DOCUMENTATION

ASSIGNMENT 2

STUDENT NAME: GRAD LAURENȚIU-CĂLIN

GROUP: 30425

Contents

[1. Assignment Objective 3](#_Toc163198336)

[2. Problem Analysis, Modeling, Scenarios, Use Cases 3](#_Toc163198337)

[3. Design 3](#_Toc163198338)

[4. Implementation 3](#_Toc163198339)

[5. Results 3](#_Toc163198340)

[6. Bibliography 3](#_Toc163198341)

# Assignment Objective

Design and implement a queues management application which assigns clients to queues such that the waiting time is minimized. Queues are commonly used to model real world domains. The main objective of a queue is to provide a place for a "client" to wait before receiving a "service". The management of queue-based systems is interested in minimizing the time amount their "clients" are waiting in queues before they are served. One way to minimize the waiting time is to add more servers, i.e., more queues in the system (each queue is considered as having an associated processor) but this approach increases the costs of the service supplier. The queues management application should simulate (by defining a simulation time ) 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), 𝑡\_𝑎𝑟𝑟𝑖𝑣𝑎𝑙 (simulation time when they are ready to enter the queue) and 𝑡\_𝑠𝑒𝑟𝑣𝑖𝑐𝑒 (time interval or duration needed to serve the client, i.e. waiting time when the client is in front of the queue). The application tracks the total time spent by every client in the queues and computes the average waiting time. Each client is added to the queue with the minimum waiting time when its 𝑡\_𝑎𝑟𝑟𝑖𝑣𝑎𝑙 time is greater than or equal to the simulation time (𝑡\_𝑎𝑟𝑟𝑖𝑣𝑎𝑙 ≥ 𝑡\_𝑠𝑖𝑚𝑢𝑙𝑎𝑡𝑖𝑜𝑛). The following data should be considered as input data for the application that should be inserted by the user in the application’s user interface:

* Number of clients (N)
* Number of queues (Q)
* Simulation interval (𝑡\_𝑠𝑖𝑚𝑢𝑙𝑎𝑡𝑖𝑜𝑛\_𝑀𝐴𝑋)
* Minimum and maximum arrival time (𝑡\_𝑎𝑟𝑟𝑖𝑣𝑎𝑙\_𝑀𝐼𝑁 ≤ 𝑡\_𝑎𝑟𝑟𝑖𝑣𝑎𝑙 ≤ 𝑡\_𝑎𝑟𝑟𝑖𝑣𝑎𝑙\_𝑀𝐴𝑋)
* Minimum and maximum service time (𝑡\_𝑠𝑒𝑟𝑣𝑖𝑐𝑒\_𝑀𝐼𝑁 ≤ 𝑡\_𝑠𝑒𝑟𝑣𝑖𝑐𝑒 ≤ 𝑡\_s𝑒𝑟𝑣𝑖𝑐𝑒\_𝑀𝐴𝑋)

# Problem Analysis, Modeling, Scenarios, Use Cases

The problem aims to model a real-life situation: choosing the optimal queue (out of a certain number) when waiting at a public counter, for example. One wants to finish what he/she has to do as quickly as possible so he/she tries to pick the optimal queue based on some sort of inherent strategy. The problem suggests that there are two main strategies: one picks the shortest waiting line (this is probably the way most of us would do) and one picks the queue with the smallest average waiting time (however, in real life, this means one should know for sure how much each task of each person in front of him takes, let alone the computations required to make an educated guess with such a strategy). The first strategy is of real interest because the results it provides should be similar to what should happen in a real-life scenario. However, the second strategy is the mathematically optimal one because it minimizes the overall working time. The problem statement underlines the requested parameters of the simulation we intend to create: a global time limit of the simulation (*simulation interval*), the number of clients (*N*)and the number of queues these people can go to (*Q*). More limiting parameters are provided in order to manage better the generation of people with random tasks: arrival time constants (*MIN* and *MAX*) specify the bounds of the time interval when people may appear in the simulation (and should be distributed to the corresponding optimal queue) and the service time constants (*MIN* and *MAX*) define the interval of time a task may take. These additional parameters may prove useful especially for statistical analysis.

The Queue Simulator Application is designed to create the environment mentioned above. It generates *N* clients, each aiming to solve a single task that takes somewhere between *service\_time\_MIN* and *service\_time\_MAX*. The time stamp when any of these clients appear is randomly generated and is somewhere in the interval [*arrival\_time\_MIN*; *arrival\_time\_MAX*]. After the task generation occurs, the application launches *Q* queues (threads) that process in parallel the people that are corresponding to each one. The application uses this information to generate a window that simulates time stamp by time stamp how the queues evolve from time 0 until the time limit.

The application is intended to work under some assumptions:

1. The source files are run inside a Java Integrated Development Environment such as IntelliJ Idea, Eclipse IDE etc.
2. Once the project is opened with an IDE, the *.main()* method can be run.
3. A pop-up window should appear that asks for the mentioned input parameters. In addition to them a strategy to position in the queue should be chosen. The “pick the shortest queue at current time” strategy is the default chosen policy.
4. If all the parameters are inserted, the “START SIMMULATION” button can be pressed and the window changes into the simulation panel. If any parameter is not a valid number, the console will print an exception message and the application will not start. However, the application will not halt or close.
5. The application will start the simulation and wait indefinitely (even if the simulation finishes) until the close button is pressed.

NOTE: At any time, the application can be closed if the close button (X) in the top right corner is pressed.

The user diagram describes the dependencies between the user’s interactions and the system represented by the Queue Simulator Application. The user tries to follow the simulation that he/she has described conceptually by the already fed input parameters.

# Design

# Implementation

# Results

# Bibliography

1. [Unified Modeling Language - Wikipedia](https://en.wikipedia.org/wiki/Unified_Modeling_Language)
2. [Use Case Diagrams | Unified Modeling Language (UML) - GeeksforGeeks](https://www.geeksforgeeks.org/use-case-diagram/)
3. Programming Techniques lectures, laboratories, and support materials
4. Paul Deitel, Harvey Deitel, Java How to program (10th edition), Publisher: Pearson Education, Inc., Upper Saddle River, NJ, United States, ISBN: 978-0-13-380780-6, Published: 2015
5. Ștefan Tanasă, Ștefan Andrei, Cristian Olaru, Java: de la 0 la expert (2nd edition), Publisher: Polirom, Iași, Romania, ISBN: 978-973-46-2405-8, Published: 2011