Queue Simulator

By Bogdan Tudor-Alexandru

**1) Assignment Objectives**

Design and implement a queue simulator with a dedicated graphical interface through which the user can insert the simulations constraints (Nr of customers, nr of queues, simulation time, mid and max arrival, min and max service) and see the development of the shop during simulation, but also the average waiting time for a customer and the peak hour.

Secondary objectives:

* Analyze the problem and identify the requirements
* Design the queue simulator using threads for concurrent operations of the queues
* Implement the queue simulator
* Test the queue simulator

**2) Analyze the problem and identify the requirements**

Considering that organizing a shop is not and easy task, we want to design and implement a Queue Simulator that given some constraints (such as Nr of customers, nr of queues, simulation time, mid and max arrival, min and max service) we can see what the development of the shop is during that interval. This will help the manager to better organize his workers in queues for a better service towards customers. By analyzing this we can define the functional and non-functional requirements.

Functional Requirements:

* The Queue Simulator should allow the user to easily insert the simulation constraints
* The Queue Simulator should display the live-time development of the store in the GUI

Non-Functional Requirements:

* + The Polynomial calculator should be intuitive to use;
  + The Polynomial calculator should be easy to use.

A) **Use case**: Start Simulation

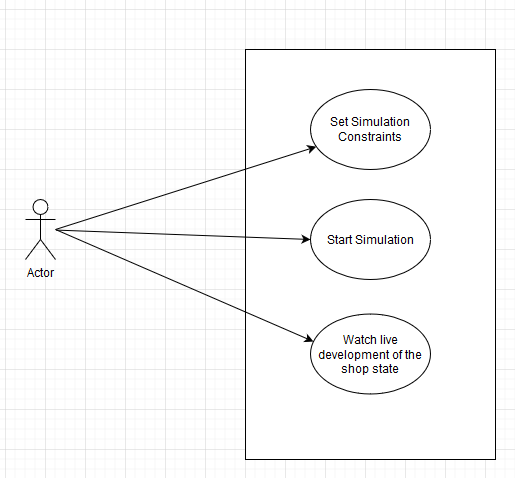
**Primary Actor:** user

**Main Success Scenario:**

* The user inserts the simulation constraints in the TextBoxes on the left of the GUI such as :
  + N – nr of customers
  + M – nr of Queues
  + MaxSimulation – the duration of the simulation
  + MinArrival – The Fastest time a client will arrive at the shop
  + MaxArrival – The latest time a client will arrive at the shop
  + MinService – The shortest time it takes to fully server a customer
  + MaxService – The longest time it takes to fully server a customer
* By pressing the “Start” button the Simulation will begin with the given constraints and it will display the live development of the store (the current waiting customers and the customers in each Queue).

**Alternative Scenario:**

* In case of errors a message will be shown and nothing will be done.

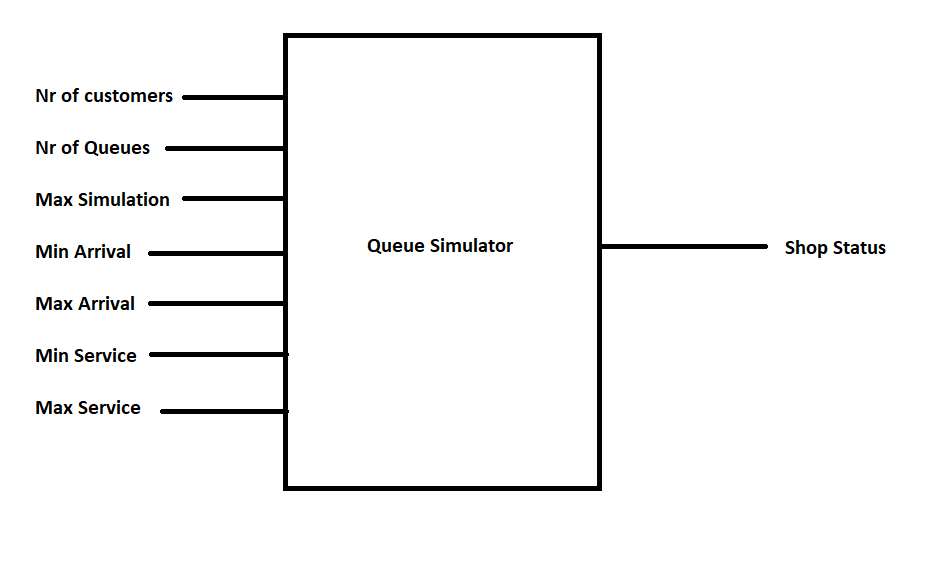


**3) Design the Queue Simulator**

The Design will be implemented in 5 different levels.

**A) Level 1**

In the overall design the user will input 7 numbers constraints (Nr of customers, nr of queues, simulation time, mid and max arrival, min and max service) into the Simulator and by pressing “Start” which will show the development of the store.



**B) Level 2**

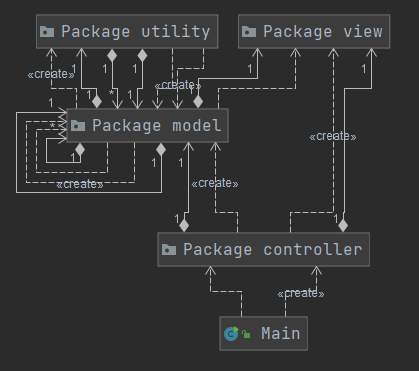
In the second level or design we will split our classes into different packages by using the Model-View-Controller architectural Pattern:

* + **Model components** - encapsulates the core data and functionality
  + **View components** - display information to the user and obtains data it displays from the model
  + **Controller** – This component will deal with receiving input and with the events from mouse (inserting the polynomials and selecting the operations)
  + **Utility**  - It will contain the customers simulator for our Program.

**C) Level 3**

Each package will be split into classes. Base on the Mode-View-Controller Architectural pattern we will have:

* **Model Package**: Customer, CustomerQueue, QueueManager, SimulationConstraints
* **View Package**: the Graphic User Interface class (GUI)
* **Controller**: the Controller class.
* **Utility**: the CustomerGenerator class.



**D) Level 4**

Furthermore, each class will be divided into routines. Each class will have methods for getting or setting data (getters and setters) and constructors. The Specialized methods for each class are:

**Controller**:

* Controller (constructor)

**Customer**:

* decreaseServiceTime
* toString()
* compareTo()

**CustomerQueue**:

* CustomerQueue (constructor)
* Status
* Terminate
* addCustomer
* printContent
* serveCustomer
* run

**SimulationConstraints**:

* SimulationConstraints(constructor)

**QueueManager**:

* QueueManager (constructor)
* waitingCustomers
* printShopStatus
* work
* run

**CustomerGenerator**:

* CustomerGenerator (constructor)
* generateOneCustomer
* removeCustomer
* print

**GUI**:

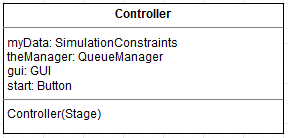
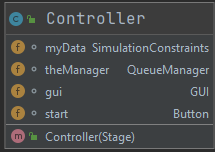
* GUI (constructor)

**E) Level 5**

For the Internal routine design (the implementation) please check the code for more details.

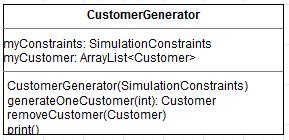
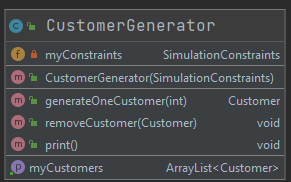
**Controller**:

* Controller (constructor) – to construct the Class with the given parameters



**CustomerGenerator**:

* CustomerGenerator (constructor) - to construct the Class with the given parameters
* generateOneCustomer – it generates one random Customers using the give Constraints
* removeCustomer – removes a customer from the list of generates customers
* print – prints the list of generated customers.



**Customer**:

* decreaseServiceTime – it decreases the service time of the given object
* toString() – generates a string for the given object to print it easier
* compareTo() – method used in the sorting of the objects after they have been generated
* Customer - to construct the Class with the given parameters

**CustomerQueue**:

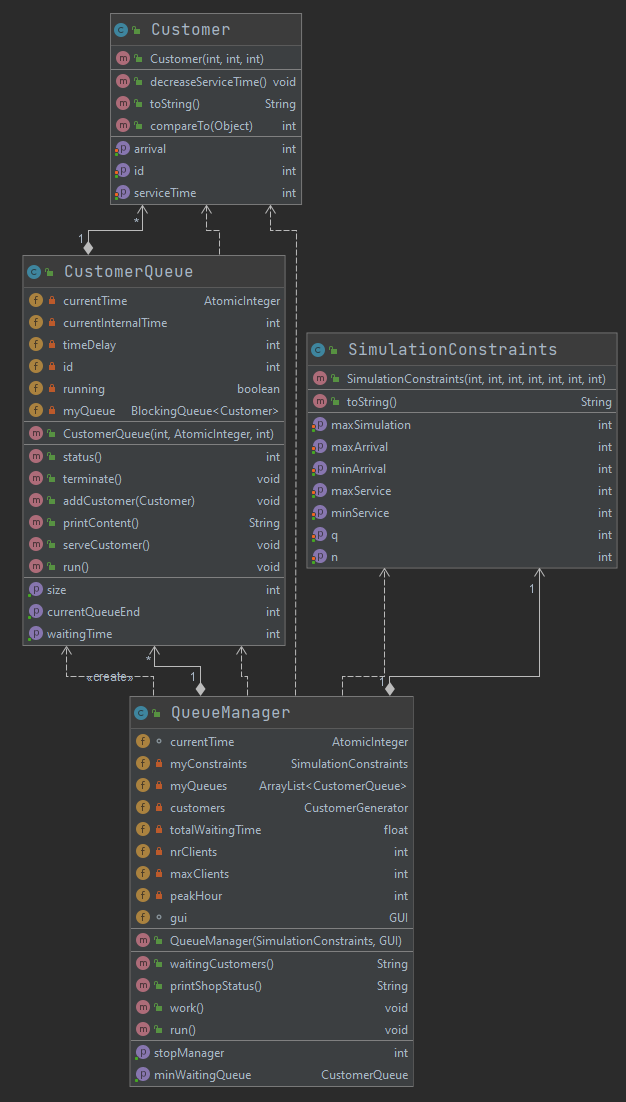
* CustomerQueue (constructor) - to construct the Class with the given parameters
* Status – return 0 for empty and 1 for not empty Queue (0 = closed queue)
* Terminate – sets the running flag to flase
* addCustomer – adds a customer to the given Queue
* printContent – generates a string with the current Queue content
* serveCustomer – decreases the service time of the head customer and is <= 0 removes him
* run – the Runnable method used for implementing the thread that simulates our Queue

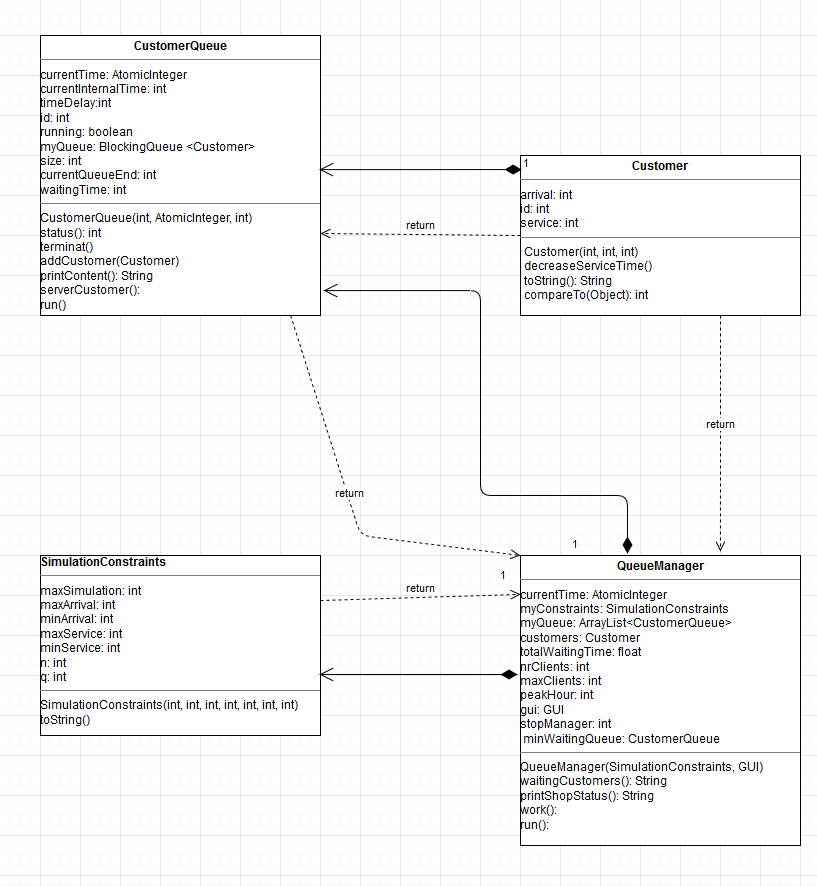
**SimulationConstraints**:

* SimulationConstraints(constructor) - to construct the Class with the given parameters

**QueueManager**:

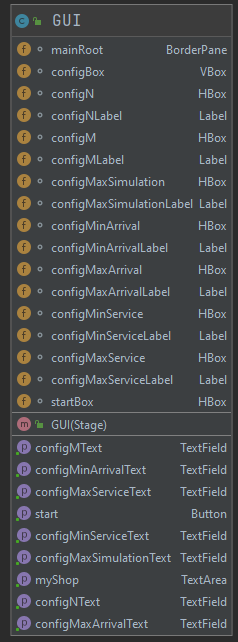
* QueueManager (constructor) - to construct the Class with the given parameters
* waitingCustomers – generates a string with all the customers that have not been disctributed to queues
* printShopStatus – generates a string that contains the current time, the waiting customers and the content of all Queues
* work – gets the Queue with the minimum waiting time and distributes the current customer (if possible) to the given Queue.
* run - the Runnable method used for implementing the thread that simulates our QueueManager





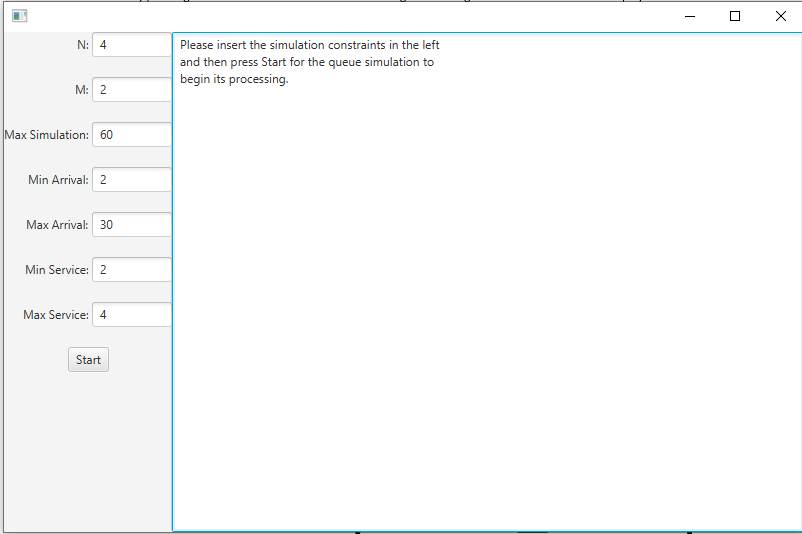
**GUI**:

* GUI (constructor) - to construct the Class with the given parameters



**4) Implementation**

Since the description and inplementation of the classes has been done in the previous part I will just present the Graphical user interface and the Threads used in the implementation of the simulator.



For for Graphical User Interface this time I’ve choosed to use JavaFX. On the Primary Stage I added a Vertical Box that cointaints multi Horizontal Boxes composed of one laber and one text box in which the data cand be introduced but also a TextAre to print the current state of the Store. The Vertical Box for the constraints By inserting the constraints in the given fields and pressing the button “Start” the simulation will begin. For each second there will be displayed: The Time (the current second), The Waiting customers (as a list composed of ID, Arrival, Service time) and the Queues with ther customers in the tem.

Along the main thread I also used another thread for the Queue managers but also some threads for each of the queues. The thread will run as long as the current time (which is an atomic integer but also volatile so it is visible for all the threads) is smaller or equal to the maximum simulation time. The Queue Manager is responsible for creating the Queues with customers, distributing the Customers to the Queue with the minimum waiting time, but also to generate strings with the current content of the store. Each Queue will be another thread that will run as long as the current time (which is an atomic integer but also volatile so it is visible for all the threads gotten from the Queue Manager) is smaller or equal to the maximum simulation time and do one operation per second. Each second they decrease the service time of the customer at the front of the Queue and when this time is <= 0 he will be removed from Queue and we start processing the next Customer.

**5) Results**

Results can be visible in the “log.txt” file along with : Total Waiting time, Average Waiting Time and the Peak Hour (second 😊).

Time 0  
Waiting customers: (1, 5, 3) (2, 14, 4) (3, 20, 3) (4, 21, 4)   
Queue 1: closed  
Queue 2: closed  
  
  
Time 1  
Waiting customers: (1, 5, 3) (2, 14, 4) (3, 20, 3) (4, 21, 4)   
Queue 1: closed  
Queue 2: closed  
  
  
Time 2  
Waiting customers: (1, 5, 3) (2, 14, 4) (3, 20, 3) (4, 21, 4)   
Queue 1: closed  
Queue 2: closed  
  
  
Time 3  
Waiting customers: (1, 5, 3) (2, 14, 4) (3, 20, 3) (4, 21, 4)   
Queue 1: closed  
Queue 2: closed  
  
  
Time 4  
Waiting customers: (1, 5, 3) (2, 14, 4) (3, 20, 3) (4, 21, 4)   
Queue 1: closed  
Queue 2: closed  
  
  
Time 5  
Waiting customers: (2, 14, 4) (3, 20, 3) (4, 21, 4)   
Queue 1: (1, 5, 3)   
Queue 2: closed  
  
  
Time 6  
Waiting customers: (2, 14, 4) (3, 20, 3) (4, 21, 4)   
Queue 1: (1, 5, 2)   
Queue 2: closed  
  
  
Time 7  
Waiting customers: (2, 14, 4) (3, 20, 3) (4, 21, 4)   
Queue 1: (1, 5, 1)   
Queue 2: closed  
  
  
Time 8  
Waiting customers: (2, 14, 4) (3, 20, 3) (4, 21, 4)   
Queue 1: closed  
Queue 2: closed  
  
  
Time 9  
Waiting customers: (2, 14, 4) (3, 20, 3) (4, 21, 4)   
Queue 1: closed  
Queue 2: closed  
  
  
Time 10  
Waiting customers: (2, 14, 4) (3, 20, 3) (4, 21, 4)   
Queue 1: closed  
Queue 2: closed  
  
  
Time 11  
Waiting customers: (2, 14, 4) (3, 20, 3) (4, 21, 4)   
Queue 1: closed  
Queue 2: closed  
  
  
Time 12  
Waiting customers: (2, 14, 4) (3, 20, 3) (4, 21, 4)   
Queue 1: closed  
Queue 2: closed  
  
  
Time 13  
Waiting customers: (2, 14, 4) (3, 20, 3) (4, 21, 4)   
Queue 1: closed  
Queue 2: closed  
  
  
Time 14  
Waiting customers: (3, 20, 3) (4, 21, 4)   
Queue 1: closed  
Queue 2: (2, 14, 4)   
  
  
Time 15  
Waiting customers: (3, 20, 3) (4, 21, 4)   
Queue 1: closed  
Queue 2: (2, 14, 3)   
  
  
Time 16  
Waiting customers: (3, 20, 3) (4, 21, 4)   
Queue 1: closed  
Queue 2: (2, 14, 2)   
  
  
Time 17  
Waiting customers: (3, 20, 3) (4, 21, 4)   
Queue 1: closed  
Queue 2: (2, 14, 1)   
  
  
Time 18  
Waiting customers: (3, 20, 3) (4, 21, 4)   
Queue 1: closed  
Queue 2: closed  
  
  
Time 19  
Waiting customers: (3, 20, 3) (4, 21, 4)   
Queue 1: closed  
Queue 2: closed  
  
  
Time 20  
Waiting customers: (4, 21, 4)   
Queue 1: (3, 20, 3)   
Queue 2: closed  
  
  
Time 21  
Waiting customers:   
Queue 1: (3, 20, 2)   
Queue 2: (4, 21, 4)   
  
  
Time 22  
Waiting customers:   
Queue 1: (3, 20, 1)   
Queue 2: (4, 21, 3)   
  
  
Time 23  
Waiting customers:   
Queue 1: closed  
Queue 2: (4, 21, 2)   
  
  
Time 24  
Waiting customers:   
Queue 1: closed  
Queue 2: (4, 21, 1)   
  
  
Time 25  
Waiting customers:   
Queue 1: closed  
Queue 2: closed  
  
  
Time 26  
Waiting customers:   
Queue 1: closed  
Queue 2: closed  
  
  
Time 27  
Waiting customers:   
Queue 1: closed  
Queue 2: closed  
  
  
Time 28  
Waiting customers:   
Queue 1: closed  
Queue 2: closed  
  
  
Time 29  
Waiting customers:   
Queue 1: closed  
Queue 2: closed  
  
  
Time 30  
Waiting customers:   
Queue 1: closed  
Queue 2: closed  
  
  
 … (for further details please check the log.txt file)  
  
  
  
Total waiting time = 0.0  
Average waiting time = 0.0  
Peak Hour = 21

**6) Conclusions**

This application was an interesting one and I got the opportunity to learn more about OOP and to deepen my knowledge in this field. I also got the opportunity to use the knowledge that I got in the first semester and to implement a project based on what I learned. This time I learned to work with Threads (Concurrency being considered an interesting topic for me, especially regarding optimization) but to also experience with JavaFX instead of Swing. I Think that this Simulator is a good tool for designing a shop and to help the shop manager to make better decision regarding the money spent and the good opinion of the clients with regards to the shop, a small waiting time always making clients happier.

**7) Bibliography**

<http://docs.oracle.com/javase/tutorial/essential/concurrency/index.html>

<http://www.tutorialspoint.com/java/util/timer_schedule_period.htm>

<http://www.javacodegeeks.com/2013/01/java-thread-pool-example-using-executors-and>

<https://www.youtube.com/watch?v=TCd8QIS-2KI&t=1858s>

<https://www.youtube.com/watch?v=J09TLPgwd0Y&t=225s>