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 1	2	3	4	5	6	PLO
		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:

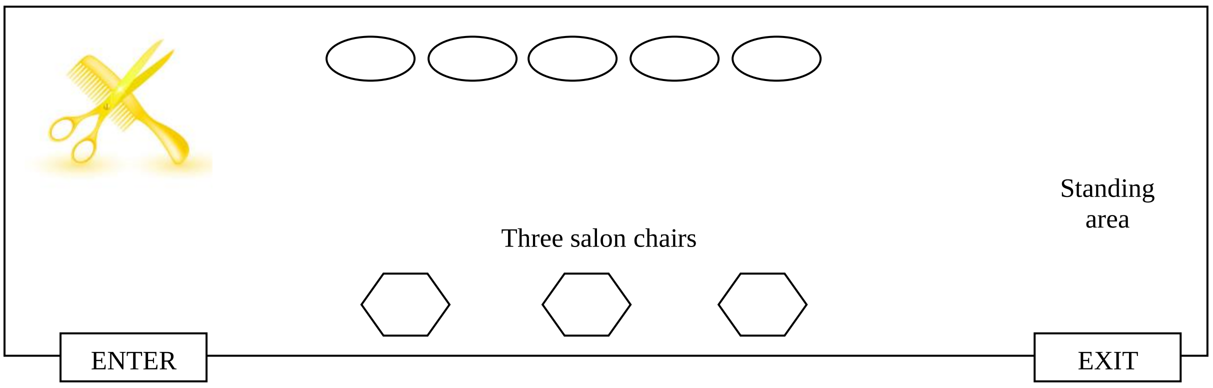
Assignment Handout Date : 17th August 2023

Assignment Due Date : 10th November 2023

Case Study

The Problem

The Golden Sleepy Salon is famous for its gold comb and scissors.



Three hairdressers work independently in a salon shop:

The salon has 3 salon chairs, each of which is assigned to one hairdresser.

Due to budget restrictions, there are **only 2 gold combs and 2 gold scissors** in the salon.

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

Each hairdresser follows the same work plan:

- The hairdresser sleeps when no customer is waiting (and is not in the hairdresser's own chair).
- When the hairdresser is asleep, the hairdresser waits to be awakened by a new customer. (Any available hairdresser can be told to wake up if multiple hairdressers are asleep.)
- Once awake, the hairdresser cuts the hair of a customer in the hairdresser's chair.
- The hairdresser requires a comb and a pair of scissors to cut a customer’s hair. When the haircut is done, the customer pays the hairdresser and then is free to leave.
- After receiving payment, the hairdresser calls the next waiting customer (if any). If such a customer exists, that customer sits in the hairdresser's chair and the hairdresser starts the next haircut. If no customer is waiting, the hairdresser goes back to sleep.

Each customer follows the following sequence of events.

- When the customer first enters the salon, the customer leaves immediately if more than 10 people are waiting (5 standing and 5 sitting). On the other hand, if the salon is not too full, the customer enters and waits.
- If at least one hairdresser is sleeping, the customer wakes up the hairdresser who is sleeping, and sits in that hairdresser’s chair (after the hairdresser has stood up).
- If all hairdressers are busy, the customer sits in a waiting-room chair, if one is available. Otherwise, the customer remains standing until a waiting-room chair becomes available.

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

Customers keep track of their arrival, so the person sitting the longest is always the next customer to get a haircut.

Similarly, standing customers remember their order, so the person standing the longest takes the next available waiting-room seat.

The salon should close after all customers have left and all barbers are sleeping.

Customers haircut progress should be seen.

Which comb and scissors are used by which hairdresser should be clearly stated.

Each event should take some time.

The rate should be based on the duration of the haircut.

****

Extra points: Some customers who are standing get tired after a while and leave the salon.

Deliverables:

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

Simulate each hairdresser and each customer as a separate process.

Altogether, 20 customers should try to enter.

Use a random number generator, so a new customer arrives every 0, 1, or 2 seconds.

Similarly, use a random number generator, so each haircut lasts between 3 and 6 seconds.

Each hairdresser should report when he/she starts each haircut and when he/she finishes each haircut.

Each customer should report when he/she enters the salon. The customer also should report if he/she decides to leave immediately.

Similarly, if the customer must stand or sit in the waiting room, the customer should report when each activity begins.

Finally, the customer should report when the haircut begins and when the customer finally exits the shop.

Sample Output

In order to see what is happening dynamically you must have output from the Customers and the hairdressers 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-Hairdresser: 21.31: Hairdresser1: Customer 3 is done!
main: 21.50: Hairdresser: Next customer please!
Thread-Customer-12 : 21.50: Customer12 is waiting for a chair.
Thread-Hairdresser: 21.31: Hairdresser2: Acquiring comb2!

Where you can see that not only the hairdresser thread but also the main thread is acting for the hairdresser.

Note that realistic time stamps are not required, it is fine to use any function to generate them.

You *must not*

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

If *any* of those primitives are found in your code, you will fail the assignment no matter the functionality of it.

State your assumptions and how you will implement them.

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

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

Submission for System

- *Java files required to run the simulation.*
- *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.*

Both code and presentation Zipped into a single zip file named TP0XXXXX CCP.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.