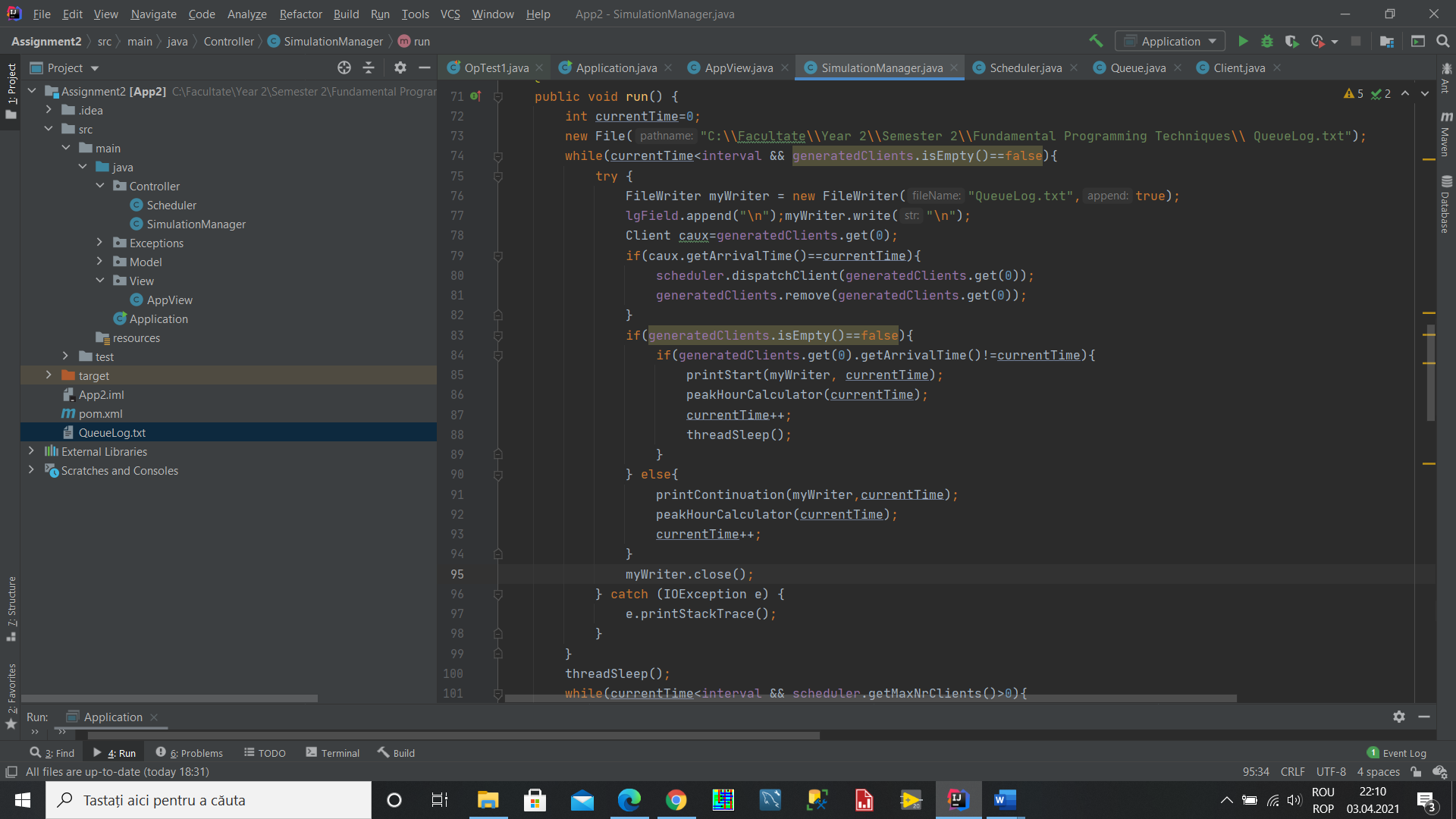
* public void initialize : initializes the user interface;
* private void initializeListeners : initializes the actions of the buttons;
* private void initializeForm : constructs the user interface;

1. The Application:

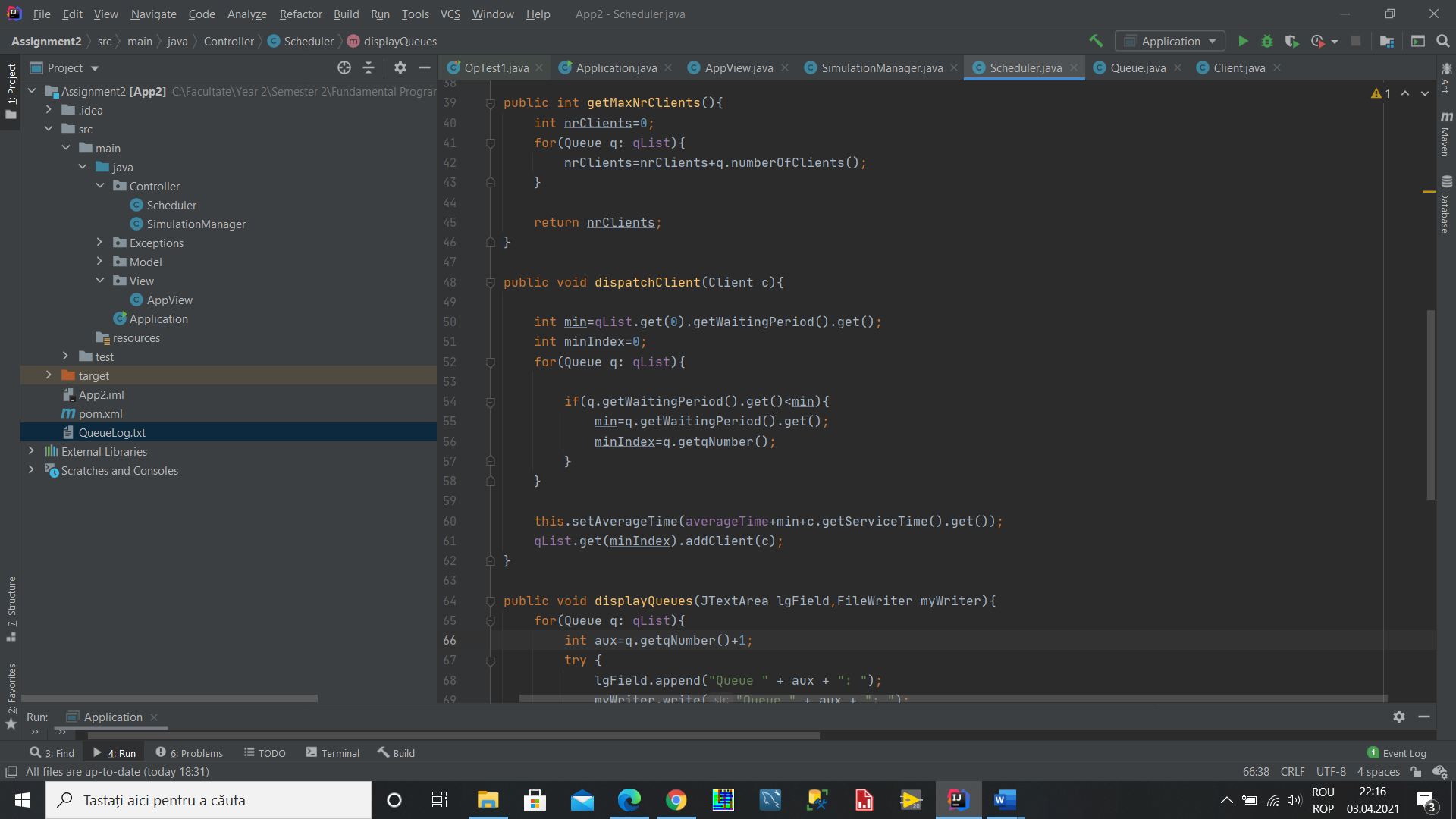
Initializes the entire application starting with the view.

Algorithms:

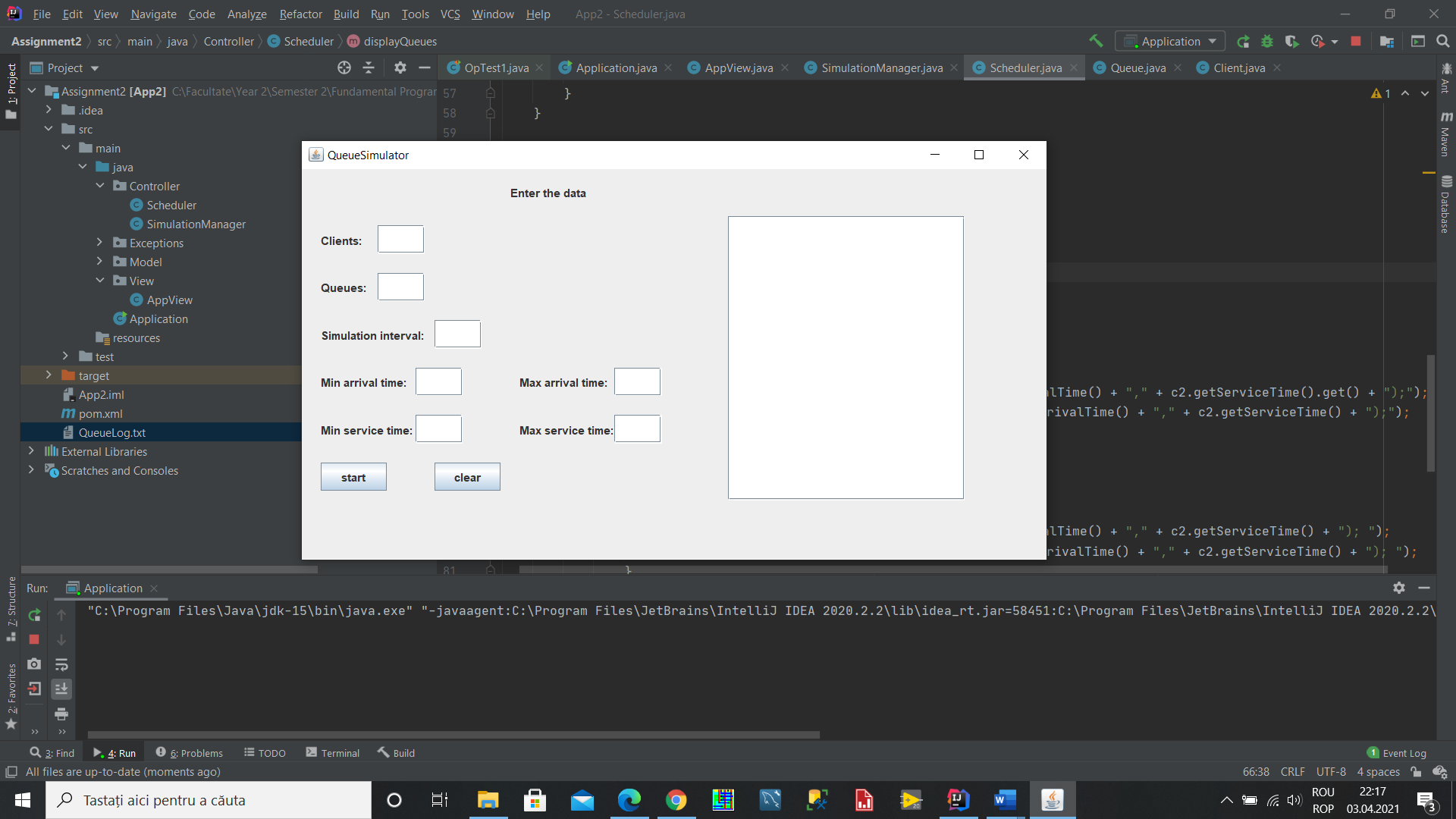
The algorithm for running the main thread initializes the cuurentTime with zero and then starts in the while, going whiel the currentTime is smaller than the time interval and while there are still clients waiting to be put in the queue. Client caux takes the first client from the list and if its arrival time coincides with the current time, then the scheduler is called to dispatch clients. Afterwards the client is removed. the algorithm checks if the next client exists and if he has the same arrival time than the current time is not incremented, otherwise it is. The command then calls the sleep method to sleep for 1 second.



The client dispatcher from the client gets the waiting period from the first Queue in the queue list. Then we parse through each queue in the list using a foreach structure and then comparing their waiting periods with the first one and if their waiting period is smaller than the current one, than that one is chosen as the minimum. At the end the index of the minimum is taken and the client is set in the queue with that index.

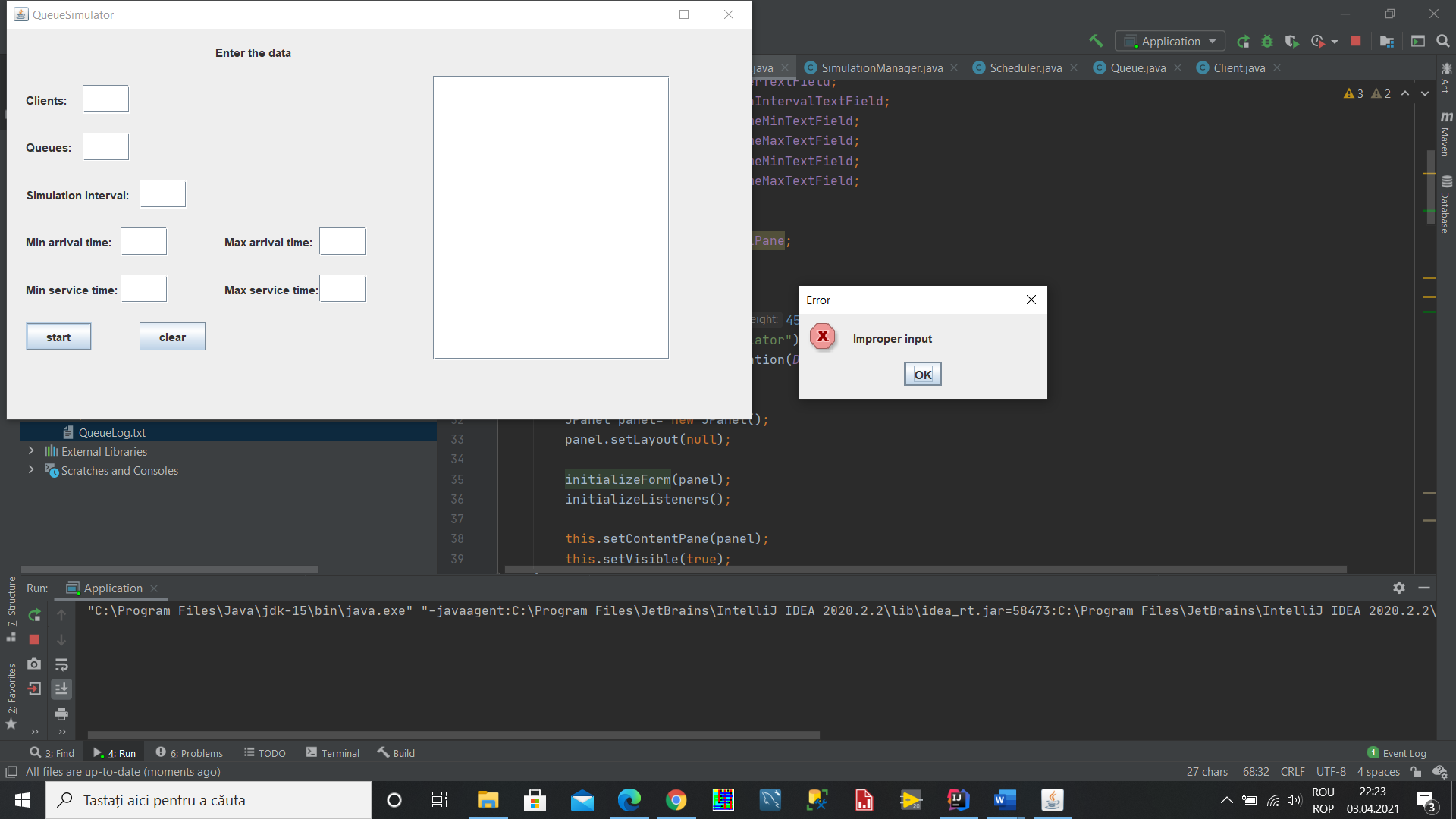


Graphical User Interface



The fields for data are the ones marked with each of the name of the needed variables, such as the client field for the number of clients, the number of queues for the simualtion, the minimum and maximum arrival and service times and the simulation interval itself. Also the logField on the left for the log of events.

When there is a problem with the input a pop-up screen will appear featuring the Message that the input is not proper.



5.) Results

The application is run on the three input tests presented below:

|  |  |  |
| --- | --- | --- |
| Test 1 | Test 2 | Test 3 |
| N = 4  Q = 2  𝑡𝑠𝑖𝑚𝑢𝑙𝑎𝑡𝑖𝑜𝑛=60seconds  [𝑡𝑎𝑟𝑟𝑖𝑣𝑎𝑙 𝑀𝐼𝑁 , 𝑡𝑎𝑟𝑟𝑖𝑣𝑎𝑙 𝑀𝐴𝑋 ] = [2, 30]  [𝑡𝑠𝑒𝑟𝑣𝑖𝑐𝑒 𝑀𝐼𝑁 , 𝑡𝑠𝑒𝑟𝑣𝑖𝑐𝑒 𝑀𝐴𝑋 ] = [2, 4] | N = 50  Q = 5  𝑡𝑠𝑖𝑚𝑢𝑙𝑎𝑡𝑖𝑜𝑛=60seconds [𝑡𝑎𝑟𝑟𝑖𝑣𝑎𝑙 𝑀𝐼𝑁 , 𝑡𝑎𝑟𝑟𝑖𝑣𝑎𝑙 𝑀𝐴𝑋 ] = [2, 40]  [𝑡𝑠𝑒𝑟𝑣𝑖𝑐𝑒 𝑀𝐼𝑁 , 𝑡𝑠𝑒𝑟𝑣𝑖𝑐𝑒 𝑀𝐴𝑋 ] = [1, 7] | N = 1000  Q = 20  𝑡𝑠𝑖𝑚𝑢𝑙𝑎𝑡𝑖𝑜𝑛=200seconds [𝑡𝑎𝑟𝑟𝑖𝑣𝑎𝑙 𝑀𝐼𝑁 , 𝑡𝑎𝑟𝑟𝑖𝑣𝑎𝑙 𝑀𝐴𝑋 ] = [10, 100]  [𝑡𝑠𝑒𝑟𝑣𝑖𝑐𝑒 𝑀𝐼𝑁 , 𝑡𝑠𝑒𝑟𝑣𝑖𝑐𝑒 𝑀𝐴𝑋 ] = [3, 9] |

The results of the tests show that the program is properly taking all the clients and putting them in the queues according to the waiting period and discarding them after the service time has expired.

6.) Conclusions

This assignment was very good for remembering some OOP concepts learned in the first semetser but also learning new ones. In this regard, the use of threads was something new and their managment was the most challenging part of the assignment. It was important to properly organize the code and take care so that there will not be any problems with the threads and their execution. The sleep time was important and the coordination between the main thread and the various queue threads was important, but also between the threads and the output and the coordination between all of them in order to have to proper output at the proper time.

In the future more implementations could be done to expand upon the project. There could be more options presented for choosing the queue, other than the minimal waiting period, such as the number of people in a queue.

7.) Bibliography

<https://www.w3schools.com/java/java_threads.asp>

<https://www.tutorialspoint.com/java/java_multithreading.htm>

<https://howtodoinjava.com/java/multi-threading/atomicinteger-example/>

https://docs.microsoft.com/en-us/dotnet/api/system.threading.thread.sleep?view=net-5.0

Object-Oriented Programming - Lecture Slides of prof. Marius JOLDOS