56 Mahbub
57- Rudro

University of Information Technology & Sciences (UITS)

Faculty of Science and Engineering

Department of Computer Science and Engineering

Program: B.Sc. in CSE

Term Final Examination, Autumn 2025

Course Title: Object-Oriented Programming Language

Course Code: CSE0613121

Marks: 50 **Time: 3*(three)* hours**

(Answer all questions.)

Marks

1. a) You are creating a simple Book Information System. Each book should have a title, an author, and a publication year. The goal is to properly encapsulate these details. **[05]**

   i. Create a class `Book` with:
      - Private fields: `title` (String), `author` (String), and `publicationYear` (int).

   ii. Implement public getter methods for each field.

   iii. Implement public setter methods for each field. (No validation needed here; assume any input from `main` is considered valid for this system).

   iv. In the `main` method:
      - Create two `Book` objects.
      - Set the details for both books using the setter methods.
      - Print the full details (Title, Author, Publication Year) of both books using their respective getter methods.

   b) Analyze why Java does not support multiple inheritance with classes. If two interfaces have a default method with the same name, and a class implements both, how can the class resolve this conflict? Provide an explanation. **[05]**

2. a) The following scenario demonstrates the use of abstract classes: **[04]**

   A company system tracks different types of employees. The base class `Employee` is abstract and defines an abstract method `calculateSalary()`.The subclass `FullTimeEmployee` provides an implementation.

   Here, Create an abstract class `Employee` with `abstract void calculateSalary()`.Create a subclass `FullTimeEmployee` that overrides it and prints `"Full-time employee salary calculated"`.Can you create an object of `Employee`? Why or why not? Analyze it.In `main()`, create a `FullTimeEmployee` object and call the method.

b) **Demonstrate** inheritance in Figure:01.Create an object of `ElectricCar` in the `main()` method and show the information of following methods. **[04]**

- `startEngine()`
- `drive()`
- `charge()`

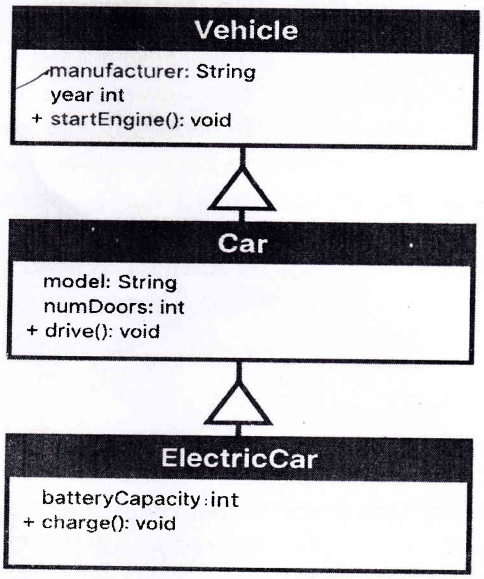

Figure:01

c) Differentiate between method overloading and method overriding in Object Oriented Programming Language. **[02]**

3. a) Consider the following code: **[03]**

```
class Student {
    static int studentCount = 0;
    String name;

    Student(String name) {
        this.name = name;
        studentCount++;
    }

    static void showCount() {
        System.out.println("Total students: " +
studentCount);
    }
}
```

Construct the java code in main method to call the showCount() method and studentCount variable.

b) You are designing an online learning platform. Different types of users have multiple capabilities: **[04]**

- interface "`Learner`"→ method `attendCourse(String courseName)`.

- interface "`Instructor`"→ method `teachCourse(String courseName)`.
- interface "`Admin`"→ default method `accessDashboard()` that prints `"Accessing admin dashboard"`.

Some users can have multiple roles at the same time, e.g., a user can be both a learner and an instructor. Now,

i. Create the interfaces `Learner`, `Instructor`, and `Admin` with the methods described.
ii. Create a class `PlatformUser` that implements all three interfaces.
iii. Override the `accessDashboard()` method to print `"Admin dashboard access granted"`.
iv. In the main method, create a `PlatformUser` object and call:
   - `attendCourse("Java Basics")`
   - `teachCourse("Advanced Java")`
   - `accessDashboard()`

c) State "**this**" keyword in Java and write a program with "**this**" keyword to demonstrate its necessity. **[03]**

4. a) Consider the following Java program that contains different types of exceptions. **[04]**

```
public class ExceptionDemo {

    public static void main(String[] args) {

        int[] numbers = {10, 20, 30};

        String str = null;

        int result = 10 / 0;

System.out.println("Length of string: " + str.length());

        System.out.println("Number: " + numbers[5]);

        System.out.println("Program completed
successfully!");

    }}
```

- Identify the possible exceptions in the above code.
- Modify the program using try-catch-finally blocks (or multiple catch blocks) to handle all exceptions properly.
- Display appropriate meaningful messages for each exception (e.g., `"Cannot divide by zero"`, `"String is null"`, etc.).

b) Explain the difference between the keywords **`throw`** and **`throws`** in Java. **[02]**

c) Suppose, you are designing a banking application. There is a base class `BankAccount` that has a protected method `calculateInterest()` which calculates monthly interest based on the account balance. Only classes that extend `BankAccount` (like `SavingsAccount` or `FixedDepositAccount`) can use **[04]**

or override this method.

Your task is to:

i. Create a base class BankAccount with:
   - A protected method calculateInterest() that prints "Calculating base interest...".
   - A field balance to store account balance.
   - A constructor to initialize the balance.

ii. Create a subclass SavingsAccount that:
   - Overrides calculateInterest() to print "Calculating savings account interest...".
   - Adds a method showInterest() that calls the protected calculateInterest() method.

iii. In the main method:
   - Create an object of SavingsAccount.
   - Call the showInterest() method to display the interest calculation.

5. a) Implement the following classes demonstrating hierarchical inheritance. **[06]**

i. Create a base class **Person** with:
   - Fields: name (String), age (int).
   - Constructor to initialize name and age.
   - A method showPersonInfo() that prints "Name: <name>, Age: <age>".

ii. Create subclasses:
   - Teacher – adds a field subject (String) and a method showTeacherInfo() that calls showPersonInfo() and prints "Subject: <subject>".
   - Student – adds a field grade (String) and a method showStudentInfo() that calls showPersonInfo() and prints "Grade: <grade>".
   - Staff – adds a field department (String) and a method showStaffInfo() that calls showPersonInfo() and prints "Department: <department>".

iii. In the main method:
   - Create objects for Teacher, Student, and Staff.
   - Call their respective info methods to display full details.

Now, explain why this is an example of hierarchical inheritance.

b) Explain the life cycle/states of a thread in Java. **[04]**

public void main ~~year~~ () {

sys . . . . (Publication year : " . . . . }

class main {

Public sta . .

~~b~~ book b1 = ... ( ) ;

System . . . . (Book ;

b1 . set au . . (

" " pubi . .