EXAM PAPER
School of Computing and Information Technology

Course code: CSE 305	**Exam code: 00**
Course name: Software Engineering	**Duration: 120'**
Type of Exam: Open book/ Practice on Computer	**Maximum mark: 100**

Instructions:

Unless otherwise specified, other aspects of the problem are open for the students to interpret and decide on their own. Submit your completed code to Moodle by the end of the exam.

Read the document outlining the functional requirements for a Music Streaming Service (MSS) below, and then proceed with the following tasks:
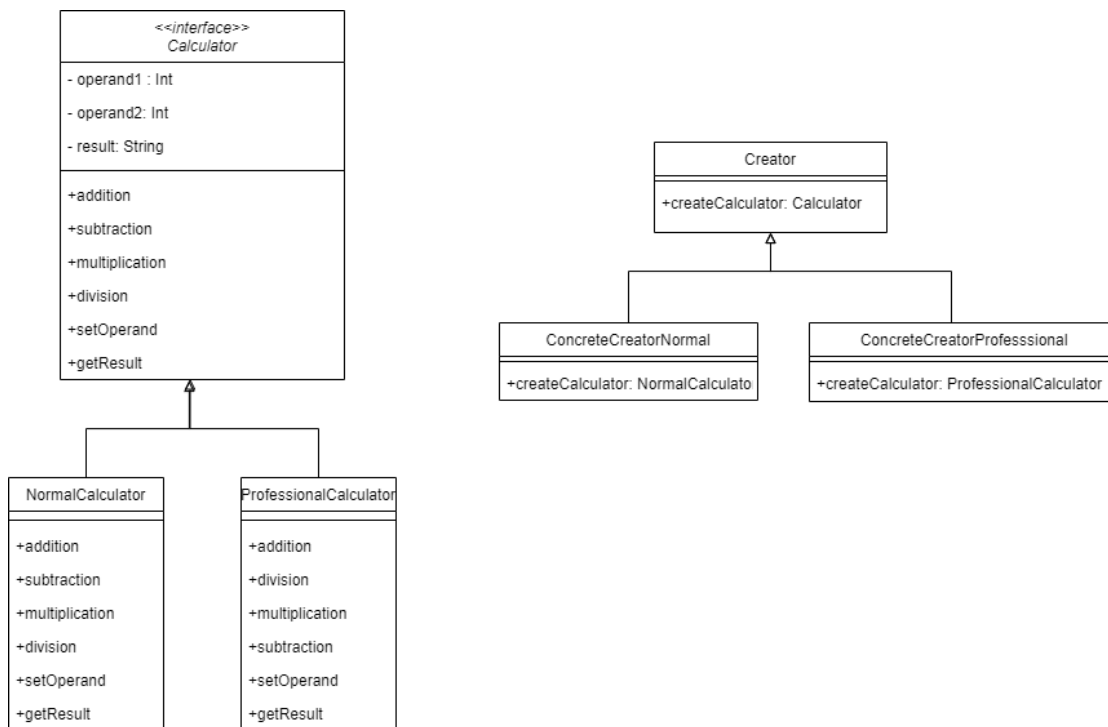
Music Streaming Service

All users of the Music Streaming Service (MSS) must first sign up for the service or sign in using their existing account. Users sign up by providing a username and password and may also provide optional information like name, email address. This process also involves selecting a subscription plan (Free or Premium). Only after signing in, user can search for music by various criteria or select a song or album to start playing music or create custom playlists. All the basic Features come with Free subscription plan while Premium user have the additional benefit of downloading songs or playlists for offline listening. Additionally, MSS also allows independent artists to upload their music to the system. Unlike regular users, artists not only have access to Premium features but can also view statistics on their music streams.

Question 1 (20 marks) Draw Use case diagram by identifying the actors, use cases, and associations for the MSS.
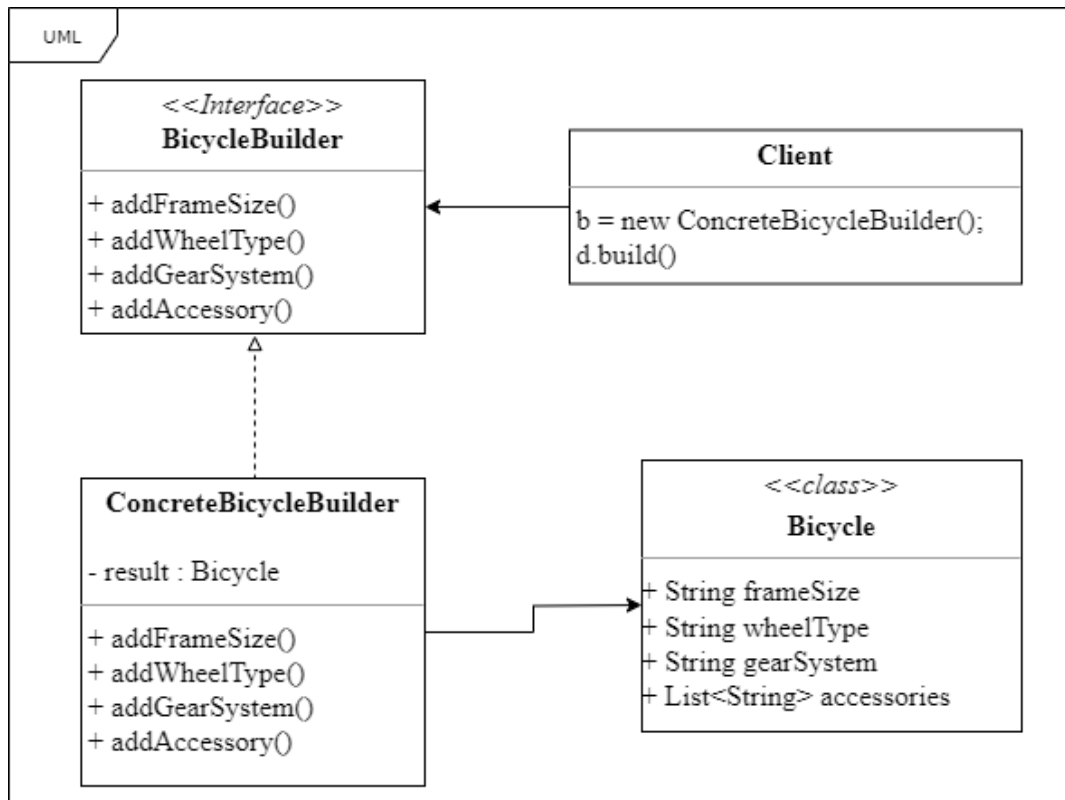
Question 2 (20 marks) Draw Class Diagram for the MSS to show the relationships between its objects, classes, attributes and operations.

Question 3 (20 marks) Implement the Factory Method pattern to streamline the creation process for the following program (Refer to the Class Diagram for more information).

Develop a program that functions as a calculator with 2 separate Mode: Normal calculator and Professional calculator. Normal calculator should be able to perform 2 basic arithmetic operations: addition and subtraction while Professional calculator allow two addition arithmetic operations: multiplication and division. The user should be able to enter two numbers then the program performs the chosen operation on the entered numbers and displays the result. If the chosen arithmetic operation isn't allowed, then your program should return "Operation not support".

<<interface>>
Calculator

- operand1 : Int
- operand2: Int
- result: String

+addition
+subtraction
+multiplication
+division
+setOperand
+getResult

Creator

+createCalculator: Calculator

ConcreteCreatorNormal

+createCalculator: NormalCalculato

ConcreteCreatorProfesssional

+createCalculator: ProfessionalCalculator

NormalCalculator

+addition
+subtraction
+multiplication
+division
+setOperand
+getResult

ProfessionalCalculator

+addition
+division
+multiplication
+subtraction
+setOperand
+getResult

Question 4 (20 marks) Imagine you are developing an e-commerce website that sells custom-made bicycles. Customers can order bicycles by selecting various options, such as frame size, wheel type, gear system, and accessories. To handle the complexity of these customizable bicycles, implement the Builder pattern to streamline the order creation process as below.

UML

<<Interface>>
BicycleBuilder

+ addFrameSize()
+ addWheelType()
+ addGearSystem()
+ addAccessory()

Client

b = new ConcreteBicycleBuilder();
d.build()

ConcreteBicycleBuilder

- result : Bicycle

+ addFrameSize()
+ addWheelType()
+ addGearSystem()
+ addAccessory()

<<class>>
Bicycle

+ String frameSize
+ String wheelType
+ String gearSystem
+ List<String> accessories

Question 5 (20 marks) Look at the code and see if you can identify any code smells or areas that might need improvement. Your task will be to identify the code smell and consider how it might be improved by providing your own implementation.

```java
public class ReportGenerator {
 public void generateReport(String reportType) {
  if (reportType.equals("Sales")) {
   generateSalesReport();
  } else if (reportType.equals("Inventory")) {
   generateInventoryReport();
  } else {
   System.out.println("Error: Invalid report type");
  }
 }
 private void generateSalesReport() {
  // Complex logic to fetch sales data, potentially iterating through large datasets (not shown)
  System.out.println("Sales Report Generated");
 }
 private void generateInventoryReport() {
  // Complex logic to fetch and process inventory data, potentially with calculations (not shown)
  System.out.println("Inventory Report Generated");
 }
}
```

------ **Hết/***End* ------