Queue Simulator

Nadu Laura Andreea
Group 30421

Table of Contents

Assignment Objective

Main Objective

Design and implement a simulation application aiming to analyse queuing based systems for determining and minimizing clients' waiting time.

Sub-Objectives

O1. Analyse the problem and identify the requirements

O2. Design the queue simulator

O3. Implement the simulation process

O4. Implement the User Interface

O5. Test the queue simulator

O6. Save the simulation evolution into a log file

Problem analysis, modelling, scenarios, use cases

Analysing the problem

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 queues to the system, but this approach increases the service supplier's costs.

The application should simulate 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 go to the queue)
- $tservice$ (time interval or duration needed to serve the client)

The application tracks every client's total time in the queues and a minimum required output containing:

- The current moment of the simulation
- The average waiting time of a client (computed from the moment he/she enters the queue and until he/she leaves the queue)
- The average service time of a client
- The peak hour, meaning the hour (i.e, the moment) with the most clients in any queue

From this brief description of the problem, we may associate it with the popular Producer-Consumer Problem (also known as the Bounded-Buffer Problem). In the first version of the problem, there was a producer, a consumer and a buffer. The producer would continuously create products and place them into the buffer. During this time, the consumer will take stuff out of the buffer and use it at its own will.

One of the latest versions of the problem involves having more than one producer, consumer and buffer. This is what the Queue Simulator could be turned into.

Modelling the solution

To solve the Producer-Consumer problem we first have to identify the 3 main elements:

1. The producer would be a simulation manager that would take care of simulating a real queue system. This manager holds information about the whole simulation. It will take care of generating random products (i.e, tasks or clients) and by using a scheduler, it would add the task to the most appropriate queue.
2. The buffer in this example would be the queue. There could be as many queues as the user wishes.
3. The consumer will be the queue itself. It will take out the first task when it finishes processing and then move onto the next one.

To have a real simulation, these components must work independently on one another. A queue (also called a server) should extract the first task after it finishes processing until the simulation stops. The manager should add tasks to queues when the right time comes, no matter what tasks are already in the queue.

Having two entities that should work in parallel could be achieved by using different threads for each independent entity. A thread of execution is a sequence of programmed instructions that can be managed independently by a scheduler which decides the way the resources should be used.

For the queue simulator to be implemented, the scheduler would be a special class whose only job will be to decide, based on an imposed strategy, where to place the current task.

The simulation depends on various data which should be introduced by the user to assure that the simulation yields the desired results. Thus, the input required is:

- The maximum time the simulation should run, a positive integer which measures the maximum number of seconds of the run
- The queues number and tasks number, two different positive integers that specify how many servers and tasks have to be created
- The minimum and maximum values for the arrival time and the service time of a client. These are two intervals of positive integers that specify the limits of the time when a client should enter the queue and just how much time should it take to process his/her needs.

Of course, because this is a simulation, the application should guarantee that the events are random. That is why the before-explained input is the only data necessary. All the other information is randomly generated. For example, a task could have any value for the arrival time which is between the introduced bounds.

Functional requirements:

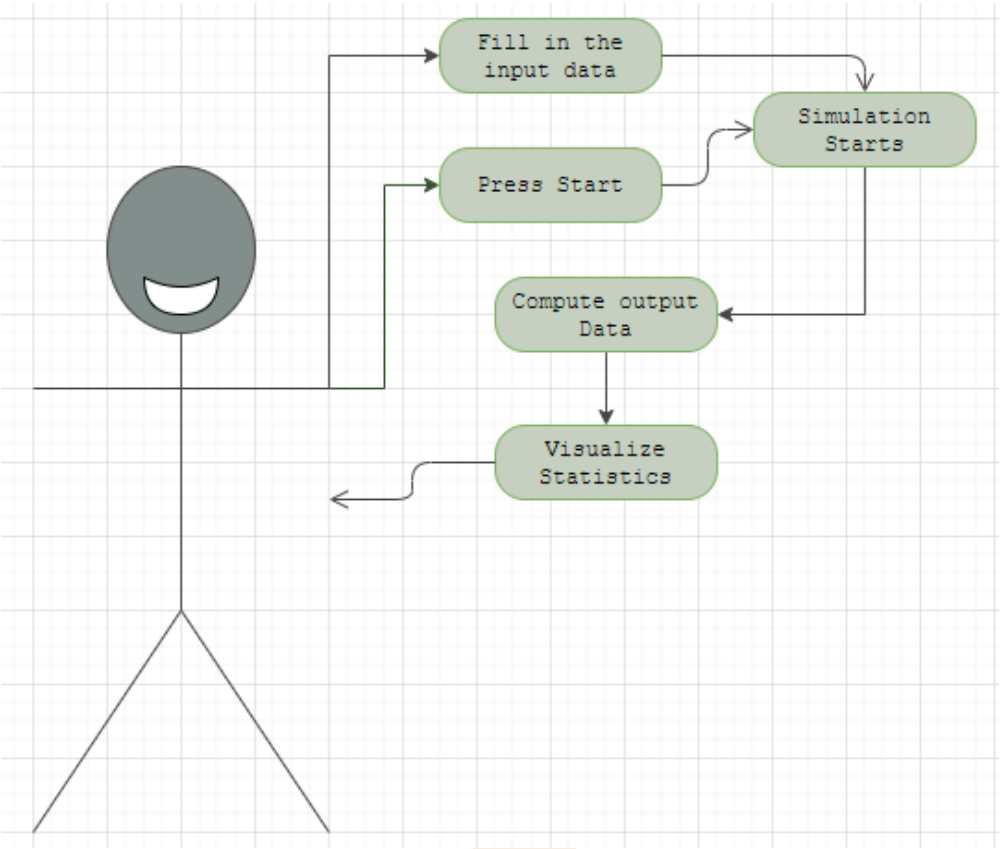
- The queue simulator should allow the user to insert the simulation specifications like the number of queues or clients, the simulation time and the minimum or maximum times for service and arrival.
- The queue simulator should allow the user to start the simulation whenever he/she/they would like to start.
- The queue simulator should display a real-time evolution of the simulation.
- The queue simulator should generate a log file containing a detailed evolution of the queue.
- The queue simulator should display the minimum information about the current moment of the simulation on the user interface.

Non-functional requirements:

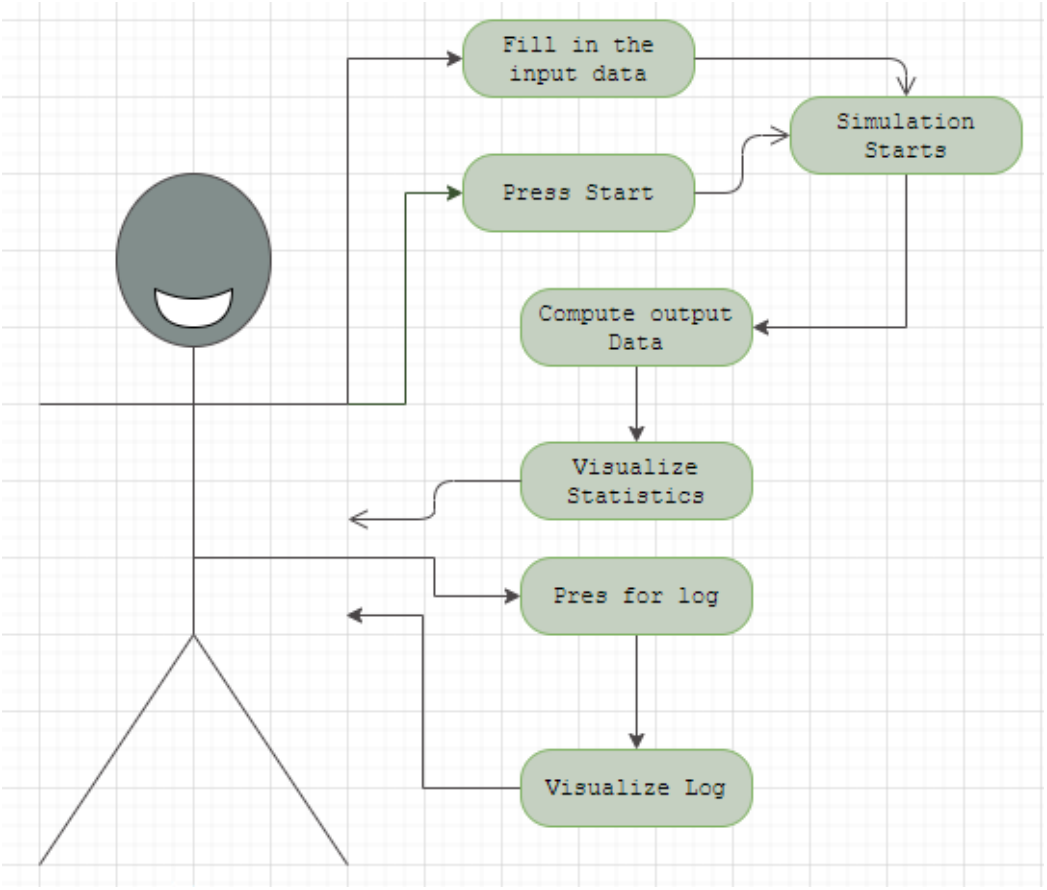
- The queue simulator should be intuitive and easy to use by any user, regardless of their knowledge.
- The queue simulator should verify the correctness of the input data inserted by the user.
- The queue simulator should take care of resetting the user interface according to the evolution of the simulation.

Use Case Diagram

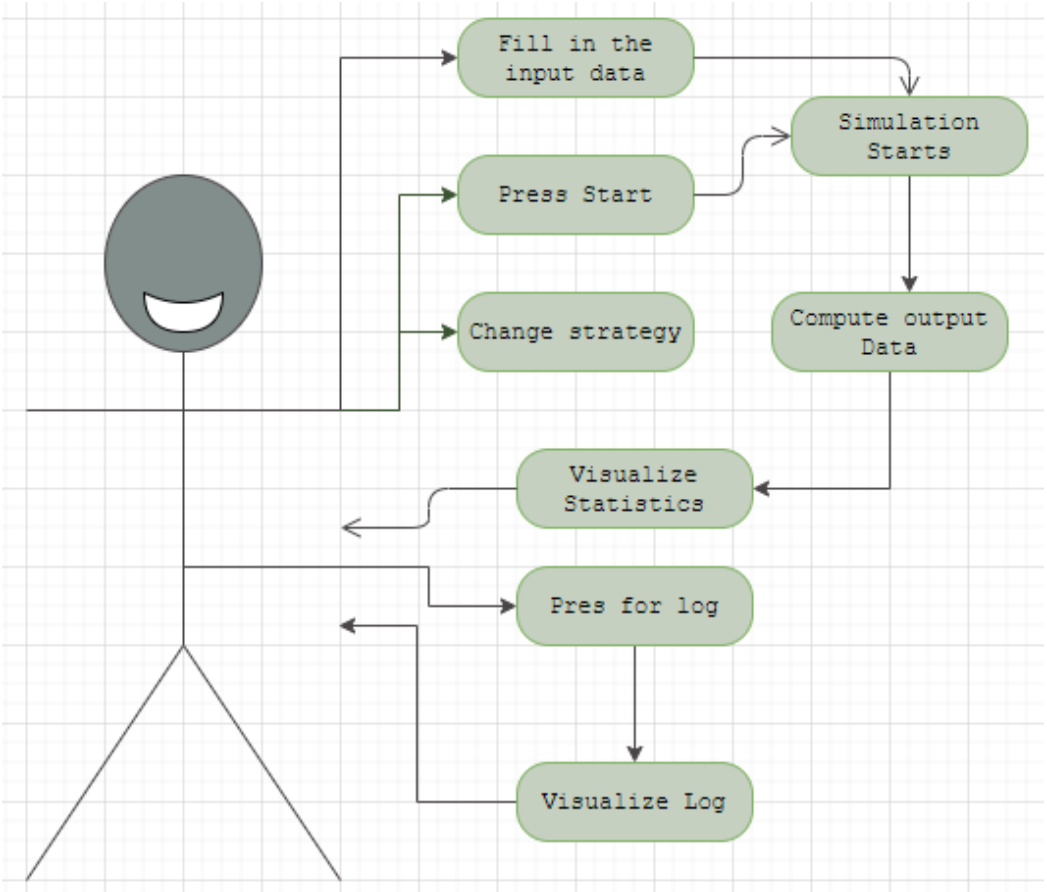
Case 1: The user introduces correct data in the application, presses “Start” and then observes the results and the evolution of the queues.



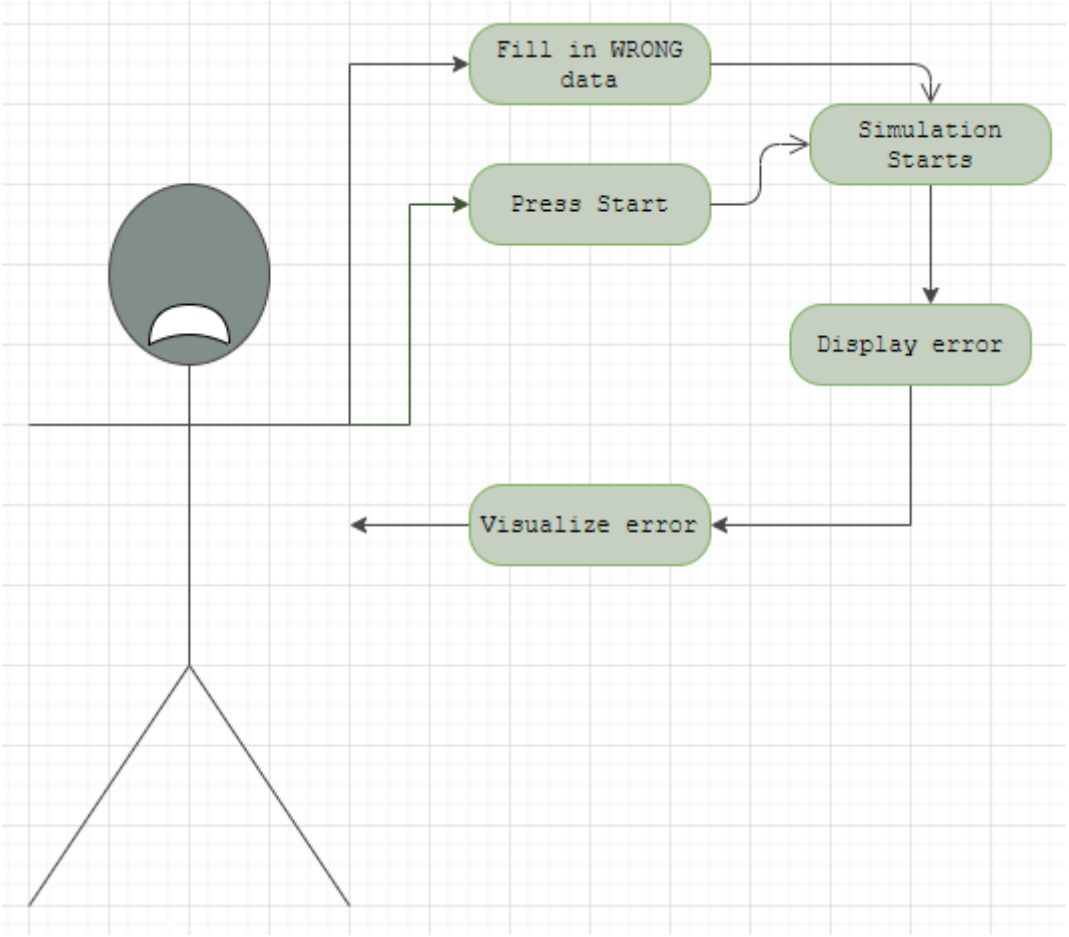
Case 2: The user introduces correct data in the application, presses “Start” and then observes the results and the evolution of the queues. When the simulation ends, he/she/they press the button to view the log file.



Case 3: The user introduces correct data in the application, changes the policy, presses “Start” and then observes the results and the evolution of the queues.

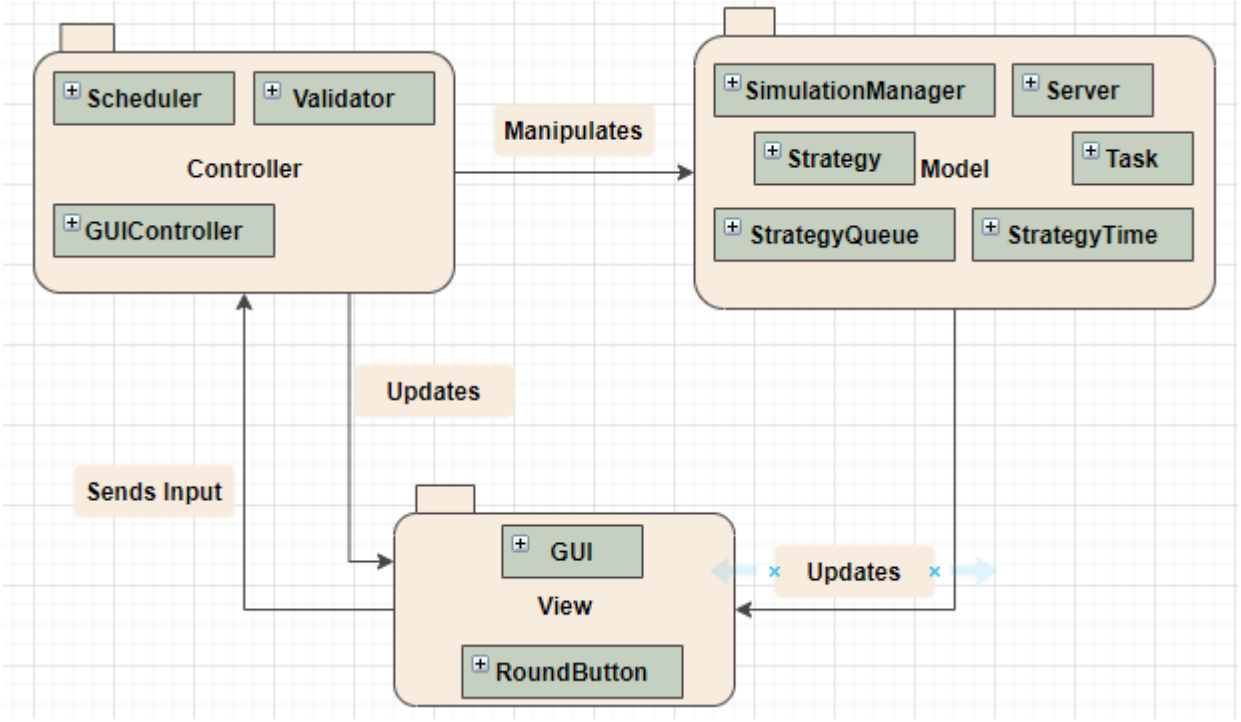


Case 4: The user introduces wrong data in the application, changes the policy, presses “Start” and then observes an error message.

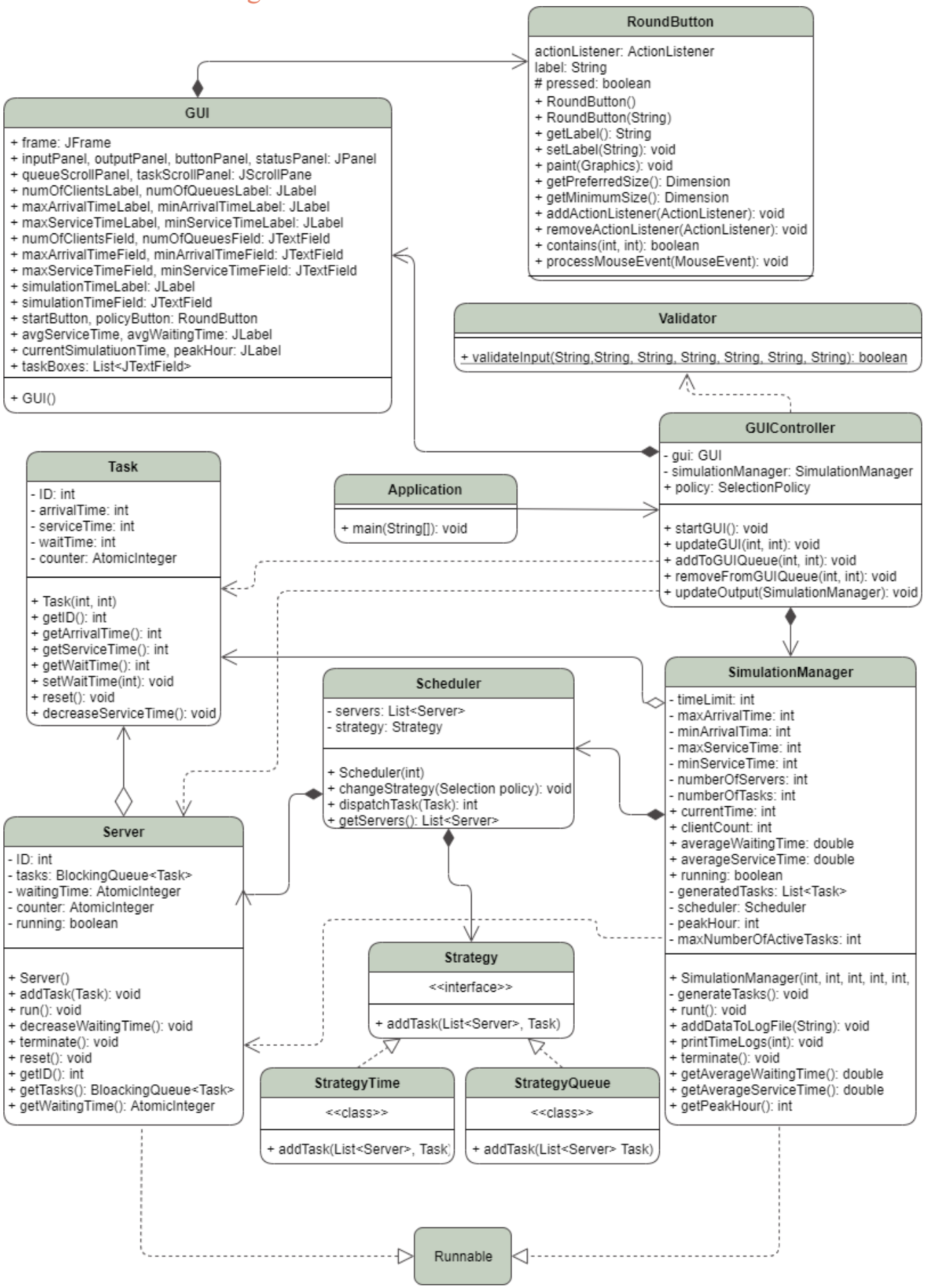


Design Decisions

UML Package Diagram



UML Class Diagram



Data Structures

The performance of an application can be drastically changed by choosing the appropriate data structures. To fit the problem specifications, 3 types of data structures were used:

1. ArrayList, a member of the Java Collection Framework, was used mostly to store the list of servers or the randomly generated tasks. It is a dynamic data structure whose size changes based on the number of elements stored. By using these data structures, access was granted to methods that allow access to any element in the list, which can be just returned or removed, or to methods that return the size of the array so we won't have to store it in a special variable. ArrayList is not a synchronized data structure, meaning that it cannot be used in threads as it is. It could be synchronized by using the "synchronized" keyword, but, as it was not used in by multiple threads, it was not necessary to synchronize it.
2. BlockingQueue, a member of the Java Concurrent Utility, was used because it is a queue data structure in which elements are added based on the FIFO (First In, First Out) principle: the first element inserted is the first element taken out of the list. This is the best way to describe the queues for the simulation, as they work in the same way where an element is a task. The reasoning behind choosing this structure and not a usual Queue structure is because the BlockingQueue is a synchronized data structure that assures thread safety.

Class Design

Non-User Interface Classes:

As presented on the class diagram, the classes describe a more general situation, where the clients are mere tasks to be completed and the queues are just servers that perform the tasks using the "First In, First Out" principle.

For a client (or Task) we are interested in when does it enter the queue (the arrival time), how long does it take for the client to reach the front of the queue (the waiting time) and how much does it take to process the said client (the service time), while, for a queue (or Server), we are interested in knowing which tasks are inside the queue and what is the waiting time of the queue, but also we must be able to add and remove tasks.

The SimulationManager would take care of the whole simulation. It will compute the output, write into the log file, generate random tasks and, most importantly, it will call the scheduler to dispatch the tasks. This means that both the Server and the SimulationManager would work on the same queues simultaneously. Thus, these two classes must be on two separate threads running in parallel.

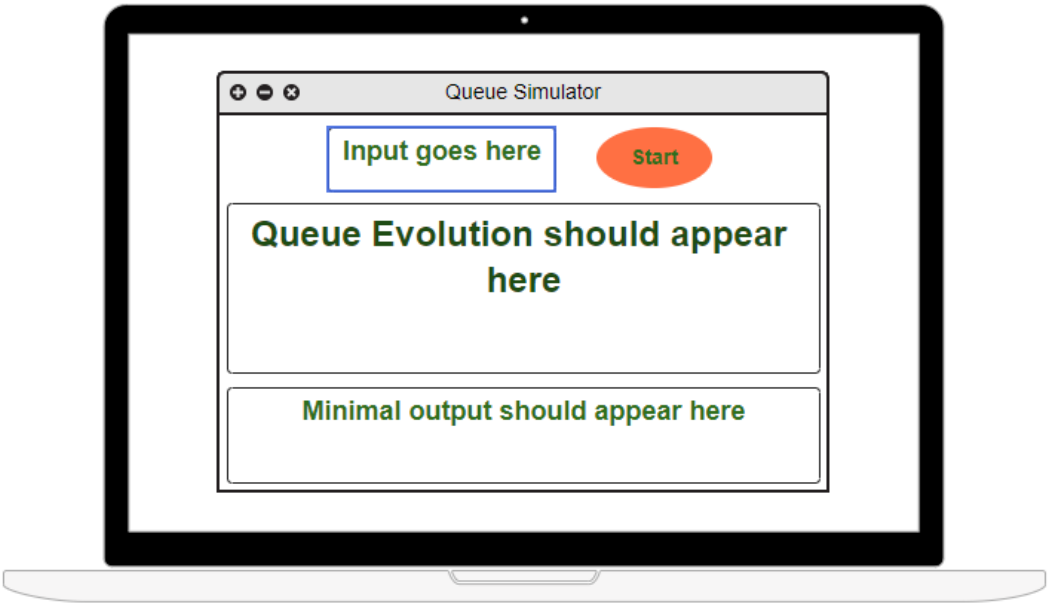
The SimulationManager only decides which task should enter the queues and when. The job of finding the most suited queue comes to the Scheduler class. Based on one of the two strategies, the scheduler chooses the appropriate algorithm to be applied and adds the task to the resulting server.

The algorithms are found inside the Strategy classes (StrategyQueue and StrategyTime). StrategyQueue implements an algorithm that chooses the queue with the least amount of tasks, while StrategyTime chooses the queue with the smallest waiting time.

User Interface Classes:

The GUI class is the class that defines the basic look of the application. It contains buttons, labels, panels and text fields. They have a starting value, but to update and initialize the GUIController class was created. This class updates the GUI based on the status of the SimulationManager and verifies the input introduced by the user by calling the verification method of the Validator class.

Besides these classes, a RoundButton class was introduced to define round buttons in Java Swing, as they do not exist by default.



Algorithms

The application is mostly composed of straight forward methods. The only algorithms we could speak of are the queue selection algorithms imposed by the previously presented strategies.

Get the queue with the lowest waiting time:

Considering that the current waiting time of a queue is updated by the manager for each time iteration of the simulation, to get the queue with the lowest waiting time, we just need to traverse all the available servers and save the one with the smallest waiting time. The task will be added to the found server.

```
addTask( task )
        minimum = Infinity
        for each server in the Scheduler
                if serverWaitingTime < minimum then
                        minimum = serverWaitingTime
                        bestServer = server
        addTask( bestServer, task )
```

Get the smallest queue:

Considering that we know the size of each server, to get the smallest queue we have to iterate through each server and save the one with the smallest size. The task will be added to the found server.

```
addTask( task )
        minimum = Infinity
        for each server in the Scheduler
                if serverSize < minimum then
                        minimum = serverSize
                        bestServer = server
        addTask( bestServer, task )
```

Implementation

Server

This is the class that stores and empties the queues. Because it has to perform actions upon the servers, it must run on a different thread, so this class implements the Runnable interface which allows creating a new thread. The constructor of the class initializes the queue, the ID and the waiting time of one specific server.

The class has many methods like getters and setters, methods for adding a task, decreasing the waiting time or for terminating the server, but the most important method is the run method. It is the first method called when a server thread is started. While running, the thread has to extract the first element from the queue and wait until it is processed.

Because the size of the queues is unknown, LinkedBlockingQueue was used.

```
@Override
public void run() {
    while (running) {
        try {
            Task task = tasks.peek();
            if(task != null) {
                SimulationManager.averageServiceTime += task.getServiceTime();
                SimulationManager.clientCount++;
                Thread.sleep(task.getServiceTime() * 1000);
                if (SimulationManager.running) {
                    GUIController.removeFromGUIQueue(this.ID, task.getID());
                    tasks.take();
                }
            }
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

SimulationManager

This is the main class of the back-end system. Because it has to perform actions upon the servers, it must run on a different thread, so this class implements the Runnable interface which allows creating a new thread. The constructor of the class initializes the attributes, creates a new Scheduler based on the given policy, generates the tasks and creates the Log file.

The tasks are generated randomly by the bounds imposed by the user. For each task, we generate a random number for each given interval and then create a task when a valid pair of values is found. Then save the task into an ArrayList which will be sorted increasingly based on the arrival time of the tasks inside.

```
private void generateTasks() {
    generatedTasks = new ArrayList<>();
    int arrivalTime, serviceTime;
    for (int i = 0; i < numberOfTasks; i++) {
        do {
            Random random = new Random();
            arrivalTime = random.nextInt(maxArrivalTime);
        } while(arrivalTime < minArrivalTime);
        do {
            Random random = new Random();
            serviceTime = random.nextInt(maxServiceTime);
        } while(serviceTime < minServiceTime);
        Task task = new Task(arrivalTime, serviceTime);
        generatedTasks.add(task);
    }
    generatedTasks.sort((t1, t2) -> t1.getArrivalTime() - t2.getArrivalTime());
}
```

There is one more method to be presented: the run method. This is the most important method of the class, being called right when the thread is started. The method's run corresponds to the thread run. So as long as the thread is running, the thread has to complete a sequence of tasks:

1. First, for the current time, add all the tasks that have an arrival time equal to the current moment to the queues. For each task to be added, the scheduler will find the perfect queue.
2. For each server in the scheduler, we have to decrease the waiting time and to see how many tasks are in the all of the queues, based on the number found, we may or may not update the peak hour
3. Then the thread has to "sleep" or pause execution for 1 second.
4. When it resumes, it terminates if either all the clients have left the queues or if the current time reached the simulation time limit. If neither is true, then the current time is increased and the process is repeated.

Besides the queues evolution wish is updated both in the gui and in the log file, the simulator has to compute, in real time, 3 values:

1. The peak hour, which is the moment of time where there were the most clients into a queue
2. The average waiting time, this time is computed for the clients that entered the queues. It is defined as the period from the moment a client enters the queue and the moment he/she leaves.
3. The average service time, which is the average time a client spends in front of the queue. This time is computed based on how many clients have left the queues (have finished processing)

Task

This class is defining the structure of a task (or client). The methods inside refer to mainly computing the waiting time and getting or setting the attributes. The waiting time of a client is the waiting time of the queue from the moment the task was added (how much time it has to wait until it reaches the front) and the service time of the task.

```
public void setWaitTime(int queueWaitingTime) {
    this.waitTime = queueWaitingTime + serviceTime;
}
```

Strategy

Strategy is just an interface that specifies it has to handle the process of adding a task to a server. The handling is done through just one method: addTask. It receives the task to be added and the list of available servers from the scheduler.

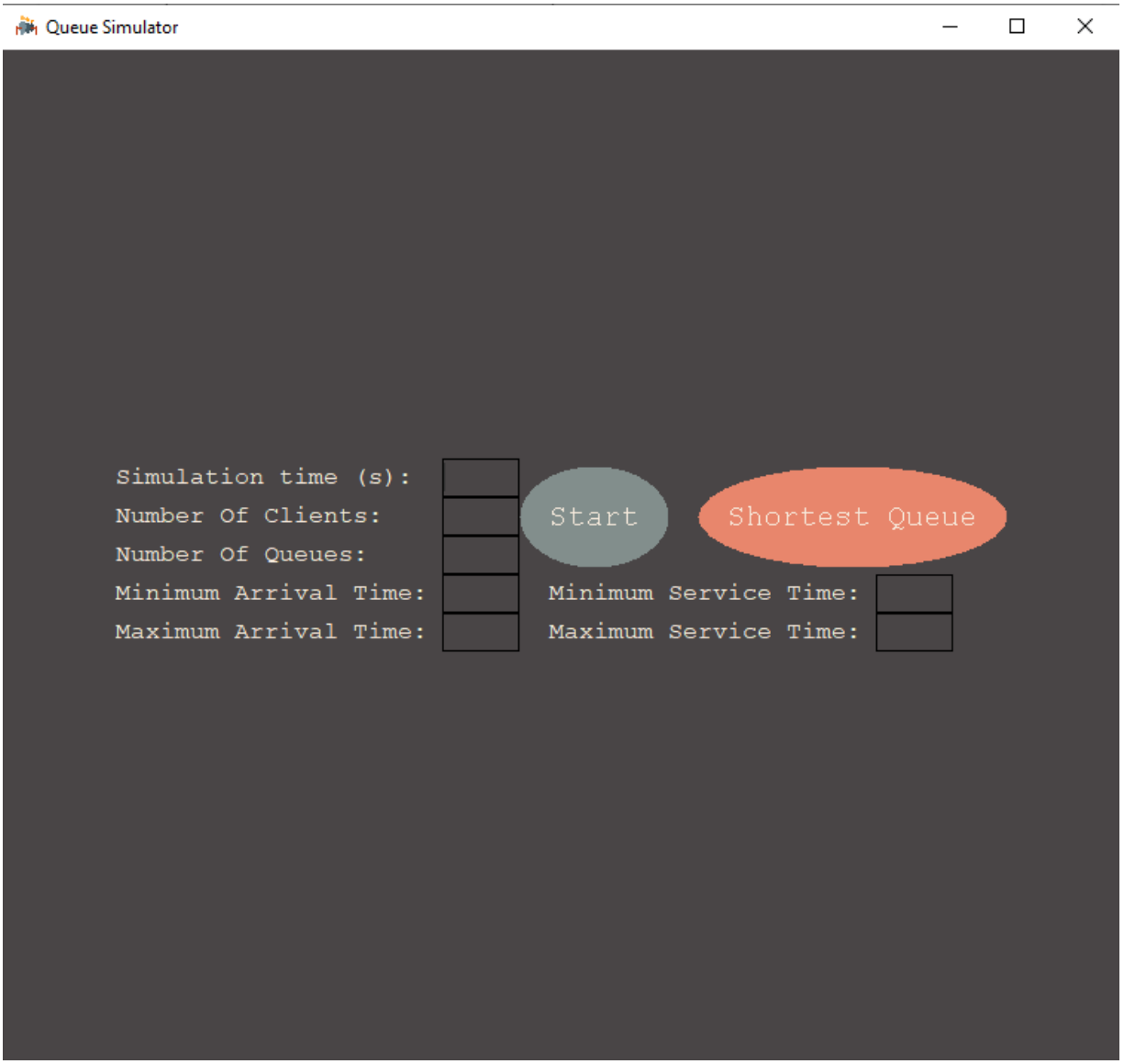
There are two classes that implement this interface: StrategyTime and StrategyQueue.

StrategyTime: As described by the algorithm, this class chooses the server with the lowest waiting time.

StrategyQueue: As described by the algorithm, this class chooses the server with the smallest number of tasks.

GUI

This is the class that defines the components used inside of the user interface. In the constructor of this class, only the elements from the input panel are visible to the user. When created, the interface looks like this:



Just after the user starts the simulation, the other panels should appear.

GUIController

GUIController is the class that links the back-end simulation with the front-end seen by the user.

When the “Shortest Queue” button is pressed, the default policy is modified from testing time to testing the size of the queues.

Just when the “Start” button is pressed the simulation actually starts. The thread for the manager is started, the panels are cleared, and the GUI is then updated.

The updateGUI() method is called for each second that passes in the simulation. When called, the panels are reset and then re-filled with the new structure of the queues. To set a new structure to the queue, the simulator calls for two methods to move a task from waiting to a queue and to remove a task from queues entirely.

To move a task around the GUI, I used a list of text fields for the tasks and a list of panels for the queues. When a task is moved, it’s corresponding text field is added to the specific queue panel and removed from the previous one.

Validator

It has only a static method which makes sure that the input is composed of only numbers and that the minimum values are not greater than the maximum ones.

RoundButton

This class is mostly taken from StackOverflow to help define a feature missing from Java Swing.

Testing

In the assignment description there were 3 use cases given, so I decided to use them for testing.

Test 1

Test 1
$N = 4$
$Q = 2$
$t^{MAX}_{simulation} = 60$ seconds
$[t^{MIN}_{arrival}, t^{MAX}_{arrival}] = [2, 30]$
$[t^{MIN}_{service}, t^{MAX}_{service}] = [2, 4]$

```
Current Simulation Time: 31

Average Waiting Time: 2.5

Average Service Time: 2.5

Peak Hour: 2
```

*** Log File ***

Time 0
Waiting clients: (2, 2, 2); (1, 6, 3); (3, 26, 3); (4, 29, 2);
Queue 1: closed
Queue 2: closed

Time 1
Waiting clients: (2, 2, 2); (1, 6, 3); (3, 26, 3); (4, 29, 2);
Queue 1: closed
Queue 2: closed

Time 2
Waiting clients: (1, 6, 3); (3, 26, 3); (4, 29, 2);
Queue 1: (2, 2, 2);
Queue 2: closed

Time 3
Waiting clients: (1, 6, 3); (3, 26, 3); (4, 29, 2);
Queue 1: (2, 2, 1);
Queue 2: closed

Time 4
Waiting clients: (1, 6, 3); (3, 26, 3); (4, 29, 2);
Queue 1: closed
Queue 2: closed

Time 5
Waiting clients: (1, 6, 3); (3, 26, 3); (4, 29, 2);
Queue 1: closed
Queue 2: closed

Time 6
Waiting clients: (3, 26, 3); (4, 29, 2);
Queue 1: (1, 6, 3);
Queue 2: closed

Time 7
Waiting clients: (3, 26, 3); (4, 29, 2);
Queue 1: (1, 6, 2);
Queue 2: closed

Time 8
Waiting clients: (3, 26, 3); (4, 29, 2);
Queue 1: (1, 6, 1);
Queue 2: closed

Time 9
Waiting clients: (3, 26, 3); (4, 29, 2);
Queue 1: closed

Queue 2: closed

Time 10
Waiting clients: (3, 26, 3); (4, 29, 2);
Queue 1: closed
Queue 2: closed

Time 11
Waiting clients: (3, 26, 3); (4, 29, 2);
Queue 1: closed
Queue 2: closed

Time 12
Waiting clients: (3, 26, 3); (4, 29, 2);
Queue 1: closed
Queue 2: closed

Time 13
Waiting clients: (3, 26, 3); (4, 29, 2);
Queue 1: closed
Queue 2: closed

Time 14
Waiting clients: (3, 26, 3); (4, 29, 2);
Queue 1: closed
Queue 2: closed

Time 15
Waiting clients: (3, 26, 3); (4, 29, 2);
Queue 1: closed
Queue 2: closed

Time 16
Waiting clients: (3, 26, 3); (4, 29, 2);
Queue 1: closed
Queue 2: closed

Time 17
Waiting clients: (3, 26, 3); (4, 29, 2);
Queue 1: closed
Queue 2: closed

Time 18
Waiting clients: (3, 26, 3); (4, 29, 2);
Queue 1: closed
Queue 2: closed

Time 19
Waiting clients: (3, 26, 3); (4, 29, 2);
Queue 1: closed
Queue 2: closed

Time 20
Waiting clients: (3, 26, 3); (4, 29, 2);
Queue 1: closed
Queue 2: closed

Time 21
Waiting clients: (3, 26, 3); (4, 29, 2);
Queue 1: closed
Queue 2: closed

Time 22
Waiting clients: (3, 26, 3); (4, 29, 2);
Queue 1: closed

Queue 2: closed

Time 23
Waiting clients: (3, 26, 3); (4, 29, 2);
Queue 1: closed
Queue 2: closed

Time 24
Waiting clients: (3, 26, 3); (4, 29, 2);
Queue 1: closed
Queue 2: closed

Time 25
Waiting clients: (3, 26, 3); (4, 29, 2);
Queue 1: closed
Queue 2: closed

Time 26
Waiting clients: (4, 29, 2);
Queue 1: (3, 26, 3);
Queue 2: closed

Time 27
Waiting clients: (4, 29, 2);
Queue 1: (3, 26, 2);
Queue 2: closed

Time 28
Waiting clients: (4, 29, 2);
Queue 1: (3, 26, 1);
Queue 2: closed

Time 29
Waiting clients:
Queue 1: (4, 29, 2);
Queue 2: closed

Time 30
Waiting clients:
Queue 1: (4, 29, 1);
Queue 2: closed

Time 31
Waiting clients:
Queue 1: closed
Queue 2: closed

Average Waiting Time: 2.5
Average Service Time: 2.5
Peak Hour: 2

*** End of Log File ***

Test 2

Test 2
N = 50
Q = 5
$t_{simulation}^{MAX} = 60$ seconds
$[t_{arrival}^{MIN}, t_{arrival}^{MAX}] = [2, 40]$
$[t_{service}^{MIN}, t_{service}^{MAX}] = [1, 7]$

```
Current Simulation Time: 60

Average Waiting Time: 30.24

Average Service Time: 3.484848484848485

Peak Hour: 39
```

*** Log File ***

Time 0

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

Queue 1: closed

Queue 2: closed

Time 1

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

Queue 1: closed

Queue 2: closed

Time 2

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

Queue 1: closed

Queue 2: closed

Time 3

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

Queue 1: closed

Queue 2: closed

Time 4

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

Queue 1: (50, 4, 4);

Queue 2: closed

Time 5

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

Queue 1: (50, 4, 3); (49, 5, 3);

Queue 2: (44, 5, 3);

Time 6

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

Queue 1: (50, 4, 2); (49, 5, 3);

Queue 2: (44, 5, 2);

Time 7

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

Queue 1: (50, 4, 1); (49, 5, 3); (31, 7, 3);

Queue 2: (44, 5, 1); (23, 7, 3);

Time 8

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

Queue 1: (49, 5, 3); (31, 7, 3);

Queue 2: (23, 7, 3); (34, 8, 2); (36, 8, 3);

Time 9

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

Queue 1: (49, 5, 2); (31, 7, 3);

Queue 2: (23, 7, 2); (34, 8, 2); (36, 8, 3);

Time 10

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

Queue 1: (49, 5, 1); (31, 7, 3); (5, 10, 2); (40, 10, 5);

Queue 2: (23, 7, 1); (34, 8, 2); (36, 8, 3);

Time 11

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

Queue 1: (31, 7, 3); (5, 10, 2); (40, 10, 5);

Queue 2: (34, 8, 2); (36, 8, 3); (10, 11, 3); (30, 11, 5);

Time 12

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

Queue 1: (31, 7, 2); (5, 10, 2); (40, 10, 5); (12, 12, 1); (20, 12, 3); (47, 12, 5);

Queue 2: (34, 8, 1); (36, 8, 3); (10, 11, 3); (30, 11, 5); (27, 12, 6);

Time 13

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

Queue 1: (31, 7, 1); (5, 10, 2); (40, 10, 5); (12, 12, 1); (20, 12, 3); (47, 12, 5); (4, 13, 5);

Queue 2: (36, 8, 3); (10, 11, 3); (30, 11, 5); (27, 12, 6); (15, 13, 6);

Time 14

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

Queue 1: (5, 10, 2); (40, 10, 5); (12, 12, 1); (20, 12, 3); (47, 12, 5); (4, 13, 5); (21, 14, 6);

Queue 2: (36, 8, 2); (10, 11, 3); (30, 11, 5); (27, 12, 6); (15, 13, 6);

Time 15

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

Queue 1: (5, 10, 1); (40, 10, 5); (12, 12, 1); (20, 12, 3); (47, 12, 5); (4, 13, 5); (21, 14, 6);

Queue 2: (36, 8, 1); (10, 11, 3); (30, 11, 5); (27, 12, 6); (15, 13, 6);

Time 16

Waiting clients: (22, 19, 6); (9, 20, 4); (13, 20, 4); (42, 20, 1); (11, 21, 2); (26, 21, 1); (1, 23, 4); (25, 23, 1); (43, 23, 2); (45, 23, 4); (28, 24, 3); (39, 25, 4); (6, 27, 4); (37, 28, 4); (48, 28, 4); (8, 29, 4); (2, 30, 2); (3, 30, 5); (41, 30, 4); (14, 31, 2); (19, 31, 1); (24, 31, 6); (29, 31, 3); (18, 35, 5); (38, 35, 5); (7, 37, 3); (33, 38, 6); (46, 39, 4);

Queue 1: (40, 10, 5); (12, 12, 1); (20, 12, 3); (47, 12, 5); (4, 13, 5); (21, 14, 6); (32, 16, 6);

Queue 2: (10, 11, 3); (30, 11, 5); (27, 12, 6); (15, 13, 6); (16, 16, 2); (17, 16, 4); (35, 16, 3);

Time 17

Waiting clients: (22, 19, 6); (9, 20, 4); (13, 20, 4); (42, 20, 1); (11, 21, 2); (26, 21, 1); (1, 23, 4); (25, 23, 1); (43, 23, 2); (45, 23, 4); (28, 24, 3); (39, 25, 4); (6, 27, 4); (37, 28, 4); (48, 28, 4); (8, 29, 4); (2, 30, 2); (3, 30, 5); (41, 30, 4); (14, 31, 2); (19, 31, 1); (24, 31, 6); (29, 31, 3); (18, 35, 5); (38, 35, 5); (7, 37, 3); (33, 38, 6); (46, 39, 4);

Queue 1: (40, 10, 4); (12, 12, 1); (20, 12, 3); (47, 12, 5); (4, 13, 5); (21, 14, 6); (32, 16, 6);

Queue 2: (10, 11, 2); (30, 11, 5); (27, 12, 6); (15, 13, 6); (16, 16, 2); (17, 16, 4); (35, 16, 3);

Time 18

Waiting clients: (22, 19, 6); (9, 20, 4); (13, 20, 4); (42, 20, 1); (11, 21, 2); (26, 21, 1); (1, 23, 4); (25, 23, 1); (43, 23, 2); (45, 23, 4); (28, 24, 3); (39, 25, 4); (6, 27, 4); (37, 28, 4); (48, 28, 4); (8, 29, 4); (2, 30, 2); (3, 30, 5); (41, 30, 4); (14, 31, 2); (19, 31, 1); (24, 31, 6); (29, 31, 3); (18, 35, 5); (38, 35, 5); (7, 37, 3); (33, 38, 6); (46, 39, 4);

Queue 1: (40, 10, 3); (12, 12, 1); (20, 12, 3); (47, 12, 5); (4, 13, 5); (21, 14, 6); (32, 16, 6);

Queue 2: (10, 11, 1); (30, 11, 5); (27, 12, 6); (15, 13, 6); (16, 16, 2); (17, 16, 4); (35, 16, 3);

Time 19

Waiting clients: (9, 20, 4); (13, 20, 4); (42, 20, 1); (11, 21, 2); (26, 21, 1); (1, 23, 4); (25, 23, 1); (43, 23, 2); (45, 23, 4); (28, 24, 3); (39, 25, 4); (6, 27, 4); (37, 28, 4); (48, 28, 4); (8, 29, 4); (2, 30, 2); (3, 30, 5); (41, 30, 4); (14, 31, 2); (19, 31, 1); (24, 31, 6); (29, 31, 3); (18, 35, 5); (38, 35, 5); (7, 37, 3); (33, 38, 6); (46, 39, 4);

Queue 1: (40, 10, 2); (12, 12, 1); (20, 12, 3); (47, 12, 5); (4, 13, 5); (21, 14, 6); (32, 16, 6);

Queue 2: (30, 11, 5); (27, 12, 6); (15, 13, 6); (16, 16, 2); (17, 16, 4); (35, 16, 3); (22, 19, 6);

Time 20

Waiting clients: (11, 21, 2); (26, 21, 1); (1, 23, 4); (25, 23, 1); (43, 23, 2); (45, 23, 4); (28, 24, 3); (39, 25, 4); (6, 27, 4); (37, 28, 4); (48, 28, 4); (8, 29, 4); (2, 30, 2); (3, 30, 5); (41, 30, 4); (14, 31, 2); (19, 31, 1); (24, 31, 6); (29, 31, 3); (18, 35, 5); (38, 35, 5); (7, 37, 3); (33, 38, 6); (46, 39, 4);

Queue 1: (40, 10, 1); (12, 12, 1); (20, 12, 3); (47, 12, 5); (4, 13, 5); (21, 14, 6); (32, 16, 6); (9, 20, 4); (13, 20, 4);

Queue 2: (30, 11, 4); (27, 12, 6); (15, 13, 6); (16, 16, 2); (17, 16, 4); (35, 16, 3); (22, 19, 6); (42, 20, 1);

Time 21

Waiting clients: (1, 23, 4); (25, 23, 1); (43, 23, 2); (45, 23, 4); (28, 24, 3); (39, 25, 4); (6, 27, 4); (37, 28, 4); (48, 28, 4); (8, 29, 4); (2, 30, 2); (3, 30, 5); (41, 30, 4); (14, 31, 2); (19, 31, 1); (24, 31, 6); (29, 31, 3); (18, 35, 5); (38, 35, 5); (7, 37, 3); (33, 38, 6); (46, 39, 4);

Queue 1: (12, 12, 1); (20, 12, 3); (47, 12, 5); (4, 13, 5); (21, 14, 6); (32, 16, 6); (9, 20, 4); (13, 20, 4);

Queue 2: (30, 11, 3); (27, 12, 6); (15, 13, 6); (16, 16, 2); (17, 16, 4); (35, 16, 3); (22, 19, 6); (42, 20, 1); (11, 21, 2); (26, 21, 1);

Time 22

Waiting clients: (1, 23, 4); (25, 23, 1); (43, 23, 2); (45, 23, 4); (28, 24, 3); (39, 25, 4); (6, 27, 4); (37, 28, 4); (48, 28, 4); (8, 29, 4); (2, 30, 2); (3, 30, 5); (41, 30, 4); (14, 31, 2); (19, 31, 1); (24, 31, 6); (29, 31, 3); (18, 35, 5); (38, 35, 5); (7, 37, 3); (33, 38, 6); (46, 39, 4);

Queue 1: (20, 12, 3); (47, 12, 5); (4, 13, 5); (21, 14, 6); (32, 16, 6); (9, 20, 4); (13, 20, 4);

Queue 2: (30, 11, 2); (27, 12, 6); (15, 13, 6); (16, 16, 2); (17, 16, 4); (35, 16, 3); (22, 19, 6); (42, 20, 1); (11, 21, 2); (26, 21, 1);

Time 23

Waiting clients: (28, 24, 3); (39, 25, 4); (6, 27, 4); (37, 28, 4); (48, 28, 4); (8, 29, 4); (2, 30, 2); (3, 30, 5); (41, 30, 4); (14, 31, 2); (19, 31, 1); (24, 31, 6); (29, 31, 3); (18, 35, 5); (38, 35, 5); (7, 37, 3); (33, 38, 6); (46, 39, 4);

Queue 1: (20, 12, 2); (47, 12, 5); (4, 13, 5); (21, 14, 6); (32, 16, 6); (9, 20, 4); (13, 20, 4); (1, 23, 4);

Queue 2: (30, 11, 1); (27, 12, 6); (15, 13, 6); (16, 16, 2); (17, 16, 4); (35, 16, 3); (22, 19, 6); (42, 20, 1); (11, 21, 2); (26, 21, 1); (25, 23, 1); (43, 23, 2); (45, 23, 4);

Time 24

Waiting clients: (39, 25, 4); (6, 27, 4); (37, 28, 4); (48, 28, 4); (8, 29, 4); (2, 30, 2); (3, 30, 5); (41, 30, 4); (14, 31, 2); (19, 31, 1); (24, 31, 6); (29, 31, 3); (18, 35, 5); (38, 35, 5); (7, 37, 3); (33, 38, 6); (46, 39, 4);

Queue 1: (20, 12, 1); (47, 12, 5); (4, 13, 5); (21, 14, 6); (32, 16, 6); (9, 20, 4); (13, 20, 4); (1, 23, 4); (28, 24, 3);

Queue 2: (27, 12, 6); (15, 13, 6); (16, 16, 2); (17, 16, 4); (35, 16, 3); (22, 19, 6); (42, 20, 1); (11, 21, 2); (26, 21, 1); (25, 23, 1); (43, 23, 2); (45, 23, 4);

Time 25

Waiting clients: (6, 27, 4); (37, 28, 4); (48, 28, 4); (8, 29, 4); (2, 30, 2); (3, 30, 5); (41, 30, 4); (14, 31, 2); (19, 31, 1); (24, 31, 6); (29, 31, 3); (18, 35, 5); (38, 35, 5); (7, 37, 3); (33, 38, 6); (46, 39, 4);

Queue 1: (47, 12, 5); (4, 13, 5); (21, 14, 6); (32, 16, 6); (9, 20, 4); (13, 20, 4); (1, 23, 4); (28, 24, 3); (39, 25, 4);

Queue 2: (27, 12, 5); (15, 13, 6); (16, 16, 2); (17, 16, 4); (35, 16, 3); (22, 19, 6); (42, 20, 1); (11, 21, 2); (26, 21, 1); (25, 23, 1); (43, 23, 2); (45, 23, 4);

Time 26

Waiting clients: (6, 27, 4); (37, 28, 4); (48, 28, 4); (8, 29, 4); (2, 30, 2); (3, 30, 5); (41, 30, 4); (14, 31, 2); (19, 31, 1); (24, 31, 6); (29, 31, 3); (18, 35, 5); (38, 35, 5); (7, 37, 3); (33, 38, 6); (46, 39, 4);

Queue 1: (47, 12, 4); (4, 13, 5); (21, 14, 6); (32, 16, 6); (9, 20, 4); (13, 20, 4); (1, 23, 4); (28, 24, 3); (39, 25, 4);

Queue 2: (27, 12, 4); (15, 13, 6); (16, 16, 2); (17, 16, 4); (35, 16, 3); (22, 19, 6); (42, 20, 1); (11, 21, 2); (26, 21, 1); (25, 23, 1); (43, 23, 2); (45, 23, 4);

Time 27

Waiting clients: (37, 28, 4); (48, 28, 4); (8, 29, 4); (2, 30, 2); (3, 30, 5); (41, 30, 4); (14, 31, 2); (19, 31, 1); (24, 31, 6); (29, 31, 3); (18, 35, 5); (38, 35, 5); (7, 37, 3); (33, 38, 6); (46, 39, 4);

Queue 1: (47, 12, 3); (4, 13, 5); (21, 14, 6); (32, 16, 6); (9, 20, 4); (13, 20, 4); (1, 23, 4); (28, 24, 3); (39, 25, 4);

Queue 2: (27, 12, 3); (15, 13, 6); (16, 16, 2); (17, 16, 4); (35, 16, 3); (22, 19, 6); (42, 20, 1); (11, 21, 2); (26, 21, 1); (25, 23, 1); (43, 23, 2); (45, 23, 4); (6, 27, 4);

Time 28

Waiting clients: (8, 29, 4); (2, 30, 2); (3, 30, 5); (41, 30, 4); (14, 31, 2); (19, 31, 1); (24, 31, 6); (29, 31, 3); (18, 35, 5); (38, 35, 5); (7, 37, 3); (33, 38, 6); (46, 39, 4);

Queue 1: (47, 12, 2); (4, 13, 5); (21, 14, 6); (32, 16, 6); (9, 20, 4); (13, 20, 4); (1, 23, 4); (28, 24, 3); (39, 25, 4); (37, 28, 4);

Queue 2: (27, 12, 2); (15, 13, 6); (16, 16, 2); (17, 16, 4); (35, 16, 3); (22, 19, 6); (42, 20, 1); (11, 21, 2); (26, 21, 1); (25, 23, 1); (43, 23, 2); (45, 23, 4); (6, 27, 4); (48, 28, 4);

Time 29

Waiting clients: (2, 30, 2); (3, 30, 5); (41, 30, 4); (14, 31, 2); (19, 31, 1); (24, 31, 6); (29, 31, 3); (18, 35, 5); (38, 35, 5); (7, 37, 3); (33, 38, 6); (46, 39, 4);

Queue 1: (47, 12, 1); (4, 13, 5); (21, 14, 6); (32, 16, 6); (9, 20, 4); (13, 20, 4); (1, 23, 4); (28, 24, 3); (39, 25, 4); (37, 28, 4); (8, 29, 4);

Queue 2: (27, 12, 1); (15, 13, 6); (16, 16, 2); (17, 16, 4); (35, 16, 3); (22, 19, 6); (42, 20, 1); (11, 21, 2); (26, 21, 1); (25, 23, 1); (43, 23, 2); (45, 23, 4); (6, 27, 4); (48, 28, 4);

Time 30

Waiting clients: (14, 31, 2); (19, 31, 1); (24, 31, 6); (29, 31, 3); (18, 35, 5); (38, 35, 5); (7, 37, 3); (33, 38, 6); (46, 39, 4);

Queue 1: (4, 13, 5); (21, 14, 6); (32, 16, 6); (9, 20, 4); (13, 20, 4); (1, 23, 4); (28, 24, 3); (39, 25, 4); (37, 28, 4); (8, 29, 4); (41, 30, 4);

Queue 2: (15, 13, 6); (16, 16, 2); (17, 16, 4); (35, 16, 3); (22, 19, 6); (42, 20, 1); (11, 21, 2); (26, 21, 1); (25, 23, 1); (43, 23, 2); (45, 23, 4); (6, 27, 4); (48, 28, 4); (2, 30, 2); (3, 30, 5);

Time 31

Waiting clients: (18, 35, 5); (38, 35, 5); (7, 37, 3); (33, 38, 6); (46, 39, 4);

Queue 1: (4, 13, 4); (21, 14, 6); (32, 16, 6); (9, 20, 4); (13, 20, 4); (1, 23, 4); (28, 24, 3); (39, 25, 4); (37, 28, 4); (8, 29, 4); (41, 30, 4); (19, 31, 1); (24, 31, 6);

Queue 2: (15, 13, 5); (16, 16, 2); (17, 16, 4); (35, 16, 3); (22, 19, 6); (42, 20, 1); (11, 21, 2); (26, 21, 1); (25, 23, 1); (43, 23, 2); (45, 23, 4); (6, 27, 4); (48, 28, 4); (2, 30, 2); (3, 30, 5); (14, 31, 2); (29, 31, 3);

Time 32

Waiting clients: (18, 35, 5); (38, 35, 5); (7, 37, 3); (33, 38, 6); (46, 39, 4);

Queue 1: (4, 13, 3); (21, 14, 6); (32, 16, 6); (9, 20, 4); (13, 20, 4); (1, 23, 4); (28, 24, 3); (39, 25, 4); (37, 28, 4); (8, 29, 4); (41, 30, 4); (19, 31, 1); (24, 31, 6);

Queue 2: (15, 13, 4); (16, 16, 2); (17, 16, 4); (35, 16, 3); (22, 19, 6); (42, 20, 1); (11, 21, 2); (26, 21, 1); (25, 23, 1); (43, 23, 2); (45, 23, 4); (6, 27, 4); (48, 28, 4); (2, 30, 2); (3, 30, 5); (14, 31, 2); (29, 31, 3);

Time 33

Waiting clients: (18, 35, 5); (38, 35, 5); (7, 37, 3); (33, 38, 6); (46, 39, 4);

Queue 1: (4, 13, 2); (21, 14, 6); (32, 16, 6); (9, 20, 4); (13, 20, 4); (1, 23, 4); (28, 24, 3); (39, 25, 4); (37, 28, 4); (8, 29, 4); (41, 30, 4); (19, 31, 1); (24, 31, 6);

Queue 2: (15, 13, 3); (16, 16, 2); (17, 16, 4); (35, 16, 3); (22, 19, 6); (42, 20, 1); (11, 21, 2); (26, 21, 1); (25, 23, 1); (43, 23, 2); (45, 23, 4); (6, 27, 4); (48, 28, 4); (2, 30, 2); (3, 30, 5); (14, 31, 2); (29, 31, 3);

Time 34

Waiting clients: (18, 35, 5); (38, 35, 5); (7, 37, 3); (33, 38, 6); (46, 39, 4);

Queue 1: (4, 13, 1); (21, 14, 6); (32, 16, 6); (9, 20, 4); (13, 20, 4); (1, 23, 4); (28, 24, 3); (39, 25, 4); (37, 28, 4); (8, 29, 4); (41, 30, 4); (19, 31, 1); (24, 31, 6);

Queue 2: (15, 13, 2); (16, 16, 2); (17, 16, 4); (35, 16, 3); (22, 19, 6); (42, 20, 1); (11, 21, 2); (26, 21, 1); (25, 23, 1); (43, 23, 2); (45, 23, 4); (6, 27, 4); (48, 28, 4); (2, 30, 2); (3, 30, 5); (14, 31, 2); (29, 31, 3);

Time 35

Waiting clients: (7, 37, 3); (33, 38, 6); (46, 39, 4);

Queue 1: (21, 14, 6); (32, 16, 6); (9, 20, 4); (13, 20, 4); (1, 23, 4); (28, 24, 3); (39, 25, 4); (37, 28, 4); (8, 29, 4); (41, 30, 4); (19, 31, 1); (24, 31, 6); (38, 35, 5);

Queue 2: (15, 13, 1); (16, 16, 2); (17, 16, 4); (35, 16, 3); (22, 19, 6); (42, 20, 1); (11, 21, 2); (26, 21, 1); (25, 23, 1); (43, 23, 2); (45, 23, 4); (6, 27, 4); (48, 28, 4); (2, 30, 2); (3, 30, 5); (14, 31, 2); (29, 31, 3); (18, 35, 5);

Time 36

Waiting clients: (7, 37, 3); (33, 38, 6); (46, 39, 4);

Queue 1: (21, 14, 5); (32, 16, 6); (9, 20, 4); (13, 20, 4); (1, 23, 4); (28, 24, 3); (39, 25, 4); (37, 28, 4); (8, 29, 4); (41, 30, 4); (19, 31, 1); (24, 31, 6); (38, 35, 5);

Queue 2: (16, 16, 2); (17, 16, 4); (35, 16, 3); (22, 19, 6); (42, 20, 1); (11, 21, 2); (26, 21, 1); (25, 23, 1); (43, 23, 2); (45, 23, 4); (6, 27, 4); (48, 28, 4); (2, 30, 2); (3, 30, 5); (14, 31, 2); (29, 31, 3); (18, 35, 5);

Time 37

Waiting clients: (33, 38, 6); (46, 39, 4);

Queue 1: (21, 14, 4); (32, 16, 6); (9, 20, 4); (13, 20, 4); (1, 23, 4); (28, 24, 3); (39, 25, 4); (37, 28, 4); (8, 29, 4); (41, 30, 4); (19, 31, 1); (24, 31, 6); (38, 35, 5);

Queue 2: (16, 16, 1); (17, 16, 4); (35, 16, 3); (22, 19, 6); (42, 20, 1); (11, 21, 2); (26, 21, 1); (25, 23, 1); (43, 23, 2); (45, 23, 4); (6, 27, 4); (48, 28, 4); (2, 30, 2); (3, 30, 5); (14, 31, 2); (29, 31, 3); (18, 35, 5); (7, 37, 3);

Time 38

Waiting clients: (46, 39, 4);

Queue 1: (21, 14, 3); (32, 16, 6); (9, 20, 4); (13, 20, 4); (1, 23, 4); (28, 24, 3); (39, 25, 4); (37, 28, 4); (8, 29, 4); (41, 30, 4); (19, 31, 1); (24, 31, 6); (38, 35, 5); (33, 38, 6);

Queue 2: (17, 16, 4); (35, 16, 3); (22, 19, 6); (42, 20, 1); (11, 21, 2); (26, 21, 1); (25, 23, 1); (43, 23, 2); (45, 23, 4); (6, 27, 4); (48, 28, 4); (2, 30, 2); (3, 30, 5); (14, 31, 2); (29, 31, 3); (18, 35, 5); (7, 37, 3);

Time 39

Waiting clients:

Queue 1: (21, 14, 2); (32, 16, 6); (9, 20, 4); (13, 20, 4); (1, 23, 4); (28, 24, 3); (39, 25, 4); (37, 28, 4); (8, 29, 4); (41, 30, 4); (19, 31, 1); (24, 31, 6); (38, 35, 5); (33, 38, 6);

Queue 2: (17, 16, 3); (35, 16, 3); (22, 19, 6); (42, 20, 1); (11, 21, 2); (26, 21, 1); (25, 23, 1); (43, 23, 2); (45, 23, 4); (6, 27, 4); (48, 28, 4); (2, 30, 2); (3, 30, 5); (14, 31, 2); (29, 31, 3); (18, 35, 5); (7, 37, 3); (46, 39, 4);

Time 40

Waiting clients:

Queue 1: (21, 14, 1); (32, 16, 6); (9, 20, 4); (13, 20, 4); (1, 23, 4); (28, 24, 3); (39, 25, 4); (37, 28, 4); (8, 29, 4); (41, 30, 4); (19, 31, 1); (24, 31, 6); (38, 35, 5); (33, 38, 6);

Queue 2: (17, 16, 2); (35, 16, 3); (22, 19, 6); (42, 20, 1); (11, 21, 2); (26, 21, 1); (25, 23, 1); (43, 23, 2); (45, 23, 4); (6, 27, 4); (48, 28, 4); (2, 30, 2); (3, 30, 5); (14, 31, 2); (29, 31, 3); (18, 35, 5); (7, 37, 3); (46, 39, 4);

Time 41

Waiting clients:

Queue 1: (32, 16, 6); (9, 20, 4); (13, 20, 4); (1, 23, 4); (28, 24, 3); (39, 25, 4); (37, 28, 4); (8, 29, 4); (41, 30, 4); (19, 31, 1); (24, 31, 6); (38, 35, 5); (33, 38, 6);

Queue 2: (17, 16, 1); (35, 16, 3); (22, 19, 6); (42, 20, 1); (11, 21, 2); (26, 21, 1); (25, 23, 1); (43, 23, 2); (45, 23, 4); (6, 27, 4); (48, 28, 4); (2, 30, 2); (3, 30, 5); (14, 31, 2); (29, 31, 3); (18, 35, 5); (7, 37, 3); (46, 39, 4);

Time 42

Waiting clients:

Queue 1: (32, 16, 5); (9, 20, 4); (13, 20, 4); (1, 23, 4); (28, 24, 3); (39, 25, 4); (37, 28, 4); (8, 29, 4); (41, 30, 4); (19, 31, 1); (24, 31, 6); (38, 35, 5); (33, 38, 6);

Queue 2: (35, 16, 3); (22, 19, 6); (42, 20, 1); (11, 21, 2); (26, 21, 1); (25, 23, 1); (43, 23, 2); (45, 23, 4); (6, 27, 4); (48, 28, 4); (2, 30, 2); (3, 30, 5); (14, 31, 2); (29, 31, 3); (18, 35, 5); (7, 37, 3); (46, 39, 4);

Time 43

Waiting clients:

Queue 1: (32, 16, 4); (9, 20, 4); (13, 20, 4); (1, 23, 4); (28, 24, 3); (39, 25, 4); (37, 28, 4); (8, 29, 4); (41, 30, 4); (19, 31, 1); (24, 31, 6); (38, 35, 5); (33, 38, 6);

Queue 2: (35, 16, 2); (22, 19, 6); (42, 20, 1); (11, 21, 2); (26, 21, 1); (25, 23, 1); (43, 23, 2); (45, 23, 4); (6, 27, 4); (48, 28, 4); (2, 30, 2); (3, 30, 5); (14, 31, 2); (29, 31, 3); (18, 35, 5); (7, 37, 3); (46, 39, 4);

Time 44

Waiting clients:

Queue 1: (32, 16, 3); (9, 20, 4); (13, 20, 4); (1, 23, 4); (28, 24, 3); (39, 25, 4); (37, 28, 4); (8, 29, 4); (41, 30, 4); (19, 31, 1); (24, 31, 6); (38, 35, 5); (33, 38, 6);

Queue 2: (35, 16, 1); (22, 19, 6); (42, 20, 1); (11, 21, 2); (26, 21, 1); (25, 23, 1); (43, 23, 2); (45, 23, 4); (6, 27, 4); (48, 28, 4); (2, 30, 2); (3, 30, 5); (14, 31, 2); (29, 31, 3); (18, 35, 5); (7, 37, 3); (46, 39, 4);

Time 45

Waiting clients:

Queue 1: (32, 16, 2); (9, 20, 4); (13, 20, 4); (1, 23, 4); (28, 24, 3); (39, 25, 4); (37, 28, 4); (8, 29, 4); (41, 30, 4); (19, 31, 1); (24, 31, 6); (38, 35, 5); (33, 38, 6);

Queue 2: (22, 19, 6); (42, 20, 1); (11, 21, 2); (26, 21, 1); (25, 23, 1); (43, 23, 2); (45, 23, 4); (6, 27, 4); (48, 28, 4); (2, 30, 2); (3, 30, 5); (14, 31, 2); (29, 31, 3); (18, 35, 5); (7, 37, 3); (46, 39, 4);

Time 46

Waiting clients:

Queue 1: (32, 16, 1); (9, 20, 4); (13, 20, 4); (1, 23, 4); (28, 24, 3); (39, 25, 4); (37, 28, 4); (8, 29, 4); (41, 30, 4); (19, 31, 1); (24, 31, 6); (38, 35, 5); (33, 38, 6);

Queue 2: (22, 19, 5); (42, 20, 1); (11, 21, 2); (26, 21, 1); (25, 23, 1); (43, 23, 2); (45, 23, 4); (6, 27, 4); (48, 28, 4); (2, 30, 2); (3, 30, 5); (14, 31, 2); (29, 31, 3); (18, 35, 5); (7, 37, 3); (46, 39, 4);

Time 47

Waiting clients:

Queue 1: (9, 20, 4); (13, 20, 4); (1, 23, 4); (28, 24, 3); (39, 25, 4); (37, 28, 4); (8, 29, 4); (41, 30, 4); (19, 31, 1); (24, 31, 6); (38, 35, 5); (33, 38, 6);

Queue 2: (22, 19, 4); (42, 20, 1); (11, 21, 2); (26, 21, 1); (25, 23, 1); (43, 23, 2); (45, 23, 4); (6, 27, 4); (48, 28, 4); (2, 30, 2); (3, 30, 5); (14, 31, 2); (29, 31, 3); (18, 35, 5); (7, 37, 3); (46, 39, 4);

Time 48

Waiting clients:

Queue 1: (9, 20, 3); (13, 20, 4); (1, 23, 4); (28, 24, 3); (39, 25, 4); (37, 28, 4); (8, 29, 4); (41, 30, 4); (19, 31, 1); (24, 31, 6); (38, 35, 5); (33, 38, 6);

Queue 2: (22, 19, 3); (42, 20, 1); (11, 21, 2); (26, 21, 1); (25, 23, 1); (43, 23, 2); (45, 23, 4); (6, 27, 4); (48, 28, 4); (2, 30, 2); (3, 30, 5); (14, 31, 2); (29, 31, 3); (18, 35, 5); (7, 37, 3); (46, 39, 4);

Time 49

Waiting clients:

Queue 1: (9, 20, 2); (13, 20, 4); (1, 23, 4); (28, 24, 3); (39, 25, 4); (37, 28, 4); (8, 29, 4); (41, 30, 4); (19, 31, 1); (24, 31, 6); (38, 35, 5); (33, 38, 6);

Queue 2: (22, 19, 2); (42, 20, 1); (11, 21, 2); (26, 21, 1); (25, 23, 1); (43, 23, 2); (45, 23, 4); (6, 27, 4); (48, 28, 4); (2, 30, 2); (3, 30, 5); (14, 31, 2); (29, 31, 3); (18, 35, 5); (7, 37, 3); (46, 39, 4);

Time 50

Waiting clients:

Queue 1: (9, 20, 1); (13, 20, 4); (1, 23, 4); (28, 24, 3); (39, 25, 4); (37, 28, 4); (8, 29, 4); (41, 30, 4); (19, 31, 1); (24, 31, 6); (38, 35, 5); (33, 38, 6);

Queue 2: (22, 19, 1); (42, 20, 1); (11, 21, 2); (26, 21, 1); (25, 23, 1); (43, 23, 2); (45, 23, 4); (6, 27, 4); (48, 28, 4); (2, 30, 2); (3, 30, 5); (14, 31, 2); (29, 31, 3); (18, 35, 5); (7, 37, 3); (46, 39, 4);

Time 51

Waiting clients:

Queue 1: (13, 20, 4); (1, 23, 4); (28, 24, 3); (39, 25, 4); (37, 28, 4); (8, 29, 4); (41, 30, 4); (19, 31, 1); (24, 31, 6); (38, 35, 5); (33, 38, 6);

Queue 2: (42, 20, 1); (11, 21, 2); (26, 21, 1); (25, 23, 1); (43, 23, 2); (45, 23, 4); (6, 27, 4); (48, 28, 4); (2, 30, 2); (3, 30, 5); (14, 31, 2); (29, 31, 3); (18, 35, 5); (7, 37, 3); (46, 39, 4);

Time 52

Waiting clients:

Queue 1: (13, 20, 3); (1, 23, 4); (28, 24, 3); (39, 25, 4); (37, 28, 4); (8, 29, 4); (41, 30, 4); (19, 31, 1); (24, 31, 6); (38, 35, 5); (33, 38, 6);

Queue 2: (11, 21, 2); (26, 21, 1); (25, 23, 1); (43, 23, 2); (45, 23, 4); (6, 27, 4); (48, 28, 4); (2, 30, 2); (3, 30, 5); (14, 31, 2); (29, 31, 3); (18, 35, 5); (7, 37, 3); (46, 39, 4);

Time 53
Waiting clients:
Queue 1: (13, 20, 2); (1, 23, 4); (28, 24, 3); (39, 25, 4); (37, 28, 4); (8, 29, 4); (41, 30, 4); (19, 31, 1); (24, 31, 6); (38, 35, 5); (33, 38, 6);
Queue 2: (11, 21, 1); (26, 21, 1); (25, 23, 1); (43, 23, 2); (45, 23, 4); (6, 27, 4); (48, 28, 4); (2, 30, 2); (3, 30, 5); (14, 31, 2); (29, 31, 3); (18, 35, 5); (7, 37, 3); (46, 39, 4);

Time 54
Waiting clients:
Queue 1: (13, 20, 1); (1, 23, 4); (28, 24, 3); (39, 25, 4); (37, 28, 4); (8, 29, 4); (41, 30, 4); (19, 31, 1); (24, 31, 6); (38, 35, 5); (33, 38, 6);
Queue 2: (26, 21, 1); (25, 23, 1); (43, 23, 2); (45, 23, 4); (6, 27, 4); (48, 28, 4); (2, 30, 2); (3, 30, 5); (14, 31, 2); (29, 31, 3); (18, 35, 5); (7, 37, 3); (46, 39, 4);

Time 55
Waiting clients:
Queue 1: (1, 23, 4); (28, 24, 3); (39, 25, 4); (37, 28, 4); (8, 29, 4); (41, 30, 4); (19, 31, 1); (24, 31, 6); (38, 35, 5); (33, 38, 6);
Queue 2: (25, 23, 1); (43, 23, 2); (45, 23, 4); (6, 27, 4); (48, 28, 4); (2, 30, 2); (3, 30, 5); (14, 31, 2); (29, 31, 3); (18, 35, 5); (7, 37, 3); (46, 39, 4);

Time 56
Waiting clients:
Queue 1: (1, 23, 3); (28, 24, 3); (39, 25, 4); (37, 28, 4); (8, 29, 4); (41, 30, 4); (19, 31, 1); (24, 31, 6); (38, 35, 5); (33, 38, 6);
Queue 2: (43, 23, 2); (45, 23, 4); (6, 27, 4); (48, 28, 4); (2, 30, 2); (3, 30, 5); (14, 31, 2); (29, 31, 3); (18, 35, 5); (7, 37, 3); (46, 39, 4);

Time 57
Waiting clients:
Queue 1: (1, 23, 2); (28, 24, 3); (39, 25, 4); (37, 28, 4); (8, 29, 4); (41, 30, 4); (19, 31, 1); (24, 31, 6); (38, 35, 5); (33, 38, 6);
Queue 2: (43, 23, 1); (45, 23, 4); (6, 27, 4); (48, 28, 4); (2, 30, 2); (3, 30, 5); (14, 31, 2); (29, 31, 3); (18, 35, 5); (7, 37, 3); (46, 39, 4);

Time 58
Waiting clients:
Queue 1: (1, 23, 1); (28, 24, 3); (39, 25, 4); (37, 28, 4); (8, 29, 4); (41, 30, 4); (19, 31, 1); (24, 31, 6); (38, 35, 5); (33, 38, 6);
Queue 2: (45, 23, 4); (6, 27, 4); (48, 28, 4); (2, 30, 2); (3, 30, 5); (14, 31, 2); (29, 31, 3); (18, 35, 5); (7, 37, 3); (46, 39, 4);

Time 59
Waiting clients:
Queue 1: (28, 24, 3); (39, 25, 4); (37, 28, 4); (8, 29, 4); (41, 30, 4); (19, 31, 1); (24, 31, 6); (38, 35, 5); (33, 38, 6);
Queue 2: (45, 23, 3); (6, 27, 4); (48, 28, 4); (2, 30, 2); (3, 30, 5); (14, 31, 2); (29, 31, 3); (18, 35, 5); (7, 37, 3); (46, 39, 4);

Time 60
Waiting clients:
Queue 1: (28, 24, 2); (39, 25, 4); (37, 28, 4); (8, 29, 4); (41, 30, 4); (19, 31, 1); (24, 31, 6); (38, 35, 5); (33, 38, 6);
Queue 2: (45, 23, 2); (6, 27, 4); (48, 28, 4); (2, 30, 2); (3, 30, 5); (14, 31, 2); (29, 31, 3); (18, 35, 5); (7, 37, 3); (46, 39, 4);

Average Waiting Time: 30.24
Average Service Time: 3.484848484848485
Peak Hour: 39

*** End of Log File ***

Test 3

Test 3
N = 1000
Q = 20
$t_{simulation}^{MAX} = 200$ seconds
$[t_{arrival}^{MIN}, t_{arrival}^{MAX}] = [10, 100]$
$[t_{service}^{MIN}, t_{service}^{MAX}] = [3, 9]$

```
Current Simulation Time: 200

Average Waiting Time: 95.674

Average Service Time: 5.528832630098453

Peak Hour: 99
```

*** Log File is too big to be printed. It will be attached to the documentation ***

Conclusions

As a final note for this assignment, I may say I found it a bit tricky at first but, I think I managed to solve the problems pretty well. I managed to learn how threads should work and the proper way they should be managed and I also learned some new stuff about Java Swing and JavaFX. I used Java Swing for this project, but for the next one, I would like to give JavaFX a try.

Regarding future updates of the application, I would love to ease the GUI processing by implementing it using SwingWorker. Besides that, I would like to simplify and clean the code and also restructure the user interface to make it more responsive.

Bibliography

GUI

- BoxLayout Spacing, StackOverflow
- BoxLayout Usage, GeeksForGeeks
- GridBagLayout, JavaTPoint
- Layouts, Oracle
- RoundButton, StackOverflow
- RoundButton, RoseIndia

Threads

- Thread (computing), Wikipedia
- Scheduler (computing), Wikipedia
- Java Concurrency, Defog Tech
- Multi-Threading, Cave of Programming
- Java Concurrency and Multithreading, Jakob Jenkov
- BlockingQueue in Java, Amit Srivastava
- BlockingQueue Interface, GeeksForGeeks

Diagrams Tools

- Diagram.drawio