COMSATS University, Islamabad (CUI)
Department of Computer Sciences
Terminal Examination Spring-2025

Subject: CSC-241 Object Oriented Programming
Semester: BSE-III (A),
Instructor: Haseena Kainat

Dated: 2 July, 2025
Marks: 50
Time: 180 mins

Instructions:
Use of mobile phones and calculators is not allowed.
Attempt all questions on the answer sheet.
Return the question paper along with the answer sheet.

CLO 1: Demonstrate fundamental principles and concepts of object-oriented programming.

Q1. Answer the following question precisely. **[Marks = 2*3]**

i. What makes a class immutable in Java?
ii. How is abstraction implemented in Java?
iii. Explain the difference between an instance method and a static method.

Q2. What will be the output of the following Java code? **[Marks=4]**

```
public class Rectangle {
    private int length;
    private int width;

    public Rectangle() {
        this.length = 5;
        this.width = 10;
    }
    public Rectangle(int length, int width) {
        this.length = length;
        this.width = width;
    }
    public Rectangle(Rectangle obj) {
        this.length = obj.length;
        this.width = obj.width;
    }
    public void setLength(int length) {
        this.length = length;
    }

    public void setWidth(int width) {
        this.width = width;
    }
    public int calculateArea() {
        return length * width;
    }
    public void increaseDimensions(Rectangle obj) {
        this.length += obj.length;
        this.width += obj.width;
    }
    public void display() {
        System.out.println("Length: " + length + ", Width:
" + width);
    }
public static Rectangle[] doubleDimensions(Rectangle[] arr)
{
        Rectangle[] modified = new Rectangle[arr.length];
        for (int i = 0; i < arr.length; i++) {
            modified[i] = new Rectangle(arr[i]);
            modified[i].length *= 2;
            modified[i].width *= 2;
```

```
public class Runner {
    public static void main(String[] args) {
        Rectangle[] rArray = new Rectangle[3];

        rArray[0] = new Rectangle(5, 6);
        rArray[1] = new Rectangle();
        rArray[2] = new Rectangle(rArray[0]);

        rArray[0].setLength(8);
        rArray[1].setWidth(12);

        System.out.println("Original Rectangles:");
        for (int i = 0; i < rArray.length; i++) {
System.out.print("Rectangle " + (i + 1) + ": ");
            rArray[i].display();
        }

        System.out.println("==============");

        rArray[0].increaseDimensions(rArray[1]);
        System.out.println("After increasing r0 with
r1:");
        rArray[0].display();
        rArray[1].display();

        System.out.println("==============");
Rectangle[] doubledArray =
Rectangle.doubleDimensions(rArray);
        System.out.println("Doubled Dimension
Rectangles:");
   for (int i = 0; i < doubledArray.length; i++) {
  System.out.print("Rectangle " + (i + 1) + ": ");
            doubledArray[i].display();
        }
    }
}
```

```
        }
        return modified;
    }
}
```

CLO 2: Apply the concepts of object-oriented programming principles along with interfaces and exception handling to solve a real-world problem.

Q4. Implement the following system using Object Oriented Programming Principles: **[Marks=10]**
A new **vehicle registration system** has been launched by the city transport department. The system handles two types of vehicles: **Private Vehicles** and **Commercial Vehicles**. Each vehicle must be registered with its **owner**, and the registration fee is calculated based on the type of vehicle and the owner's **profession.**

i. Create an **abstract class** Vehicle with common attributes such as `registrationNumber`, `engineCapacity`, `baseFee`, and an `Owner`.The `Owner` class should have attributes: `name`, `age`, and `profession`.
ii. For **Private Vehicles**, if the owner's profession is `"Teacher"`, apply a **5% discount** on the base fee, followed by a **7% tax** on the discounted fee.
iii. For **Commercial Vehicles**, if the owner is `"Driver"`, apply a **3% discount**, followed by a **12% tax.** An interface `Displayable` should be used to declare a method `generateDisplayInfo()` which returns a message showcasing vehicle details and registration benefits.
iv. In the **main/runner class**, instantiate both vehicle types and invoke `calculateRegistrationFee()` and `generateDisplayInfo()` polymorphically.

Q5. You are developing a **Zoo Management System** for a wildlife park. The system needs to manage various types of animals, such as Lions and Elephants etc. All animals share some common behavior, such as making a sound, but each animal also has some **specific abilities**—for example, **Lions can roar and eat meat ,** and **Elephants can spray water** and **eat plants**. To manage these animals uniformly, the system should store them as objects of the general Animal type. However, when needed, it should be able to access their specific behaviors by identifying the type of each animal at runtime. You are required to implement this behavior by **[Marks=10]**

i. *Treating specific animals* as general **Animal** types
ii. *Access common behavior*
iii. *Access specific* types like Lion or Elephant when needed specific behavior.

Q6. Create a class hierarchy: `Item` **(base class)** `Book`, `Laptop`, `Mobile` **(subclasses).** Attributes of `item` class includes `name` and `price`. Class `Book` with attribute `Author_name` and class `Mobile` includes `model_No` . Then, create a generic class `Inventory<T>` that stores a list of items of any type etc. Book, Laptop, mobile. Add methods to: **[Marks=10]**

i. Add items to inventory.
ii. Display all items.
iii. Calculate total value using **`calculateTotalPrice(List<T> item)`**

Demonstrate with creating inventory:

i. A list of `Book`
ii. A list of `Laptop`

CLO 3: Create a program using standard libraries.

Q7. Suppose you are developing a Patient Health Record Management System for a hospital. The goal is to help medical staff efficiently record, manage, and retrieve patient medical records. All records should be stored persistently to allow long-term access and updates. **[Marks=10]**

1. **Add New Patient Record:** Each patient record should include:
   i. Patient ID (String)
   ii. Name (String)
   iii. Age (int)
   iv. Diagnosis (String)
   v. Treatment Plan (String)
   vi. Severity Level (Low, Moderate, Severe, Critical)
2. **Allow the user to input new patient details, and store the record persistently.**
3. **Display Records**: Display a list of all stored patient records in a readable and structured format.
4. **Search Patient by ID**: Allow searching for a patient using their unique Patient ID. if found, show the full patient record; otherwise, show a "not found" message.
5. **Update Diagnosis or Treatment Plan:** Allow the user to update the diagnosis or treatment plan of a patient. The updated information should be saved permanently.

***** GOOD LUCK *****