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						PLO
		Cognitive Level						
		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 : 24th May 2023

Assignment Due Date : 18th August 2023

Case Study

The Problem

This assignment will require you to **implement the terminal simulation**, as broken down in the sections below.

Intention of assignment

Even if valuable to the owner, the simulation is *not* the main purpose of this assignment - indeed, if this was the case there are much better techniques for *simulating* than writing a concurrent program.

The requirement of this assignment is to **implement** a program in which synchronization and communication takes place between several concurrent processes. It is intended to force you to solve (and not simply avoid) a range of interesting synchronisation problems.

Tiny Bus Terminal

Tiny Bus Terminal (TBT) is a small transport hub servicing the community around Bukit Jalil and Sri Petaling. At TBT customers can purchase tickets for travel on one of the frequent minibuses that transport up to 10 passengers on short journeys around KL and the surrounding area. The terminal has

two ticket counters manned by ticket personnel, from which customers can purchase their tickets. The terminal has also installed an automated ticketing machine although it is unreliable and prone to breakdown.

Once in possession of a ticket, passengers may proceed to one of three departure gates (depending on their direction of travel) and wait in the waiting area until their minibus arrives. The waiting area is able to accommodate 10 people, with the remainder having to wait in the terminal foyer. Prior to boarding a minibus passengers must have the ticket checked by the ticket inspector. Unfortunately, due to staff shortages the terminal currently only employs one ticket inspector, who has to go back and forth between each of the departure gates.

As the terminal is relatively small it is important to manage the number of people inside the building at any one time for health and safety purposes. Currently the building is capable of handling 15 individuals in the foyer at any one time (excluding staff), thus should the building reach its max capacity guards at each of the two entrances are instructed to prevent people from entering the terminal. New entrants will be prohibited until such time as the number of people inside the terminal has fallen to below 80% max capacity.

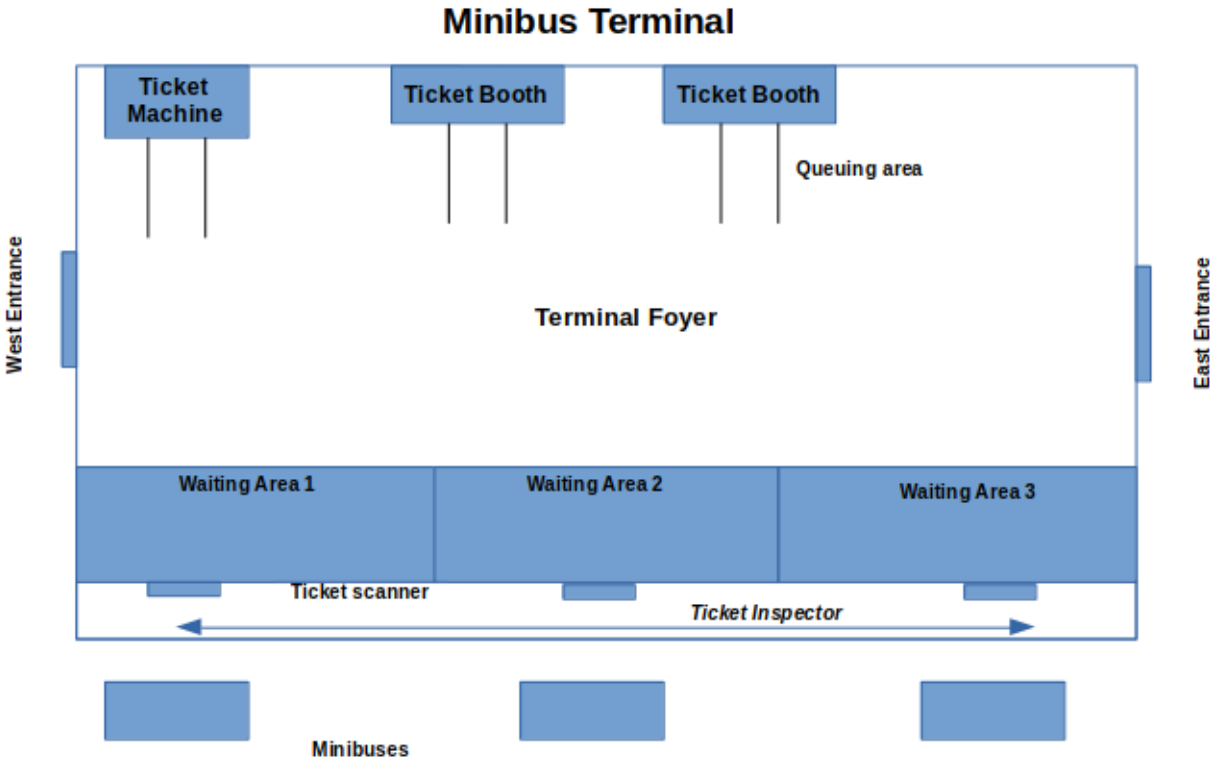

Figure 1: Inside layout of the Asia Pacific Bus Terminal (Not to scale)

Deliverables:

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

For this exercise, you are to model the bus terminal scenario and write a Java program to simulate activity for the terminal:

- Customers entering the terminal building, purchasing tickets, waiting for a bus, and boarding a bus.
- Staff operating the ticket booths, the ticket inspector checking tickets.
- Altogether, 80 passengers should try to enter the building.
- Use a random number generator, so a new customer arrives every 1, 2 or 3 seconds.
- Assume passengers only depart from the terminal (i.e., no one leaves via the entrances, and no one disembarks from each minibus)

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

Simulate scenarios in which:

The ticketing machine is not working – thereby requiring any queuing customers to shift to one of the ticketing booths.

One or more ticketing booth staff are on a toilet break.

The terminal has reached max capacity.

Buses are delayed in arriving at the terminal.

The waiting area is full.

State your assumptions and how you will implement them.

Sample Output

In order to see what is happening dynamically you must have output from the passengers, the ticketing booths, and the busses reporting all their major events.

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 is

Thread-Ticketing-Officer : Passenger 5: One Ticket please!
East Turnstile: Terminal-Guard: Please wait and join the queue.
Thread-Passenger-3: I'm boarding bus 2 now.

Where you can see that not only the Ticketing-Officer thread is acting for passenger 5, but also the East Turnstile thread is acting for the Terminal-Guard.

You *must not*

- Kill a thread or process. You may not use any of the following primitives in Java:
    - Thread.stop
    - Thread.resume
    - Thread.suspend
    - Thread.interrupt

  You may not use the destroy or stop(0) primitives in - except to take care of temporary resources like simple timers.

Implementation

You should implement your simulation in Java.

Each 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

Should not exceed 1500 words excluding references/appendix/coding.

Submission for System

- *Java files required to run the simulation.*
    - *Zipped into a single zip file named TP0XXXXX CCP.zip*
- *Video of the simulation running. Maximum 5 minutes per person.*
    - *Simulate scenarios as stated above.*
    - *Show which requirements are met in the output and corresponding code.*
    - *The video file should be named TP0XXXXX CCP Video.zip*

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	

*Based on video presentation of the simulation.