University of Information Technology & Sciences (UITS)
Faculty of Science and Engineering
Department of Computer Science & Engineering
Program: B.Sc. in CSE
Mid Term Examination, Spring 2024
Course Title: Object Oriented Programming
Course Code: CSE 151/CSE 203/IT 203

Marks: 20 **Time: 1(One)Hour**

(Answer All Questions)

Marks

1. a) Explain why is java interpreted as robust language? Differentiate between the application of Structured Programming Language and Object-Oriented Programming Language. **[03]**

   b) Implement the scenario in ***figure 1*** using the basic OOP concepts in java language. **[Use Constructor]** **[04]**

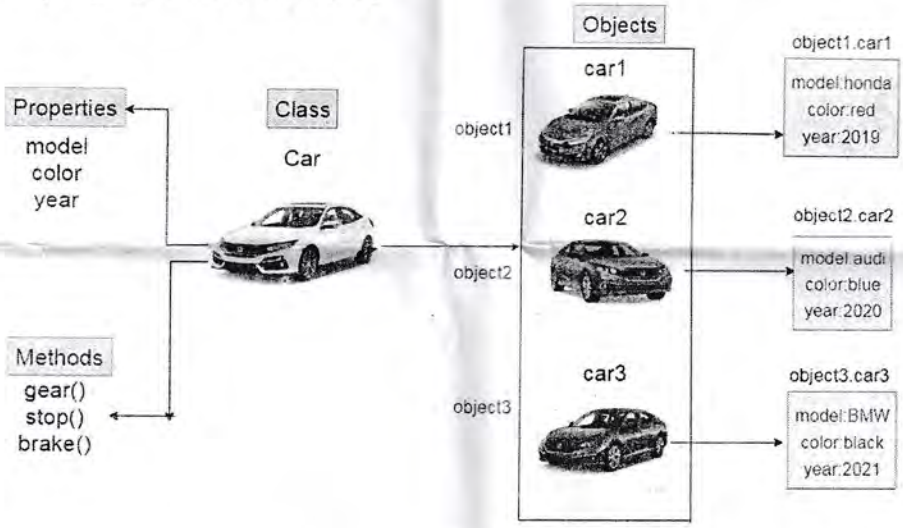

Figure 1: Class, Object and Method Visualization of a Car.

c) Write the output of the given code snippet. **[03]**

```
class trickOfMath {
  public static void main (String [] args )
    {
    int a = 5, b = 9;
    System.out.println(a++);
    System.out.println(--b);
    System.out.println(a==b),
    System.out.println(a*=3);
    System.out.println(a|b);
    System.out.println(a^b);
    }
}
```

2 a) Construct a java program to calculate the factorial of a number by creating a class named 'Factorial' having two methods. First method named as 'setNum' takes the number as parameter and the second method named as 'getfact' returns the factorial of the number. [04]

b) Explain the method overloading with proper example. [03]

c) Convert the followings: [03]

i) Integer 123456 to Short

ii) Integer 999 to Byte

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, Spring 2024

Course Title: Object Oriented Programming Language

Course Code: CSE 151

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

(Answer all questions)

1. a) Evaluate the process of extending the class A of both packages from class B of package 1 from figure 1: **[04]**

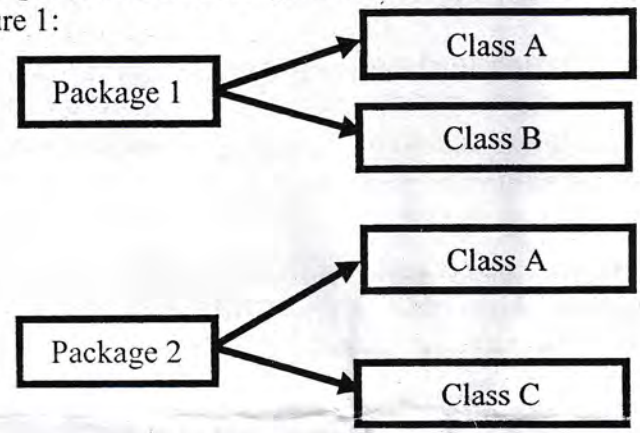

Figure 1: Different classes under different packages.

b) Explain the use of "super" keyword with proper example. **[03]**

c) Complete the table below: **[03]**

Table 1: Table of Access Modifier.

	Private	Default	Protected	Public
Same Class	Yes	Yes	Yes	Yes
Same package subclass				
Same package non-subclass				
Different package subclass				
Different package non-subclass				

2. a) Imagine you are tasked with developing a software system for managing a university. The system needs to manage two types of entities: Person and Building. Person entities include roles like Student and Professor, while Building entities include types like Library and Lecture Hall. **[03]**

Differentiate between when to use a class and when to use an interface in the design of this university management system.

b) Write the JAVA code to implement the scenario below. [04]

Abstract Class: Animal Common Properties: name, age. Abstract Methods: eat(), sleep()	Class: Dog Common Properties: breed. Implemented Methods: eat(), sleep() Method: sound()

Demonstration:

- Create object of the Dog class (child class) which will inherit the Animal (Parent Class).
- Demonstrate calling the inherited methods eat() and sleep() as well as methods specific to each child class (speak(sound) for dogs). As such bulldog sounds like "griff griff", Chihuahua sounds like "Yip Yip", husky sounds like "awoo".

c) Explain the type of exception with proper example. [03]

3. a) Analyze the provided Java code involving the Vehicle and Car classes and the main class AccessDemo. Predict the output of the program when it is executed. Explain the accessibility and visibility of the fields and methods accessed in the main method and within the Car class methods. [04]

```
class Vehicle {
   private String name = "Generic Vehicle";
   String defaultColor = "Black";
   protected int maxSpeed = 120;
   public void displayFeatures() {
      System.out.println("Vehicle features displayed.");
   }
}
class Car extends Vehicle {
   public Car() {
      System.out.println("Car max speed: " + maxSpeed);
   }
   public void showColor() {
      System.out.println("The car color is " + defaultColor);
   }
}
public class AccessDemo {
   public static void main(String[] args) {
      Vehicle myVehicle = new Vehicle();
      Car myCar = new Car();

      myVehicle.displayFeatures();
      myCar.showColor();
      System.out.println("Car Max Speed: " + myCar.maxSpeed);
      System.out.println("Vehicle Color: " + myVehicle.defaultColor);
   }}
```

b) Examine the role of static nested classes in Java by defining their nature, discussing their restrictions and characteristics in relation to non-static outer classes, and explaining their interaction with the instance members of the outer class. [04]

c) Explain the importance of the "@Override" annotation [02]

4. a) Explain the purpose and benefits of using packages in Java programming. [02]

b) Below is a Java program that performs various operations on arrays and strings. Your task is to modify this program by adding appropriate exception handling to prevent it from crashing due to runtime exceptions. Use try-catch blocks to handle possible exceptions that might be thrown by the code. Provide meaningful messages in the catch blocks. [04]

```
public class ExceptionHandlingExercise {
  public static void main(String[] args) {
    int[] array = new int[5];
    System.out.println("Accessing the sixth element: " + array[5]);

    String text = "UITS_CSE";
    char character = text.charAt(10);
    System.out.println("Character at position 10: " + character);

    int result = 100 / 0;
    System.out.println("Result of division: " + result);
  }
}
```

c) Consider the following table that defines certain methods which are part of a Plant interface in Java: [04]

Table 2: Properties of Plant Interface.

Method Name	Return Type	Parameters	Description
grow	void	none	Increases the height of the plant.
absorbWater	void	int amount	Absorbs a specified amount of water in ml.
produceOxygen	int	none	Returns the amount of oxygen produced in ml.
displayHealth	String	none	Returns a description of the plant's health.

Create a Java class Tree that implements the Plant interface. Provide implementations for each method where grow increases the height, absorbWater manages water intake, produceOxygen calculates and returns oxygen production, and displayHealth provides a status report on the tree's health.

5. a) Explain the process of passing objects as parameters to methods in Java. Specifically, address whether Java uses pass-by-value or pass-by-reference for objects, and the implications this has for object manipulation within methods. [03]

b) Using the provided code, explain how polymorphism is applied to handle different employee roles within the hospital management system. [03]

```
abstract class Employee {
    abstract void dailyRoutine();
}

class Doctor extends Employee {
    void dailyRoutine() {
        System.out.println("Doctor: Consult patients.");
    }
}

class Nurse extends Employee {
    void dailyRoutine() {
        System.out.println("Nurse: Administer care and medications.");
    }
}

class Administrator extends Employee {
    void dailyRoutine() {
        System.out.println("Administrator: Manage hospital operations.");
    }
}

public class HospitalManagement {
    public static void main(String[] args) {
        Employee[] employees = {
            new Doctor(),
            new Nurse(),
            new Administrator()
        };

        for (Employee emp : employees) {
            emp.dailyRoutine();
        }
    }
}
```

c) **Analyze** the Book class in the provided Java code snippet, explaining the purpose of overriding the toString() and equals() methods, and predict the output of the code when executed. **[04]**

```
class Book {
    private String title;
    private String author;

    Book(String title, String author) {
        this.title = title;
        this.author = author;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj) return true;
        if (obj == null || getClass() != obj.getClass()) return false;
        Book book = (Book) obj;
        return title.equals(book.title) && author.equals(book.author);
    }

    @Override
    public String toString() {
        return "Book{" +
                "title='" + title + '\'' +
                ", author='" + author + '\'' +
                '}';
    }
}

public class Library {
    public static void main(String[] args) {
        Book book1 = new Book("1984", "George Orwell");
        Book book2 = new Book("1984", "George Orwell");
        System.out.println(book1);
        System.out.println("Book1 equals Book2: " + book1.equals(book2));
    }
}
```