Objective:

This assignment has been designed for students to apply appropriate concurrent program methods in **implementing** a concurrent program from a program specification.

Learning Outcomes

ON COMPLETION OF THIS ASSIGNMENT, YOU SHOULD BE ABLE TO DEMONSTRATE THE FOLLOWING LEARNING OUTCOME(S):

No.	Learning Outcome	Assessment
1	Explain the fundamental concepts of concurrency and parallelism in the design of a concurrent system (C2, PLO1)	Exam
2	**Apply the concepts of concurrency and parallelism in the construction of a system using a suitable programming language. (C3, PLO2)**	**Individual Assignment (System)**
3	**Explain the safety aspects of multi-threaded and parallel systems (A3, PLO6)**	**Individual Assignment (Report)**

Programme Outcomes (PO):

PLO2 Cognitive Skills - This relates to thinking or intellectual capabilities and the ability to apply knowledge and skills. The capacity to develop levels of intellectual skills progressively begins from understanding, critical/creative thinking, assessment, and applying, analysing, problem solving as well as synthesizing to create new ideas, solutions, strategies, or new practices. Such intellectual skills enable the learner to search and comprehend new information from different fields of knowledge and practices.

Individual Assignment - Report (20%):

Question No.	Topic	Question Vs Taxonomy					PLO
		Affective Level					
		1	2	3	4	5	
		SQ	SQ	SQ	SQ	SQ	
1	Introduction and background			**20%**			6
2	Explanation of the safety aspects of multi-threaded system implemented			**30%**			6
3	Justification of coding techniques implemented.			**30%**			6
4	Depth of discussion of concurrency concepts.			**20%**			6
	Total			**100%**			

Individual Assignment - System (25%):

Question No.	Topic	Question Vs Taxonomy Cognitive Level						PLO
		1	2	3	4	5	6	
		SQ	SQ	SQ	SQ	SQ	SQ	
1	Appropriateness of coding techniques used to implement design with appropriate comment lines in source codes			**20%**				2
2	Appropriateness of the Java concurrent programming facilities used.			**20%**				2
3	Program runs appropriately with basic requirements			**20%**				2
4	Additional requirements met			**20%**				2
5	Explanations of concurrency concepts implemented with relevant code samples			**20%**				2
	Total			**100%**				

Submission Requirements:

Assignment Handout Date : 26th August 2025
Assignment Due Date : 26th October 2025

Case Study

Asia Pacific Airport

You have been tasked to simulate the operations of the airport.

********************************Basic Requirements********************************

- There is only 1 runway for all planes to land and depart.
- There can only be 3 airplanes on the airport grounds, including the runway. This is to ensure that the aircraft does not collide with another aircraft on the runway or gates.
- Once an aircraft obtains permission to land, it should land on the runway, coast to the assigned gate, dock to the gate, allow passengers to disembark, refill supplies and fuel, receive new passengers, undock, coast to the assigned runway and take-off.
- **Each step should take some time.**
- As the airport is small, there is no waiting area on the ground for the planes to wait for a gate to become available.

- These events should happen concurrently:
  - Passengers disembarking/embarking from the 3 gates.
  - Refill supplies and cleaning of aircraft
  - Refuelling of aircraft

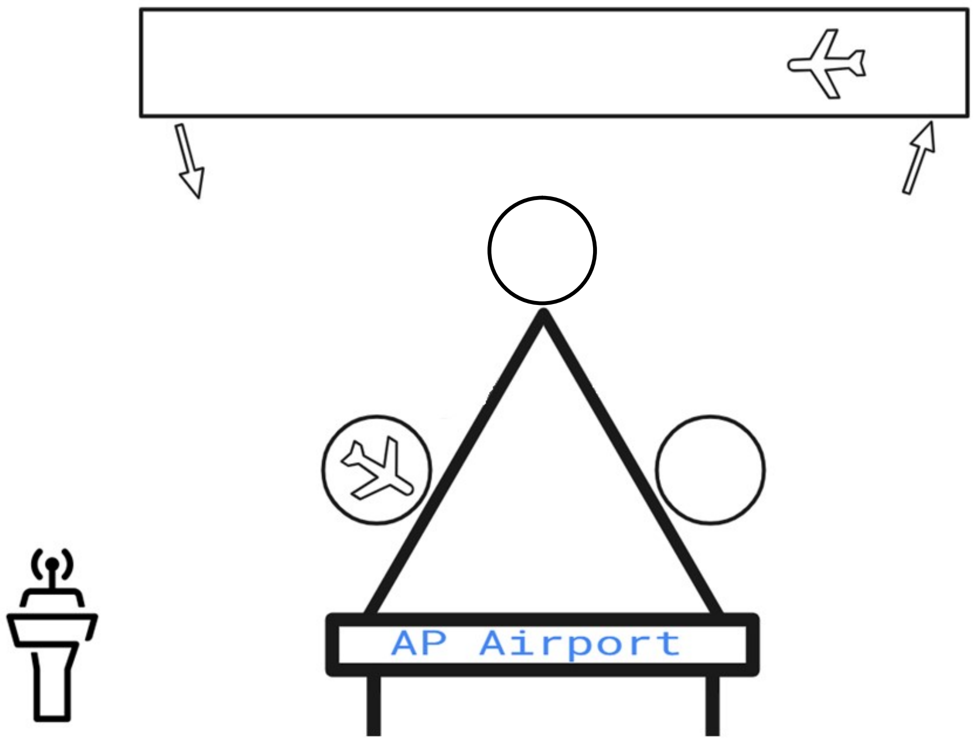

Figure 1: Layout of the Asia Pacific Airport (Not to scale)

******************************Additional Requirements******************************

- There is only 1 refuelling truck to perform:

  - Refuelling of aircraft

- A congested scenario should be simulated where 2 planes are waiting to land while the 2 gates are occupied, and a 3rd plane comes in with fuel shortage, requiring emergency landing.

NOTE: State your assumptions and how you will implement them.

The Statistics (Part of the Basic Requirements)

At the end of the simulation, i.e., when all planes have left the airport, the ATC manager should do some sanity checks of the airport and print out some statistics on the run. The result of the sanity checks must be printed. You must

- Check that all gates are indeed empty.
- Print out statistics on

  - Maximum/Average/Minimum waiting time for a plane.

  - Number of planes served/Passengers boarded.

Deliverables:

For this exercise, you are to model the ATC scenario and design a Java program to simulate activity for the airport:

- Altogether, 6 planes should try to land at the airport.
- Use a random number generator, so a new airplane arrives every 0, 1, or 2 seconds. (This *might* be accomplished by an appropriate statement sleep (rand.nextInt(2000));
- Assume each plane can accommodate maximum 50 passengers.
- Assume passengers are always ready to embark and disembark the terminal (i.e., no capacity issues inside the passenger terminals)

Sample Output

In order to see what is happening dynamically you must have a single output from every plane's passengers, the air traffic controller, and the pilots reporting all their major events. As shown in Figure 2 below:

```
Plane-1: Requesting Landing.
ATC: Landing Permission granted for Plane-1.
ATC: Gate-1 assigned for Plane-1.
Plane-1: Landing.
Plane-1: Landed.
Plane-2: Requesting Landing.
Plane-1: Coasting to Gate-1.
ATC: Landing Permission granted for Plane-2.
Plane-2: Landing.
Plane-1: Docked at Gate-1.
Plane-1's Passengers: Disembarking out of Plane-1.
        •
        •
        •
Plane-4: Requesting Landing.
ATC: Landing Permission Denied for Plane-4, Airport Full.
Plane-1: Requesting Taking off.
ATC: Taking-off is granted for Plane-1. Runway is free.
Plane-1: Taking-off.
        •
        •
        •
```

Figure 2: Correct / Desired Output

Add information about which process/thread is doing the output. This way you can see if a process/thread acts for another, **which is strictly forbidden**, but is a common error for Java solutions (objects are not processes!). An example of such **incorrect behaviour**, see Figure 3 below:

```
Thread-ATC : Plane 5: Requesting permission to land!
MainThread : ATC: Please wait and join the circle queue.
Thread-Passenger-7 : I'm boarding Plane 2 now.
Thread-Passenger-8 : I'm boarding Plane 2 now.
Thread-Passenger-9 : I'm boarding Plane 2 now.
```

Figure 3: Incorrect / Undesired Output

Where we can see that the ATC's thread is acting for Plane 5, and also the main thread is acting for the ATC. Additionally, we can see that Each passenger is acting individually.

Implementation

You should implement your simulation in Java.

The simulation run should not take more than **60 seconds** to simulate.

Documentation for System (Report)

The documentation should detail the system implementation and testing.

1. Basic requirements met:
    - List of requirements met.
        - Short explanation of concurrency concepts (atomic statements, synchronization, etc) implemented.
        - Code snippet of the Java concurrent programming facilities implemented.

2. Additional requirements met:
    - List of requirements met.
        - Short explanation of concurrency concepts (atomic statements, synchronization, etc) implemented.
        - Code snippet of the Java concurrent programming facilities implemented.

3. Requirements which were **NOT** met:
    - List of basic requirements.
    - List of additional requirements

<u>Should not exceed 1500 words excluding references/appendix/coding.</u>

Submission for System

Please Upload The following **<u>TWO FILES Only</u>**:

1- Full Report Document

2- A **SINGLE zip file named TP0XXXXX CCP.zip** that contains:

a. *Video Presentation of the simulation running. Minimum 3 minutes and Maximum 5 minutes per person.*

i. *Simulate scenarios as stated above.*

ii. Show which requirements are met in the output and corresponding code.

b. Fully Functional Code's Folder: *Java files required to run the simulation.*

Marking Scheme (NOT for student's documentation guidance)

Report (20%)

Criteria	**Total marks**	**Marks awarded**
Assumptions [LO3-PO6]	20	
Explanation of the safety aspects of multi-threaded system implemented [LO3-PO6]	30	
Justification of coding techniques implemented [LO3-PO6]	30	
Depth of discussion of concurrency concepts [LO3-PO6]	20	
TOTAL MARKS	100	

System (25%)

Criteria	**Total marks**	**Marks awarded**
Appropriateness of coding techniques used to implement design with appropriate comment lines in source codes *	20	
Appropriateness of the Java concurrent programming facilities used.	20	
Program runs appropriately with basic requirements *	20	
Additional requirements met *	20	
Explanations of concurrency concepts implemented with relevant code samples *	20	
TOTAL MARKS	100	

- **IMPORTANT:** Failure to attend the final presentation for the assignment will be deemed as non-attendance of the final assessment for this assignment item, resulting in a score of ZERO (0).

- **IMPORTANT:** Students are strictly prohibited from using any Java libraries that handle thread concurrency automatically. All concurrency-related implementations must be explicitly written by the students, without relying on built-in parallel execution mechanisms.

  Restricted Libraries and Features

  The following Java libraries and features are not allowed due to their automatic thread concurrency handling:

  — **java.util.concurrent.ExecutorService**

  — **java.util.concurrent.ForkJoinPool**

  — **java.util.concurrent.CompletableFuture**

  — **ParallelStream** in Java Streams API

— **java.util.Timer** (if used for scheduled execution)

— **Reactive Streams** (e.g., Project Reactor, RxJava)

— **Spring @Async** annotation

— **Vert.x** (if used for automatic event-loop concurrency)

— **java.util.concurrent.PriorityBlockingQueue**

— **… and any other package provides concurrency services other than those taught in the class.**

Any submission using these restricted libraries will be considered non-compliant and subject to penalties. If unsure about a library, consult the instructor before using it.