Assignment 2 Documentation

Queues Simulator

Grama Mălina Bianca

Group 30423

Teaching Assistant: Antal Marcel

Table of Contents

1. Assignment Objectives

The objective of the queue simulator is to model the real-life scenario where a number of clients will wait in lines for some services. The application will work by minimizing the waiting time of the clients, by placing them in the queue with the minimum waiting time.

a. Primary Objective

The primary objective of this second assignment is to propose, design and implement a system that is able to build this queue simulation with variable numbers of clients and queues.

b. Secondary Objectives

The secondary objectives of the assignment that I followed while implementing my solution would be the following:

- **Development of use cases and scenarios** – we need to think about how the user will interact with the application and what scenarios could arise. This will be described in detail in the **problem analysis** section of this documentation.
- **Choosing the data structures** – we need to think about how the user input will be stored and managed by our application, so that it will generate the required result. This will be briefly mentioned in the **design** section of the documentation.
- **Division of the solution into classes** – we need to think about how we will model the real-life "objects" that are the clients and the queues into an object-oriented programming object. This will be mentioned in the **design** section of this documentation and detailed in the **implementation** section.
- **Declaration of variables and methods** into aforementioned classes – after we figure out the classes that we need, we also need to think about the variables and methods that will go into these classes. Most of the times we start from the project specification, looking for nouns, which become possible candidate classes, and verbs that could play the role of class methods. This will be detailed in the **implementation** section.
- **Development of the algorithms** necessary for the simulation – we need to think about how we will implement the threads responsible for the simulation. This objective will be detailed in the **implementation** section of the documentation.
- **Implementation of the solution** – we will need to describe more specifically now each class used in the project. Also, we will describe the implementation of the user interface. This will be detailed in the **implementation** section of this documentation.
- **Testing** – detailed in the **testing and results** section.
- **Debugging**.
- **More testing**.

2. Problem Analysis

By analyzing the problem, we refer to a first abstract set of operations and properties through which we try to detect possible features and behaviors of unknown processes. Object-oriented programming offers us a clear advantage here, precisely because it allows us to tackle the problem from a higher level, without being constrained, to such an extent, by the technical characteristics.

Below we will exemplify a use case for the queue simulator:

Use Case Name: Queue Simulation

Primary Actor: User

Triggers: The user indicated that he wants to perform the simulation by clicking the "Start Simulation" button on the user interface.

Preconditions: The user has typed all of the necessary variables into the text fields: number of clients that will be generated, number of queues that will be in use, the maximum time of the simulation, the interval representing the arrival times of the clients, and the interval for the service time for the clients.

Normal Flow:

- The user inserts the required numbers in the graphical user interface.
- The user presses the "Start Simulation" button on the graphical interface.
- The Queues Simulator performs the simulation and displays the status of the pool of waiting clients and the queues as the simulation time goes from 0 to the specified maximum simulation time.

Alternate Flows:

- The user inserts the required numbers incorrectly (an example of an error in the input would be to specify a minimum arrival time that is bigger than the maximum arrival time, or typing letters instead of 0-9 digits etc.).
- The user presses the "Start Simulation" button on the graphical interface.
- The Queues Simulator checks the input fields and displays an error message saying that the input was typed incorrectly, without starting the simulation.
- The scenario returns to the first step.

We need to define a set of functional and non-functional requirements, to ensure that we cover all of the cases and provide the functionality of the application.

Some of the functional requirements could be:

- The queues simulator should allow users to insert the inputs required to run the simulation.
- The queues simulator should check if the input of the user has the correct synta, displaying an error message in the eventuality that the user typed illegal inputs.
- The queues simulator should perform correctly the simulation with the provided input.
- The queues simulator should display the correct results of the simulation in "real time", and display the status of the pool of waiting clients and queues in a .txt file, while also displaying statistics like average waiting time, average service time and peak hour.

Some of the non-functional requirements of the application:

- The queues simulator should be easy to use and intuitive.
- The results should be clearly stated so that they are easily understood by the user.

3. Design

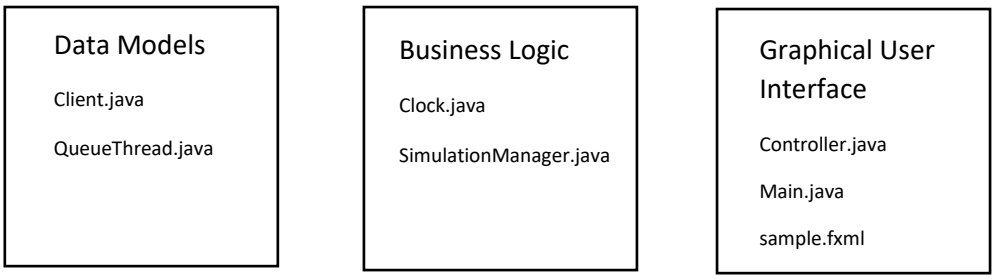
We try to think of the solution in a bottom-up manner, dividing the components of the problem into smaller pieces. As we advance to lower levels, the complexity of the problem increases.

In our example, it is clear from the start that we are going to need a class that will represent the clients, and a class representing the queues. The queue class will be implemented with the help of threads, so that the processing of clients will be done faster (concurrently). So, we implement the classes **Client** and **QueueThread**.

After this analysis, we decide that we are going to need another class that will play the role of a timer. This class will represent a thread also, and it will be controlled by a class that we will be naming SimulationManager. The manager will process the input typed by the user and will run the simulator until there are no more clients, or until the maximum simulation time is reached. Also, the SimulationManager will be generating the specified number of clients at random, and it will control all of the other threads. So, we have two more classes, **Clock** and **SimulationManager**.

Using an architectural pattern, we know that we need three types of classes in our design: **Data Models** that contains the classes modeling the application data, **Business Logic** that contains the classes implementing the functionality of the application, and finally, the **Graphical User Interface** that contains the classes and files implementing the graphical user interface.

Regarding the Graphical User Interface, we will use JavaFX to implement the Queues Simulator. Therefore, we will need a **Controller** class that will handle the functionality of the elements placed in the user interface, a **Main** class that will be the driver of our application and will hold the main method, and an fxml file that will contain information about our GUI and will link the GUI with the Controller class.



4. Implementation

a. Back-End

Now that we have decided what classes are needed for our project, let's talk briefly about the main methods of these Java Classes.

We will start with the class that models the real-life clients, namely the **Client** class. This class has the following parameters:

- int ID: represent a number that identifies the clients. The ID will be in the range [1,N], where N represents the total number of clients.
- int arrivalTime: represents the time in which the clients will join one of the available queues (the queue with the shortest waiting time). The arrivalTime is generated at random, and has the range [tMinArrival, tMaxArrival], where the bounds of this interval are inputs from the user.
- int processingTime: represents how long will the client stay at the front of the queue. For example, if we think about the process of waiting in line at a supermarket, the processingTime represents the time it takes to scan the products and pay for them. The processingTime is also generated at random, and has the bounds inserted by the user in the graphical user interface.
- int waitingTime: the time the client stays in a queue until he reaches the front of the queue. The waitingTime is used in computing the average waiting time for all of the clients in the simulation
- int serviceTime: represents the sum of the waitingTime and the processingTime (the total time spent by the client in the queue), and is used for computing the average service time at the end of the simulation.

Besides the setters and getters for some of these variables, the Client class does not contain any complex methods.

We will now describe the implementation of the **QueueThread** class. The variables of this class are:

- List<Client> clientList: a list of the clients that are in the queue at a specific moment of the simulation.
- int queueNumber: the number of the queue, ranging from [1,Q], where Q is the number of the queues that are in use, gave by the user at the start of the simulation.
- boolean isRunning: the queue Thread will run while this variable is set to true.
- int nrClients: the number of clients in the queue at a specific time of the simulation.
- int totalClients: the number of all of the clients in the queue throughout all simulation.
- Clock clock: represents the Clock Thread, and is useful for getting the current time of the simulation.
- Thread queue.

The methods of this class described briefly:

- *void start()* and *void stop()* will set the isRunning boolean to true/ false. In the start method, the queue thread will also be started.
- *void addClient()* and *void removeClient()* will add/remove clients from the clientList, while also modifying the totalClients and nrClients variables accordingly.

- *int getServiceTime()* returns the total service time required to serve all of the clients in the queue at a specific time of the simulation.
- *void run()* controls the queue Thread. The thread runs while the isRunning boolean is set to true, or while the client list is not empty. At each second of the simulation, we decrement the service time of the clients in the queue. When this time is equal to zero, the clients are removed from the front of the queue. The queue Thread sleeps for a total of 1000 milliseconds.

The **Clock** class is controlling the current time of the simulation. Its variables are int time, representing the current time, and boolean isRunning. The methods of this class are void *startClock()* and *void stopClock()* that start and stop the running of the thread, *int getTime()* that returns the current time to be used in the SimulationManager, and the override of the *void run()* method. The clock thread runs as long as isRunning is set to true, and increments the time variable every time it "sleeps" for 1000 milliseconds.

The **SimulationManager** class is the "brain" of the whole simulation, controlling all of the running threads and dividing the customers in the queues with the smallest waiting time. Its variables are:

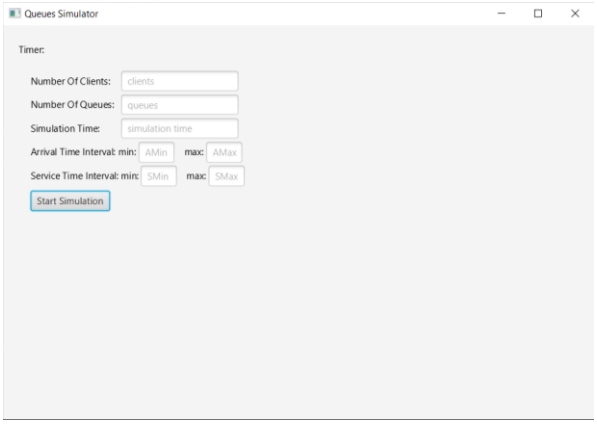
- int N: total number of clients at the start of the simulation.
- int Q: number of queues that will be in use during the simulation.
- int tMaxSimulation: the maximum time that the simulation will run.
- int tMinArrival, tMaxArrival: the bounds of the interval from which the arrival time of each client will be generated.
- int tMinService, tMaxService: the bounds of the interval from which the service time of each client will be generated.
- Clock clock.
- Thread simulation: the thread representing the SimulationManager thread.
- QueueThread[] queues: threads representing the queues.
- ArrayList<Client> generatedClients: this holds the initial list of the N clients that are generated at random
- ArrayList<Client> waitingClients: list of the clients that did not go to the queues yet.

The main methods of the SimulationManager class are described briefly bellow, as well as in the comments of the code.

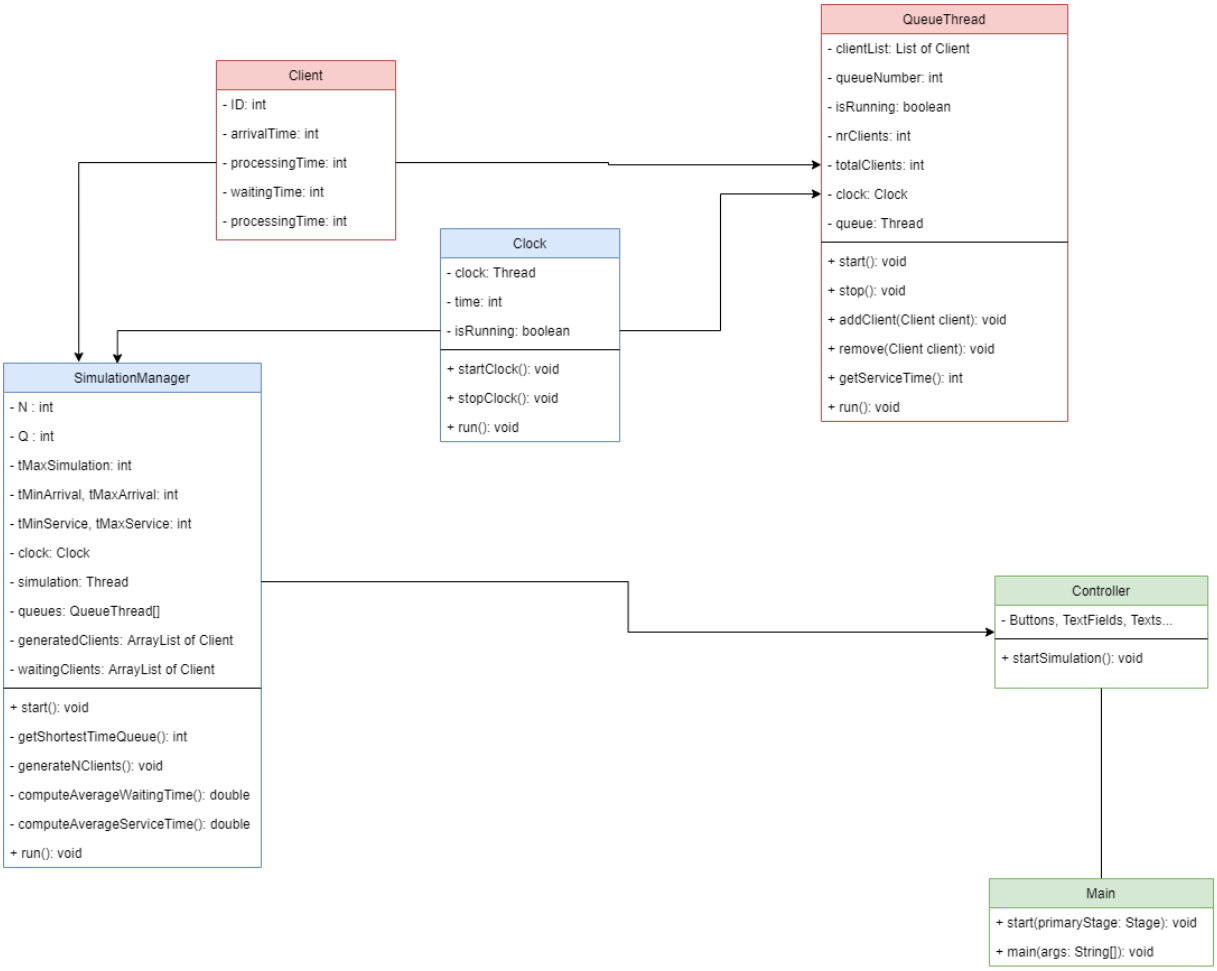
- *void start():* initializing and starting the simulation. Initializes the clock, the queue threads, and starts the simulation thread. Also calls the generateNClients() method described below.
- *int getShortestQueue()*: gets the index of the queue that has the smallest waiting time out of all the queues, so that the waiting clients that are ready to go the queues will be directed there.
- *void generateNClients()*: generates N clients and sorts them with respect to their arrival time.
- *double computeAverageWaitingTime()*: tracks the total time spent by every client until they reach the front of the queue and computes the average waiting time.
- *double computeAverageServiceTime()*: tracks the total time spent by every client in the queues and computes the average service time.
- *void run()*: the brains of the operation. Runs while the current time is smaller than the max simulation time OR while the waiting clients list and the queues are not empty. At each second of the simulation, it displays the Time of the simulation, the list with the waiting clients and the statuses of all the queues that are used. Also decides if the waiting clients are ready to go to the queues and directs them to the queue with the smallest waiting time. At the end of the simulation, closes the Clock thread and the threads of the queues, and displays the average waiting time, the average service time and the peak hour. The peak hour is computed inside of the run() method.

b. Front-End

I implemented the Graphical User Interface using JavaFX and SceneBuilder. The user is able to input the parameters of the simulation through the GUI. The input part of the application looks like this:



Once the user types the parameters of the simulation correctly and presses the Start Simulation button, the application will start running. A .txt file with the results will be generated.



UML Diagram of the project

5. Testing and Results

For the application testing I used the input data sets from the table below, and I displayed the results in three .txt files named test1.txt, test2.txt and test3.txt.

Test 1	Test 2	Test 3
N = 4 Q = 2 $t^{MAX}_{simulation} = 60$ seconds $[t^{MIN}_{arrival}, t^{MAX}_{arrival}] = [2, 30]$ $[t^{MIN}_{service}, t^{MAX}_{service}] = [2, 4]$	N = 50 Q = 5 $t^{MAX}_{simulation} = 60$ seconds $[t^{MIN}_{arrival}, t^{MAX}_{arrival}] = [2, 40]$ $[t^{MIN}_{service}, t^{MAX}_{service}] = [1, 7]$	N = 1000 Q = 20 $t^{MAX}_{simulation} = 200$ seconds $[t^{MIN}_{arrival}, t^{MAX}_{arrival}] = [10, 100]$ $[t^{MIN}_{service}, t^{MAX}_{service}] = [3, 9]$

I will copy-paste the results in these files bellow:

test1.txt

Time 0

Waiting clients: (2, 6, 4); (1, 11, 3); (3, 24, 2); (4, 27, 3);

Queue 1: closed

Queue 2: closed

Time 1

Waiting clients: (2, 6, 4); (1, 11, 3); (3, 24, 2); (4, 27, 3);

Queue 1: closed

Queue 2: closed

Time 2

Waiting clients: (2, 6, 4); (1, 11, 3); (3, 24, 2); (4, 27, 3);

Queue 1: closed

Queue 2: closed

Time 3

Waiting clients: (2, 6, 4); (1, 11, 3); (3, 24, 2); (4, 27, 3);

Queue 1: closed

Queue 2: closed

Time 4

Waiting clients: (2, 6, 4); (1, 11, 3); (3, 24, 2); (4, 27, 3);

Queue 1: closed

Queue 2: closed

Time 5

Waiting clients: (2, 6, 4); (1, 11, 3); (3, 24, 2); (4, 27, 3);

Queue 1: closed

Queue 2: closed

Time 6

Waiting clients: (1, 11, 3); (3, 24, 2); (4, 27, 3);

Queue 1: (2, 6, 4);

Queue 2: closed

Time 7

Waiting clients: (1, 11, 3); (3, 24, 2); (4, 27, 3);

Queue 1: (2, 6, 3);

Queue 2: closed

Time 8

Waiting clients: (1, 11, 3); (3, 24, 2); (4, 27, 3);

Queue 1: (2, 6, 2);

Queue 2: closed

Time 9

Waiting clients: (1, 11, 3); (3, 24, 2); (4, 27, 3);

Queue 1: (2, 6, 1);

Queue 2: closed

Time 10

Waiting clients: (1, 11, 3); (3, 24, 2); (4, 27, 3);

Queue 1: closed

Queue 2: closed

Time 11

Waiting clients: (3, 24, 2); (4, 27, 3);

Queue 1: (1, 11, 3);

Queue 2: closed

Time 12

Waiting clients: (3, 24, 2); (4, 27, 3);

Queue 1: (1, 11, 2);

Queue 2: closed

Time 13

Waiting clients: (3, 24, 2); (4, 27, 3);

Queue 1: (1, 11, 1);

Queue 2: closed

Time 14

Waiting clients: (3, 24, 2); (4, 27, 3);

Queue 1: closed

Queue 2: closed

Time 15

Waiting clients: (3, 24, 2); (4, 27, 3);

Queue 1: closed

Queue 2: closed

Time 16

Waiting clients: (3, 24, 2); (4, 27, 3);

Queue 1: closed

Queue 2: closed

Time 17

Waiting clients: (3, 24, 2); (4, 27, 3);

Queue 1: closed

Queue 2: closed

Time 18

Waiting clients: (3, 24, 2); (4, 27, 3);

Queue 1: closed

Queue 2: closed

Time 19

Waiting clients: (3, 24, 2); (4, 27, 3);

Queue 1: closed

Queue 2: closed

Time 20

Waiting clients: (3, 24, 2); (4, 27, 3);

Queue 1: closed

Queue 2: closed

Time 21

Waiting clients: (3, 24, 2); (4, 27, 3);

Queue 1: closed

Queue 2: closed

Time 22

Waiting clients: (3, 24, 2); (4, 27, 3);

Queue 1: closed

Queue 2: closed

Time 23

Waiting clients: (3, 24, 2); (4, 27, 3);

Queue 1: closed

Queue 2: closed

Time 24

Waiting clients: (4, 27, 3);

Queue 1: (3, 24, 2);

Queue 2: closed

Time 25

Waiting clients: (4, 27, 3);

Queue 1: (3, 24, 1);

Queue 2: closed

Time 26

Waiting clients: (4, 27, 3);

Queue 1: closed

Queue 2: closed

Time 27

Waiting clients:

Queue 1: (4, 27, 3);

Queue 2: closed

Time 28

Waiting clients:

Queue 1: (4, 27, 2);

Queue 2: closed

Time 29

Waiting clients:

Queue 1: (4, 27, 1);

Queue 2: closed

Time 30

Waiting clients:

Queue 1: closed

Queue 2: closed

Simulation stopped at time 30

Average waiting time: 1.0

Average service time: 4.0

Peak hour: 1 customers at time 6

test2.txt

Time 0

Waiting clients: (19, 2, 6); (29, 4, 5); (41, 4, 3); (11, 5, 7); (46, 5, 6); (43, 6, 7); (5, 6, 6); (3, 7, 1); (28, 7, 4); (24, 9, 1); (39, 11, 7); (14, 11, 5); (47, 11, 4); (50, 11, 6); (10, 13, 7); (23, 14, 6); (25, 14, 7); (31, 14, 6); (32, 15, 3); (34, 15, 2); (22, 16, 5); (13, 17, 2); (38, 17, 3); (40, 18, 5); (36, 18, 7); (49, 19, 2); (8, 19, 5); (9, 20, 7); (35, 21, 7); (4, 22, 1); (44, 24, 4); (20, 25, 5); (37, 25, 4); (21, 25, 3); (15, 25, 3); (6, 26, 2); (48, 26, 3); (30, 26, 7); (2, 26, 7); (12, 27, 5); (27, 28, 4); (26, 29, 5); (1, 31, 3); (7, 31, 1); (33, 31, 3); (17, 36, 7); (18, 36, 6); (16, 37, 1); (42, 39, 7); (45, 40, 4);

Queue 1: closed

Queue 2: closed

Queue 3: closed

Queue 4: closed

Queue 5: closed

Time 1

Waiting clients: (19, 2, 6); (29, 4, 5); (41, 4, 3); (11, 5, 7); (46, 5, 6); (43, 6, 7); (5, 6, 6); (3, 7, 1); (28, 7, 4); (24, 9, 1); (39, 11, 7); (14, 11, 5); (47, 11, 4); (50, 11, 6); (10, 13, 7); (23, 14, 6); (25, 14, 7); (31, 14, 6); (32, 15, 3); (34, 15, 2); (22, 16, 5); (13, 17, 2); (38, 17, 3); (40, 18, 5); (36, 18, 7); (49, 19, 2); (8, 19, 5); (9, 20, 7); (35, 21, 7); (4, 22, 1); (44, 24, 4); (20, 25, 5); (37, 25, 4); (21, 25, 3); (15, 25, 3); (6, 26, 2); (48, 26, 3); (30, 26, 7); (2, 26, 7); (12, 27, 5); (27, 28, 4); (26, 29, 5); (1, 31, 3); (7, 31, 1); (33, 31, 3); (17, 36, 7); (18, 36, 6); (16, 37, 1); (42, 39, 7); (45, 40, 4);

Queue 1: closed

Queue 2: closed

Queue 3: closed

Queue 4: closed

Queue 5: closed

Time 2

Waiting clients: (29, 4, 5); (41, 4, 3); (11, 5, 7); (46, 5, 6); (43, 6, 7); (5, 6, 6); (3, 7, 1); (28, 7, 4); (24, 9, 1); (39, 11, 7); (14, 11, 5); (47, 11, 4); (50, 11, 6); (10, 13, 7); (23, 14, 6); (25, 14, 7); (31, 14, 6); (32, 15, 3); (34, 15, 2); (22, 16, 5); (13, 17, 2); (38, 17, 3); (40, 18, 5); (36, 18, 7); (49, 19, 2); (8, 19, 5); (9, 20, 7); (35, 21, 7); (4, 22, 1); (44, 24, 4); (20, 25, 5); (37, 25, 4); (21, 25, 3); (15, 25, 3); (6, 26, 2); (48, 26, 3); (30, 26, 7); (2, 26, 7); (12, 27, 5); (27, 28, 4); (26, 29, 5); (1, 31, 3); (7, 31, 1); (33, 31, 3); (17, 36, 7); (18, 36, 6); (16, 37, 1); (42, 39, 7); (45, 40, 4);

Queue 1: (19, 2, 6);

Queue 2: closed

Queue 3: closed

Queue 4: closed

Queue 5: closed

Time 3

Waiting clients: (29, 4, 5); (41, 4, 3); (11, 5, 7); (46, 5, 6); (43, 6, 7); (5, 6, 6); (3, 7, 1); (28, 7, 4); (24, 9, 1); (39, 11, 7); (14, 11, 5); (47, 11, 4); (50, 11, 6); (10, 13, 7); (23, 14, 6); (25, 14, 7); (31, 14, 6); (32, 15, 3); (34, 15, 2); (22, 16, 5); (13, 17, 2); (38, 17, 3); (40, 18, 5); (36, 18, 7); (49, 19, 2); (8, 19, 5); (9, 20, 7); (35, 21, 7); (4, 22, 1); (44, 24, 4); (20, 25, 5); (37, 25, 4); (21, 25, 3); (15, 25, 3); (6, 26, 2); (48, 26, 3); (30, 26, 7); (2, 26, 7); (12, 27,

5); (27, 28, 4); (26, 29, 5); (1, 31, 3); (7, 31, 1); (33, 31, 3); (17, 36, 7); (18, 36, 6); (16, 37, 1); (42, 39, 7); (45, 40, 4);

Queue 1: (19, 2, 5);

Queue 2: closed

Queue 3: closed

Queue 4: closed

Queue 5: closed

Time 4

Waiting clients: (11, 5, 7); (46, 5, 6); (43, 6, 7); (5, 6, 6); (3, 7, 1); (28, 7, 4); (24, 9, 1); (39, 11, 7); (14, 11, 5); (47, 11, 4); (50, 11, 6); (10, 13, 7); (23, 14, 6); (25, 14, 7); (31, 14, 6); (32, 15, 3); (34, 15, 2); (22, 16, 5); (13, 17, 2); (38, 17, 3); (40, 18, 5); (36, 18, 7); (49, 19, 2); (8, 19, 5); (9, 20, 7); (35, 21, 7); (4, 22, 1); (44, 24, 4); (20, 25, 5); (37, 25, 4); (21, 25, 3); (15, 25, 3); (6, 26, 2); (48, 26, 3); (30, 26, 7); (2, 26, 7); (12, 27, 5); (27, 28, 4); (26, 29, 5); (1, 31, 3); (7, 31, 1); (33, 31, 3); (17, 36, 7); (18, 36, 6); (16, 37, 1); (42, 39, 7); (45, 40, 4);

Queue 1: (19, 2, 4);

Queue 2: (29, 4, 5);

Queue 3: (41, 4, 3);

Queue 4: closed

Queue 5: closed

Time 5

Waiting clients: (43, 6, 7); (5, 6, 6); (3, 7, 1); (28, 7, 4); (24, 9, 1); (39, 11, 7); (14, 11, 5); (47, 11, 4); (50, 11, 6); (10, 13, 7); (23, 14, 6); (25, 14, 7); (31, 14, 6); (32, 15, 3); (34, 15, 2); (22, 16, 5); (13, 17, 2); (38, 17, 3); (40, 18, 5); (36, 18, 7); (49, 19, 2); (8, 19, 5); (9, 20, 7); (35, 21, 7); (4, 22, 1); (44, 24, 4); (20, 25, 5); (37, 25, 4); (21, 25, 3); (15, 25, 3); (6, 26, 2); (48, 26, 3); (30, 26, 7); (2, 26, 7); (12, 27, 5); (27, 28, 4); (26, 29, 5); (1, 31, 3); (7, 31, 1); (33, 31, 3); (17, 36, 7); (18, 36, 6); (16, 37, 1); (42, 39, 7); (45, 40, 4);

Queue 1: (19, 2, 3);

Queue 2: (29, 4, 4);

Queue 3: (41, 4, 2);

Queue 4: (11, 5, 7);

Queue 5: (46, 5, 6);

Time 6

Waiting clients: (3, 7, 1); (28, 7, 4); (24, 9, 1); (39, 11, 7); (14, 11, 5); (47, 11, 4); (50, 11, 6); (10, 13, 7); (23, 14, 6); (25, 14, 7); (31, 14, 6); (32, 15, 3); (34, 15, 2); (22, 16, 5); (13, 17, 2); (38, 17, 3); (40, 18, 5); (36, 18, 7); (49, 19, 2); (8, 19, 5); (9, 20, 7); (35, 21, 7); (4, 22, 1); (44, 24, 4); (20, 25, 5); (37, 25, 4); (21, 25, 3); (15, 25, 3); (6, 26, 2); (48, 26, 3); (30, 26, 7); (2, 26, 7); (12, 27, 5); (27, 28, 4); (26, 29, 5); (1, 31, 3); (7, 31, 1); (33, 31, 3); (17, 36, 7); (18, 36, 6); (16, 37, 1); (42, 39, 7); (45, 40, 4);

Queue 1: (19, 2, 2); (5, 6, 6);

Queue 2: (29, 4, 3);

Queue 3: (41, 4, 1); (43, 6, 7);

Queue 4: (11, 5, 6);

Queue 5: (46, 5, 5);

Time 7

Waiting clients: (24, 9, 1); (39, 11, 7); (14, 11, 5); (47, 11, 4); (50, 11, 6); (10, 13, 7); (23, 14, 6); (25, 14, 7); (31, 14, 6); (32, 15, 3); (34, 15, 2); (22, 16, 5); (13, 17, 2); (38, 17, 3); (40, 18, 5); (36, 18, 7); (49, 19, 2); (8, 19, 5); (9, 20, 7); (35, 21, 7); (4, 22, 1); (44, 24, 4); (20, 25, 5); (37, 25, 4); (21, 25, 3); (15, 25, 3); (6, 26, 2); (48, 26, 3); (30, 26, 7); (2, 26, 7); (12, 27, 5); (27, 28, 4); (26, 29, 5); (1, 31, 3); (7, 31, 1); (33, 31, 3); (17, 36, 7); (18, 36, 6); (16, 37, 1); (42, 39, 7); (45, 40, 4);

Queue 1: (19, 2, 1); (5, 6, 6);

Queue 2: (29, 4, 2); (3, 7, 1); (28, 7, 4);

Queue 3: (43, 6, 6);

Queue 4: (11, 5, 5);

Queue 5: (46, 5, 4);

Time 8

Waiting clients: (24, 9, 1); (39, 11, 7); (14, 11, 5); (47, 11, 4); (50, 11, 6); (10, 13, 7); (23, 14, 6); (25, 14, 7); (31, 14, 6); (32, 15, 3); (34, 15, 2); (22, 16, 5); (13, 17, 2); (38, 17, 3); (40, 18, 5); (36, 18, 7); (49, 19, 2); (8, 19, 5); (9, 20, 7); (35, 21, 7); (4, 22, 1); (44, 24, 4); (20, 25, 5); (37, 25, 4); (21, 25, 3); (15, 25, 3); (6, 26, 2); (48, 26, 3); (30, 26, 7); (2, 26, 7); (12, 27, 5); (27, 28, 4); (26, 29, 5); (1, 31, 3); (7, 31, 1); (33, 31, 3); (17, 36, 7); (18, 36, 6); (16, 37, 1); (42, 39, 7); (45, 40, 4);

Queue 1: (5, 6, 5);

Queue 2: (29, 4, 1); (3, 7, 1); (28, 7, 4);

Queue 3: (43, 6, 5);

Queue 4: (11, 5, 4);

Queue 5: (46, 5, 3);

Time 9

Waiting clients: (39, 11, 7); (14, 11, 5); (47, 11, 4); (50, 11, 6); (10, 13, 7); (23, 14, 6); (25, 14, 7); (31, 14, 6); (32, 15, 3); (34, 15, 2); (22, 16, 5); (13, 17, 2); (38, 17, 3); (40, 18, 5); (36, 18, 7); (49, 19, 2); (8, 19, 5); (9, 20, 7); (35, 21, 7); (4, 22, 1); (44, 24, 4); (20, 25, 5); (37, 25, 4); (21, 25, 3); (15, 25, 3); (6, 26, 2); (48, 26, 3); (30, 26, 7); (2, 26, 7); (12, 27, 5); (27, 28, 4); (26, 29, 5); (1, 31, 3); (7, 31, 1); (33, 31, 3); (17, 36, 7); (18, 36, 6); (16, 37, 1); (42, 39, 7); (45, 40, 4);

Queue 1: (5, 6, 4);

Queue 2: (28, 7, 3);

Queue 3: (43, 6, 4);

Queue 4: (11, 5, 3);

Queue 5: (46, 5, 2); (24, 9, 1);

Time 10

Waiting clients: (39, 11, 7); (14, 11, 5); (47, 11, 4); (50, 11, 6); (10, 13, 7); (23, 14, 6); (25, 14, 7); (31, 14, 6); (32, 15, 3); (34, 15, 2); (22, 16, 5); (13, 17, 2); (38, 17, 3); (40, 18, 5); (36, 18, 7); (49, 19, 2); (8, 19, 5); (9, 20, 7); (35, 21, 7); (4, 22, 1); (44, 24, 4); (20, 25, 5); (37, 25, 4); (21, 25, 3); (15, 25, 3); (6, 26, 2); (48, 26, 3); (30, 26, 7); (2, 26, 7); (12, 27, 5); (27, 28, 4); (26, 29, 5); (1, 31, 3); (7, 31, 1); (33, 31, 3); (17, 36, 7); (18, 36, 6); (16, 37, 1); (42, 39, 7); (45, 40, 4);

Queue 1: (5, 6, 3);

Queue 2: (28, 7, 2);

Queue 3: (43, 6, 3);

Queue 4: (11, 5, 2);

Queue 5: (46, 5, 1); (24, 9, 1);

Time 11

Waiting clients: (10, 13, 7); (23, 14, 6); (25, 14, 7); (31, 14, 6); (32, 15, 3); (34, 15, 2); (22, 16, 5); (13, 17, 2); (38, 17, 3); (40, 18, 5); (36, 18, 7); (49, 19, 2); (8, 19, 5); (9, 20, 7); (35, 21, 7); (4, 22, 1); (44, 24, 4); (20, 25, 5); (37, 25, 4); (21, 25, 3); (15, 25, 3); (6, 26, 2); (48, 26, 3); (30, 26, 7); (2, 26, 7); (12, 27, 5); (27, 28, 4); (26, 29, 5); (1, 31, 3); (7, 31, 1); (33, 31, 3); (17, 36, 7); (18, 36, 6); (16, 37, 1); (42, 39, 7); (45, 40, 4);

Queue 1: (5, 6, 2); (50, 11, 6);

Queue 2: (28, 7, 1); (14, 11, 5);

Queue 3: (43, 6, 2);

Queue 4: (11, 5, 1); (47, 11, 4);

Queue 5: (39, 11, 7);

Time 12

Waiting clients: (10, 13, 7); (23, 14, 6); (25, 14, 7); (31, 14, 6); (32, 15, 3); (34, 15, 2); (22, 16, 5); (13, 17, 2); (38, 17, 3); (40, 18, 5); (36, 18, 7); (49, 19, 2); (8, 19, 5); (9, 20, 7); (35, 21, 7); (4, 22, 1); (44, 24, 4); (20, 25, 5); (37, 25, 4); (21, 25, 3); (15, 25, 3); (6, 26, 2); (48, 26, 3); (30, 26, 7); (2, 26, 7); (12, 27, 5); (27, 28, 4); (26, 29, 5); (1, 31, 3); (7, 31, 1); (33, 31, 3); (17, 36, 7); (18, 36, 6); (16, 37, 1); (42, 39, 7); (45, 40, 4);

Queue 1: (5, 6, 1); (50, 11, 6);

Queue 2: (14, 11, 4);

Queue 3: (43, 6, 1);

Queue 4: (47, 11, 3);

Queue 5: (39, 11, 6);

Time 13

Waiting clients: (23, 14, 6); (25, 14, 7); (31, 14, 6); (32, 15, 3); (34, 15, 2); (22, 16, 5); (13, 17, 2); (38, 17, 3); (40, 18, 5); (36, 18, 7); (49, 19, 2); (8, 19, 5); (9, 20, 7); (35, 21, 7); (4, 22, 1); (44, 24, 4); (20, 25, 5); (37, 25, 4); (21, 25, 3); (15, 25, 3); (6, 26, 2); (48, 26, 3); (30, 26, 7); (2, 26, 7); (12, 27, 5); (27, 28, 4); (26, 29, 5); (1, 31, 3); (7, 31, 1); (33, 31, 3); (17, 36, 7); (18, 36, 6); (16, 37, 1); (42, 39, 7); (45, 40, 4);

Queue 1: (50, 11, 5);

Queue 2: (14, 11, 3);

Queue 3: (10, 13, 7);

Queue 4: (47, 11, 2);

Queue 5: (39, 11, 5);

Time 14

Waiting clients: (32, 15, 3); (34, 15, 2); (22, 16, 5); (13, 17, 2); (38, 17, 3); (40, 18, 5); (36, 18, 7); (49, 19, 2); (8, 19, 5); (9, 20, 7); (35, 21, 7); (4, 22, 1); (44, 24, 4); (20, 25, 5); (37, 25, 4); (21, 25, 3); (15, 25, 3); (6, 26, 2); (48, 26, 3); (30, 26, 7); (2, 26, 7); (12, 27, 5); (27, 28, 4); (26, 29, 5); (1, 31, 3); (7, 31, 1); (33, 31, 3); (17, 36, 7); (18, 36, 6); (16, 37, 1); (42, 39, 7); (45, 40, 4);

Queue 1: (50, 11, 4); (31, 14, 6);

Queue 2: (14, 11, 2); (25, 14, 7);

Queue 3: (10, 13, 6);

Queue 4: (47, 11, 1); (23, 14, 6);

Queue 5: (39, 11, 4);

Time 15

Waiting clients: (22, 16, 5); (13, 17, 2); (38, 17, 3); (40, 18, 5); (36, 18, 7); (49, 19, 2); (8, 19, 5); (9, 20, 7); (35, 21, 7); (4, 22, 1); (44, 24, 4); (20, 25, 5); (37, 25, 4); (21, 25, 3); (15, 25, 3); (6, 26, 2); (48, 26, 3); (30, 26, 7); (2, 26, 7); (12, 27, 5); (27, 28, 4); (26, 29, 5); (1, 31, 3); (7, 31, 1); (33, 31, 3); (17, 36, 7); (18, 36, 6); (16, 37, 1); (42, 39, 7); (45, 40, 4);

Queue 1: (50, 11, 3); (31, 14, 6);

Queue 2: (14, 11, 1); (25, 14, 7);

Queue 3: (10, 13, 5); (34, 15, 2);

Queue 4: (23, 14, 5);

Queue 5: (39, 11, 3); (32, 15, 3);

Time 16

Waiting clients: (13, 17, 2); (38, 17, 3); (40, 18, 5); (36, 18, 7); (49, 19, 2); (8, 19, 5); (9, 20, 7); (35, 21, 7); (4, 22, 1); (44, 24, 4); (20, 25, 5); (37, 25, 4); (21, 25, 3); (15, 25, 3); (6, 26, 2); (48, 26, 3); (30, 26, 7); (2, 26, 7); (12, 27, 5); (27, 28, 4); (26, 29, 5); (1, 31, 3); (7, 31, 1); (33, 31, 3); (17, 36, 7); (18, 36, 6); (16, 37, 1); (42, 39, 7); (45, 40, 4);

Queue 1: (50, 11, 2); (31, 14, 6);

Queue 2: (25, 14, 6);

Queue 3: (10, 13, 4); (34, 15, 2);

Queue 4: (23, 14, 4); (22, 16, 5);

Queue 5: (39, 11, 2); (32, 15, 3);

Time 17

Waiting clients: (40, 18, 5); (36, 18, 7); (49, 19, 2); (8, 19, 5); (9, 20, 7); (35, 21, 7); (4, 22, 1); (44, 24, 4); (20, 25, 5); (37, 25, 4); (21, 25, 3); (15, 25, 3); (6, 26, 2); (48, 26, 3); (30, 26, 7); (2, 26, 7); (12, 27, 5); (27, 28, 4); (26, 29, 5); (1, 31, 3); (7, 31, 1); (33, 31, 3); (17, 36, 7); (18, 36, 6); (16, 37, 1); (42, 39, 7); (45, 40, 4);

Queue 1: (50, 11, 1); (31, 14, 6);

Queue 2: (25, 14, 5); (38, 17, 3);

Queue 3: (10, 13, 3); (34, 15, 2);

Queue 4: (23, 14, 3); (22, 16, 5);

Queue 5: (39, 11, 1); (32, 15, 3); (13, 17, 2);

Time 18

Waiting clients: (49, 19, 2); (8, 19, 5); (9, 20, 7); (35, 21, 7); (4, 22, 1); (44, 24, 4); (20, 25, 5); (37, 25, 4); (21, 25, 3); (15, 25, 3); (6, 26, 2); (48, 26, 3); (30, 26, 7); (2, 26, 7); (12, 27, 5); (27, 28, 4); (26, 29, 5); (1, 31, 3); (7, 31, 1); (33, 31, 3); (17, 36, 7); (18, 36, 6); (16, 37, 1); (42, 39, 7); (45, 40, 4);

Queue 1: (31, 14, 5);

Queue 2: (25, 14, 4); (38, 17, 3);

Queue 3: (10, 13, 2); (34, 15, 2); (40, 18, 5);

Queue 4: (23, 14, 2); (22, 16, 5);

Queue 5: (32, 15, 2); (13, 17, 2); (36, 18, 7);

Time 19

Waiting clients: (9, 20, 7); (35, 21, 7); (4, 22, 1); (44, 24, 4); (20, 25, 5); (37, 25, 4); (21, 25, 3); (15, 25, 3); (6, 26, 2); (48, 26, 3); (30, 26, 7); (2, 26, 7); (12, 27, 5); (27, 28, 4); (26, 29, 5); (1, 31, 3); (7, 31, 1); (33, 31, 3); (17, 36, 7); (18, 36, 6); (16, 37, 1); (42, 39, 7); (45, 40, 4);

Queue 1: (31, 14, 4); (49, 19, 2); (8, 19, 5);

Queue 2: (25, 14, 3); (38, 17, 3);

Queue 3: (10, 13, 1); (34, 15, 2); (40, 18, 5);

Queue 4: (23, 14, 1); (22, 16, 5);

Queue 5: (32, 15, 1); (13, 17, 2); (36, 18, 7);

Time 20

Waiting clients: (35, 21, 7); (4, 22, 1); (44, 24, 4); (20, 25, 5); (37, 25, 4); (21, 25, 3); (15, 25, 3); (6, 26, 2); (48, 26, 3); (30, 26, 7); (2, 26, 7); (12, 27, 5); (27, 28, 4); (26, 29, 5); (1, 31, 3); (7, 31, 1); (33, 31, 3); (17, 36, 7); (18, 36, 6); (16, 37, 1); (42, 39, 7); (45, 40, 4);

Queue 1: (31, 14, 3); (49, 19, 2); (8, 19, 5);

Queue 2: (25, 14, 2); (38, 17, 3);

Queue 3: (34, 15, 1); (40, 18, 5);

Queue 4: (22, 16, 4); (9, 20, 7);

Queue 5: (13, 17, 1); (36, 18, 7);

Time 21

Waiting clients: (4, 22, 1); (44, 24, 4); (20, 25, 5); (37, 25, 4); (21, 25, 3); (15, 25, 3); (6, 26, 2); (48, 26, 3); (30, 26, 7); (2, 26, 7); (12, 27, 5); (27, 28, 4); (26, 29, 5); (1, 31, 3); (7, 31, 1); (33, 31, 3); (17, 36, 7); (18, 36, 6); (16, 37, 1); (42, 39, 7); (45, 40, 4);

Queue 1: (31, 14, 2); (49, 19, 2); (8, 19, 5);

Queue 2: (25, 14, 1); (38, 17, 3); (35, 21, 7);

Queue 3: (40, 18, 4);

Queue 4: (22, 16, 3); (9, 20, 7);

Queue 5: (36, 18, 6);

Time 22

Waiting clients: (44, 24, 4); (20, 25, 5); (37, 25, 4); (21, 25, 3); (15, 25, 3); (6, 26, 2); (48, 26, 3); (30, 26, 7); (2, 26, 7); (12, 27, 5); (27, 28, 4); (26, 29, 5); (1, 31, 3); (7, 31, 1); (33, 31, 3); (17, 36, 7); (18, 36, 6); (16, 37, 1); (42, 39, 7); (45, 40, 4);

Queue 1: (31, 14, 1); (49, 19, 2); (8, 19, 5);

Queue 2: (38, 17, 2); (35, 21, 7);

Queue 3: (40, 18, 3); (4, 22, 1);

Queue 4: (22, 16, 2); (9, 20, 7);

Queue 5: (36, 18, 5);

Time 23

Waiting clients: (44, 24, 4); (20, 25, 5); (37, 25, 4); (21, 25, 3); (15, 25, 3); (6, 26, 2); (48, 26, 3); (30, 26, 7); (2, 26, 7); (12, 27, 5); (27, 28, 4); (26, 29, 5); (1, 31, 3); (7, 31, 1); (33, 31, 3); (17, 36, 7); (18, 36, 6); (16, 37, 1); (42, 39, 7); (45, 40, 4);

Queue 1: (49, 19, 1); (8, 19, 5);

Queue 2: (38, 17, 1); (35, 21, 7);

Queue 3: (40, 18, 2); (4, 22, 1);

Queue 4: (22, 16, 1); (9, 20, 7);

Queue 5: (36, 18, 4);

Time 24

Waiting clients: (20, 25, 5); (37, 25, 4); (21, 25, 3); (15, 25, 3); (6, 26, 2); (48, 26, 3); (30, 26, 7); (2, 26, 7); (12, 27, 5); (27, 28, 4); (26, 29, 5); (1, 31, 3); (7, 31, 1); (33, 31, 3); (17, 36, 7); (18, 36, 6); (16, 37, 1); (42, 39, 7); (45, 40, 4);

Queue 1: (8, 19, 4);

Queue 2: (35, 21, 6);

Queue 3: (40, 18, 1); (4, 22, 1); (44, 24, 4);

Queue 4: (9, 20, 6);

Queue 5: (36, 18, 3);

Time 25

Waiting clients: (6, 26, 2); (48, 26, 3); (30, 26, 7); (2, 26, 7); (12, 27, 5); (27, 28, 4); (26, 29, 5); (1, 31, 3); (7, 31, 1); (33, 31, 3); (17, 36, 7); (18, 36, 6); (16, 37, 1); (42, 39, 7); (45, 40, 4);

Queue 1: (8, 19, 3); (37, 25, 4);

Queue 2: (35, 21, 5); (15, 25, 3);

Queue 3: (44, 24, 3); (21, 25, 3);

Queue 4: (9, 20, 5);

Queue 5: (36, 18, 2); (20, 25, 5);

Time 26

Waiting clients: (12, 27, 5); (27, 28, 4); (26, 29, 5); (1, 31, 3); (7, 31, 1); (33, 31, 3); (17, 36, 7); (18, 36, 6); (16, 37, 1); (42, 39, 7); (45, 40, 4);

Queue 1: (8, 19, 2); (37, 25, 4); (30, 26, 7);

Queue 2: (35, 21, 4); (15, 25, 3);

Queue 3: (44, 24, 2); (21, 25, 3); (48, 26, 3);

Queue 4: (9, 20, 4); (6, 26, 2); (2, 26, 7);

Queue 5: (36, 18, 1); (20, 25, 5);

Time 27

Waiting clients: (27, 28, 4); (26, 29, 5); (1, 31, 3); (7, 31, 1); (33, 31, 3); (17, 36, 7); (18, 36, 6); (16, 37, 1); (42, 39, 7); (45, 40, 4);

Queue 1: (8, 19, 1); (37, 25, 4); (30, 26, 7);

Queue 2: (35, 21, 3); (15, 25, 3);

Queue 3: (44, 24, 1); (21, 25, 3); (48, 26, 3);

Queue 4: (9, 20, 3); (6, 26, 2); (2, 26, 7);

Queue 5: (20, 25, 4); (12, 27, 5);

Time 28

Waiting clients: (26, 29, 5); (1, 31, 3); (7, 31, 1); (33, 31, 3); (17, 36, 7); (18, 36, 6); (16, 37, 1); (42, 39, 7); (45, 40, 4);

Queue 1: (37, 25, 3); (30, 26, 7);

Queue 2: (35, 21, 2); (15, 25, 3); (27, 28, 4);

Queue 3: (21, 25, 2); (48, 26, 3);

Queue 4: (9, 20, 2); (6, 26, 2); (2, 26, 7);

Queue 5: (20, 25, 3); (12, 27, 5);

Time 29

Waiting clients: (1, 31, 3); (7, 31, 1); (33, 31, 3); (17, 36, 7); (18, 36, 6); (16, 37, 1); (42, 39, 7); (45, 40, 4);

Queue 1: (37, 25, 2); (30, 26, 7);

Queue 2: (35, 21, 1); (15, 25, 3); (27, 28, 4);

Queue 3: (21, 25, 1); (48, 26, 3); (26, 29, 5);

Queue 4: (9, 20, 1); (6, 26, 2); (2, 26, 7);

Queue 5: (20, 25, 2); (12, 27, 5);

Time 30

Waiting clients: (1, 31, 3); (7, 31, 1); (33, 31, 3); (17, 36, 7); (18, 36, 6); (16, 37, 1); (42, 39, 7); (45, 40, 4);

Queue 1: (37, 25, 1); (30, 26, 7);

Queue 2: (15, 25, 2); (27, 28, 4);

Queue 3: (48, 26, 2); (26, 29, 5);

Queue 4: (6, 26, 1); (2, 26, 7);

Queue 5: (20, 25, 1); (12, 27, 5);

Time 31

Waiting clients: (17, 36, 7); (18, 36, 6); (16, 37, 1); (42, 39, 7); (45, 40, 4);

Queue 1: (30, 26, 6); (33, 31, 3);

Queue 2: (15, 25, 1); (27, 28, 4); (7, 31, 1);

Queue 3: (48, 26, 1); (26, 29, 5);

Queue 4: (2, 26, 6);

Queue 5: (12, 27, 4); (1, 31, 3);

Time 32

Waiting clients: (17, 36, 7); (18, 36, 6); (16, 37, 1); (42, 39, 7); (45, 40, 4);

Queue 1: (30, 26, 5); (33, 31, 3);

Queue 2: (27, 28, 3); (7, 31, 1);

Queue 3: (26, 29, 4);

Queue 4: (2, 26, 5);

Queue 5: (12, 27, 3); (1, 31, 3);

Time 33

Waiting clients: (17, 36, 7); (18, 36, 6); (16, 37, 1); (42, 39, 7); (45, 40, 4);

Queue 1: (30, 26, 4); (33, 31, 3);

Queue 2: (27, 28, 2); (7, 31, 1);

Queue 3: (26, 29, 3);

Queue 4: (2, 26, 4);

Queue 5: (12, 27, 2); (1, 31, 3);

Time 34

Waiting clients: (17, 36, 7); (18, 36, 6); (16, 37, 1); (42, 39, 7); (45, 40, 4);

Queue 1: (30, 26, 3); (33, 31, 3);

Queue 2: (27, 28, 1); (7, 31, 1);

Queue 3: (26, 29, 2);

Queue 4: (2, 26, 3);

Queue 5: (12, 27, 1); (1, 31, 3);

Time 35

Waiting clients: (17, 36, 7); (18, 36, 6); (16, 37, 1); (42, 39, 7); (45, 40, 4);

Queue 1: (30, 26, 2); (33, 31, 3);

Queue 2: closed

Queue 3: (26, 29, 1);

Queue 4: (2, 26, 2);

Queue 5: (1, 31, 2);

Time 36

Waiting clients: (16, 37, 1); (42, 39, 7); (45, 40, 4);

Queue 1: (30, 26, 1); (33, 31, 3);

Queue 2: (17, 36, 7);

Queue 3: (18, 36, 6);

Queue 4: (2, 26, 1);

Queue 5: (1, 31, 1);

Time 37

Waiting clients: (42, 39, 7); (45, 40, 4);

Queue 1: (33, 31, 2);

Queue 2: (17, 36, 6);

Queue 3: (18, 36, 5);

Queue 4: (16, 37, 1);

Queue 5: closed

Time 38

Waiting clients: (42, 39, 7); (45, 40, 4);

Queue 1: (33, 31, 1);

Queue 2: (17, 36, 5);

Queue 3: (18, 36, 4);

Queue 4: closed

Queue 5: closed

Time 39

Waiting clients: (45, 40, 4);

Queue 1: (42, 39, 7);

Queue 2: (17, 36, 4);

Queue 3: (18, 36, 3);

Queue 4: closed

Queue 5: closed

Time 40

Waiting clients:

Queue 1: (42, 39, 6);

Queue 2: (17, 36, 3);

Queue 3: (18, 36, 2);

Queue 4: (45, 40, 4);

Queue 5: closed

Time 41

Waiting clients:

Queue 1: (42, 39, 5);

Queue 2: (17, 36, 2);

Queue 3: (18, 36, 1);

Queue 4: (45, 40, 3);

Queue 5: closed

Time 42

Waiting clients:

Queue 1: (42, 39, 4);

Queue 2: (17, 36, 1);

Queue 3: closed

Queue 4: (45, 40, 2);

Queue 5: closed

Time 43

Waiting clients:

Queue 1: (42, 39, 3);

Queue 2: closed

Queue 3: closed

Queue 4: (45, 40, 1);

Queue 5: closed

Time 44

Waiting clients:

Queue 1: (42, 39, 2);

Queue 2: closed

Queue 3: closed

Queue 4: closed

Queue 5: closed

Time 45

Waiting clients:

Queue 1: (42, 39, 1);

Queue 2: closed

Queue 3: closed

Queue 4: closed

Queue 5: closed

Time 46

Waiting clients:

Queue 1: closed

Queue 2: closed

Queue 3: closed

Queue 4: closed

Queue 5: closed

Simulation stopped at time 46

Average waiting time: 2.48

Average service time: 7.02

Peak hour: 13 customers at time 19

test3.txt – because the size of this file is pretty big, I will not include it in this documentation. The file can be found in the GIT folder with the assignment2 project, and it can be read from there.

6. Conclusions and Further Development

In conclusion, working with threads was quite a difficult task. I did quite a lot of research until I understood the concept and I was able to implement the simulation, but after I got the hang of working with threads and synchronizing them, everything else went quite smoothly. Until I reached the part where I had to implement the Graphical User Interface. The implementation of the GUI that lets the user input the parameters of the simulation was easy, but the part in which we have to display the real-time status of the simulation in real time is pretty challenging.

A thing I have learned from this first task is the importance of dividing the objectives at the very beginning of the problem. A lack of rigor in the planning phase can set back the implementation, so the best approach is to "divide and conquer" the big problem into smaller tasks and implementing, testing and debugging them before going forward.

As for further development of the Queues Simulator, the graphical user interface could benefit from some improvement. For example, the queues could be displayed in an "animated" manner at each time of the simulation, with the users staying inside of them and moving while the time passes. The users could be displayed by their IDs. The queues could contain at their top the waiting time, like in a real shop, so that the clients will know how much they will have to wait until they reach the front of the line. Also, another development would be to generate the clients at random times of the simulation, not just at the start, just like in real life.

7. Bibliography

- Class Materials provided by the professor and the teaching assistant.
- draw.io – for the UML diagram
- http://tynerblain.com/blog/2007/04/09/sample-use-case-example/
- https://docs.oracle.com/javase/tutorial/essential/concurrency/index.html
- https://www.tutorialspoint.com/java/util/timer_schedule_period.htm
- https://www.javacodegeeks.com/2013/01/java-thread-pool-example-using-executors-and-threadpoolexecutor.html