Assignment 3

Aim: Implementation of a given problem statement/s for different types of inheritance.

Que.1) Write a Java program to create a class known as "BankAccount" with methods called deposit() and withdraw(). Create a subclass called SavingsAccount that overrides the withdraw() method to prevent withdrawals if the account balance falls below one hundred

→

```
class BankAccount {
    protected double balance;

    public BankAccount(double initialBalance) {
        this.balance = initialBalance;
    }

    public void deposit(double amount) {
        if (amount > 0) {
            balance += amount;
            System.out.println("Deposited: " + amount + " | New Balance: " + balance);
        } else {
            System.out.println("Invalid deposit amount.");
        }
    }

    public void withdraw(double amount) {
        if (amount > 0 && balance >= amount) {
            balance -= amount;
            System.out.println("Withdrawn: " + amount + " | New Balance: " + balance);
        } else {
            System.out.println("Insufficient funds or invalid amount.");
        }
```

```
    }

    public double getBalance() {
        return balance;
    }
}

class SavingsAccount extends BankAccount {
    public SavingsAccount(double initialBalance) {
        super(initialBalance);
    }

    @Override
    public void withdraw(double amount) {
        if (balance - amount >= 100) {
            super.withdraw(amount);
        } else {
            System.out.println("Withdrawal denied. Minimum balance of 100 must be
maintained.");
        }
    }
}

public class BankApp {
    public static void main(String[] args) {
        SavingsAccount myAccount = new SavingsAccount(500);
        myAccount.deposit(200);
        myAccount.withdraw(550); // Should be denied
        myAccount.withdraw(100); // Should be successful
    }
```

}

Que.2)W rite a Java program that creates a class hierarchy for employees of a company. The base class should be Employee, with subclasses Manager, Developer, and Programmer. Each subclass should have properties such as name, address, salary, and job title. Implement methods for calculating bonuses, generating performance reports, and managing projects.

→

```
abstract class Employee {
    protected String name;
    protected String address;
    protected double salary;
    protected String jobTitle;

    public Employee(String name, String address, double salary, String jobTitle) {
        this.name = name;
        this.address = address;
        this.salary = salary;
        this.jobTitle = jobTitle;
    }

    public abstract double calculateBonus();
    public abstract String generatePerformanceReport();

    public void displayInfo() {
        System.out.println("Name: " + name);
        System.out.println("Address: " + address);
        System.out.println("Salary: " + salary);
        System.out.println("Job Title: " + jobTitle);
    }
}
```

```
class Manager extends Employee {
    public Manager(String name, String address, double salary) {
        super(name, address, salary, "Manager");
    }

    @Override
    public double calculateBonus() {
        return salary * 0.15;
    }

    @Override
    public String generatePerformanceReport() {
        return "Manager " + name + " has successfully led the team and improved efficiency.";
    }

    public void manageProjects() {
        System.out.println(name + " is managing multiple company projects.");
    }
}

class Developer extends Employee {
    public Developer(String name, String address, double salary) {
        super(name, address, salary, "Developer");
    }

    @Override
    public double calculateBonus() {
        return salary * 0.10;
    }
```

```
    @Override
    public String generatePerformanceReport() {
        return "Developer " + name + " has delivered high-quality software solutions.";
    }
}

class Programmer extends Employee {
    public Programmer(String name, String address, double salary) {
        super(name, address, salary, "Programmer");
    }

    @Override
    public double calculateBonus() {
        return salary * 0.08;
    }

    @Override
    public String generatePerformanceReport() {
        return "Programmer " + name + " has written efficient and optimized code.";
    }
}

public class Company {
    public static void main(String[] args) {
        Manager manager = new Manager("Alice", "123 Main St", 80000);
        Developer developer = new Developer("Bob", "456 Elm St", 60000);
        Programmer programmer = new Programmer("Charlie", "789 Oak St", 50000);
```

```
        manager.displayInfo();
        System.out.println("Bonus: " + manager.calculateBonus());
        System.out.println(manager.generatePerformanceReport());
        manager.manageProjects();

        System.out.println();

        developer.displayInfo();
        System.out.println("Bonus: " + developer.calculateBonus());
        System.out.println(developer.generatePerformanceReport());

        System.out.println();

        programmer.displayInfo();
        System.out.println("Bonus: " + programmer.calculateBonus());
        System.out.println(programmer.generatePerformanceReport());
    }
}
```

Que.3) Implement Following: a. Create abstract class shapes with dim1, dim2 variables and abstract area() method. Class b. rectangle and triangle inherits shape class. Calculate area of rectangle and triangle.

→

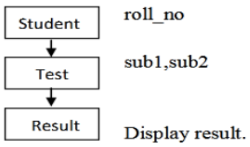
```
abstract class Shape {
    protected double dim1, dim2;

    public Shape(double dim1, double dim2) {
        this.dim1 = dim1;
        this.dim2 = dim2;
    }
```

```
    public abstract double area();
}

class Rectangle extends Shape {
    public Rectangle(double length, double width) {
        super(length, width);
    }

    @Override
    public double area() {
        return dim1 * dim2;
    }
}

class Triangle extends Shape {
    public Triangle(double base, double height) {
        super(base, height);
    }

    @Override
    public double area() {
        return 0.5 * dim1 * dim2;
    }
}

public class ShapeTest {
    public static void main(String[] args) {
        Rectangle rectangle = new Rectangle(10, 5);
```

```
        Triangle triangle = new Triangle(8, 4);

        System.out.println("Area of Rectangle: " + rectangle.area());
        System.out.println("Area of Triangle: " + triangle.area());
    }
}
```

Que.4)

4. Write a program to perform Multilevel Inheritance




```
class Student {
    int roll_no;

    public Student(int roll_no) {
        this.roll_no = roll_no;
    }

    public void displayRollNo() {
        System.out.println("Roll Number: " + roll_no);
    }
}

class Test extends Student {
    int sub1, sub2;
```

```
    public Test(int roll_no, int sub1, int sub2) {
        super(roll_no);
        this.sub1 = sub1;
        this.sub2 = sub2;
    }

    public void displayMarks() {
        System.out.println("Subject 1 Marks: " + sub1);
        System.out.println("Subject 2 Marks: " + sub2);
    }
}

class Result extends Test {
    int total;

    public Result(int roll_no, int sub1, int sub2) {
        super(roll_no, sub1, sub2);
        this.total = sub1 + sub2;
    }

    public void displayResult() {
        displayRollNo();
        displayMarks();
        System.out.println("Total Marks: " + total);
    }
}

public class MultilevelInheritance {
```

```
    public static void main(String[] args) {
        Result student1 = new Result(101, 85, 90);
        student1.displayResult();
    }
}
```