Table

Description automatically generated with low confidence

|  |  |
| --- | --- |
| **Student Details** | |
| **Student Name** | **TP Number** |
| *Rajin Muhtadee Shihab* | *TP059508* |

Table of Contents

[1.0 Abstract 3](#_Toc105138367)

[2.0 Basic Requirements 3](#_Toc105138368)

[2.1 Ensuring the use of only one (1) runway and two (2) gates 3](#_Toc105138369)

[2.2 Ensuring safety and preventing collision of aircrafts in the runway/gates 5](#_Toc105138370)

[2.3 Proper post landing procedures leading up to take off 6](#_Toc105138371)

[2.4 Each step should take some time 7](#_Toc105138372)

[3.0 Additional requirements 8](#_Toc105138373)

[3.1 Concurrent Events 8](#_Toc105138374)

[3.2 Exclusive Events 9](#_Toc105138375)

[4.0 Statistics & Telemetry 11](#_Toc105138376)

[5.0 References 12](#_Toc105138377)

[6.0 Appendix 14](#_Toc105138378)

# 1.0 Abstract

Following the proposal report submitted on week 8, this report contains the outcome of the project. It will highlight the achievements of the overall project. It will explain the concurrency concepts used and processes followed. I will be attaching code snippets where necessary. Everything will be shown as lists of requirements met, requirements not met, and additional requirements met.

# 2.0 Basic Requirements

A list of six basic requirements were provided with the scenario. All six of the basic requirements were met. I will explain the achievements regarding the requirements along with output snippets and relevant code snippets.

## 2.1 Ensuring the use of only one (1) runway and two (2) gates

The scenario states that there should be only one runway. The scenario also states that only one plane will get access to the runway (Either for landing or takeoff) at a time. We have set a constant value of one (1) for the quantity of runway and used it as a semaphore to grant access to the attribute to one plane thread at a time. As a semaphore, if the attribute is being used by a thread (Runway being used by a plane), it will lock itself or deny any other access to itself by showing the number of available instances of the attribute as zero (0). A semaphore will be used for the attribute of gates as well, with the initial value set to a constant of two (2). The arrival times of the planes are randomized, mimicking a real-world scenario.

**Semaphore:** Semaphores are integer variables that has a constant maximum value, and it grants access to itself through the use of a counter. If no instances of the semaphore is available, it will not grant any further access until and instance is released.

A picture containing text

Description automatically generated

Figure 2.1.1 – Code Snippet to show the implementation of a random number generator to dictate arrival time

Graphical user interface, text

Description automatically generated

Figure 2.1.2 – Code Snippet to show the implementation of a single runway

Text

Description automatically generated

Figure 2.2.3 – Code snippet to show the implementation of only two (2) gates

## 2.2 Ensuring safety and preventing collision of aircrafts in the runway/gates

As a key safety aspect of concurrent concept, we have used semaphores to make sure that the threads do not overlap or share an attribute that is only meant to be used by a select number of threads at a time (one thread for the runway, two for the gates).

Graphical user interface, text, application, email

Description automatically generated

Figure 2.2.1 – Code snippet for providing access to the runway and preventing access when in use

Text

Description automatically generated

Figure 2.2.2 – Code snippet for releasing instance of runway semaphore

Graphical user interface, text, application, email

Description automatically generated

Figure 2.2.3 – Code snippet for providing access to the gates and preventing access when both instances are in use

Graphical user interface, text, chat or text message

Description automatically generated

Figure 2.2.4 – Code snippet for releasing instance of gates semaphore

## 

## 2.3 Proper post landing procedures leading up to take off

Once an aircraft acquires permission to land, it would proceed to a gate, let the passengers disembark and new passengers embark, refuel and resupply before requesting runway permission again for takeoff.

Text

Description automatically generated

Figure 2.3.1 – Code snippet for post arrival processes leading up to departure

## 2.4 Each step should take some time

The simulations is set up in such a way that, upon arrival, each step is set to consume an amount of time. The time consumed is not preset, so it can be within a logical range. We have used an ‘enum’ class in this task.

‘**enum’ class: ‘**enum’ class is basically a class containing comma separated constants. Since we need constants to set up a time for each step of the post arrival process, an ‘enum’ class is an ideal method to use.



Figure 2.4.1 – Code snippet for an example of setting how much time a process takes

Graphical user interface, text, application

Description automatically generated

Figure 2.4.2 – Code snippet for an example of setting how much time a process takes

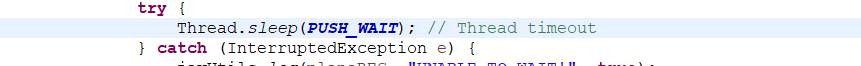


Figure 2.4.3 – Code snippet for enforcing a set time onto a thread to portray duration of process

# 3.0 Additional requirements

## 3.1 Concurrent Events

As additional requirements, some events need to happen concurrently. The events are as follows:

* Passengers Disembarking/ Embarking
* Resupply & Cleaning

Graphical user interface, text, application

Description automatically generated with medium confidence

Figure 3.1.1 – Code snippet to set up concurrency of specific events

Text

Description automatically generated with medium confidence

Figure 3.1.1 – Code snippet to dictate concurrency of specific events

## 3.2 Exclusive Events

As there is only one refueling truck, only one aircraft can be refueled at a time. Attempt at achieving this is based on the flow of programs that dictate concurrency (Refer to figure 3.1.1). The timers are set in such a way that for most instances, two aircrafts will not be able to refuel at the same time.

Graphical user interface, text, application

Description automatically generated

Figure 3.1.2 – Code snippet to dictate flow of exclusive event

Table

Description automatically generated

Figure 3.1.3 – Terminal output portraying exclusive event (Refueling)

# 4.0 Statistics & Telemetry

An algorithm was implemented into the simulation to gather data about elapsed time while accessing the runway as well as while accessing the gates. A minimum, maximum, and average telemetry for there two phases are calculated and printed out on the console. There is also a passenger count system, that assumes that every time the plane arrives or leaves, it does so with 25 passengers, and thus leading up 150 passengers travelling both ways.

Text

Description automatically generated with medium confidence

Figure 4.0.1 – Code snippet for implementing a telemetry system (Timings)

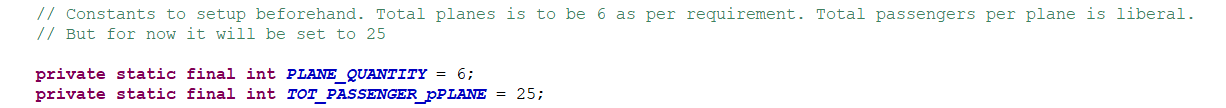


Figure 4.0.2 – Code snippet for implementing a telemetry system (Plane and Passenger Information)

Text

Description automatically generated

Figure 4.0.2 – Terminal output portraying telemetry/statistics

# 5.0 References

* *Arrays stream() method in Java - GeeksforGeeks*. GeeksforGeeks. (2022). Retrieved 3 June 2022, from <https://www.geeksforgeeks.org/arrays-stream-method-in-java/>.
* *For-each loop in Java - GeeksforGeeks*. GeeksforGeeks. (2022). Retrieved 3 June 2022, from <https://www.geeksforgeeks.org/for-each-loop-in-java/>.
* *LocalTime (Java Platform SE 8 )*. Docs.oracle.com. (2022). Retrieved 3 June 2022, from <https://docs.oracle.com/javase/8/docs/api/java/time/LocalTime.html>.
* *Java String format() method - javatpoint*. www.javatpoint.com. (2022). Retrieved 3 June 2022, from <https://www.javatpoint.com/java-string-format>.
* *GitHub - hypethral/Airport-Simulation: Concurrent Airport Simulation in Java*. GitHub. (2022). Retrieved 3 June 2022, from <https://github.com/hypethral/Airport-Simulation>.
* *GitHub - jordankzf/concurrent-airport: A multithreaded airport simulated in Java using BlockingQueue.*. GitHub. (2022). Retrieved 3 June 2022, from <https://github.com/jordankzf/concurrent-airport>.
* *GitHub - 73VW/JavaConcurrentAirport: Realization of an airport simulator with concurrent access by threads*. GitHub. (2022). Retrieved 3 June 2022, from <https://github.com/73VW/JavaConcurrentAirport>.
* *String in Java | How to Declare String in Java With Examples*. Edureka. (2022). Retrieved 3 June 2022, from <https://www.edureka.co/blog/java-string/>.
* Java, S. (2022). *String Initialization in Java | Various Examples of String Initialization*. EDUCBA. Retrieved 3 June 2022, from <https://www.educba.com/string-initialization-in-java/>.
* *Logger log() Method in Java with Examples - GeeksforGeeks*. GeeksforGeeks. (2022). Retrieved 3 June 2022, from <https://www.geeksforgeeks.org/logger-log-method-in-java-with-examples/>.
* *Generating random numbers in Java - GeeksforGeeks*. GeeksforGeeks. (2022). Retrieved 3 June 2022, from <https://www.geeksforgeeks.org/generating-random-numbers-in-java/>.
* *Java Enums*. W3schools.com. (2022). Retrieved 3 June 2022, from <https://www.w3schools.com/java/java_enums.asp>.
* *Semaphore in Java - GeeksforGeeks*. GeeksforGeeks. (2022). Retrieved 3 June 2022, from <https://www.geeksforgeeks.org/semaphore-in-java/>.
* GmbH, L. (2022). *Java concurrency (multi-threading) - Tutorial*. Vogella.com. Retrieved 3 June 2022, from <https://www.vogella.com/tutorials/JavaConcurrency/article.html#:~:text=1.1.-,What%20is%20concurrency%3F,the%20interactivity%20of%20the%20program>.

# 6.0 Appendix

Here’s the overall output of the entire runtime of the simulation -

Text, table

Description automatically generated with medium confidence

Text

Description automatically generated with medium confidence

Table

Description automatically generated Table

Description automatically generated Table

Description automatically generated Graphical user interface, application

Description automatically generated