Test 7

SET 1

1. Create a class Employee with properties name and salary, and a constructor to initialize these properties. Create a subclass Manager that adds a property department and a constructor to initialize all properties. Demonstrate creating instances of both classes.

```java
class Employee {
    String name;
    double salary;
    public Employee(String name, double salary) {
        this.name = name;
        this.salary = salary;
    }
    public void display() {
        System.out.println("Name: " + name);
        System.out.println("Salary: " + salary);
    }
}
class Manager extends Employee {
    String department;
    public Manager(String name, double salary, String department) {
        super(name, salary);
        this.department = department;
    }

    public void display() {
        super.display();
        System.out.println("Department: " + department);
    }
}

public class Main {
    public static void main(String[] args) {
        Employee emp = new Employee("akhil", 50000);
        System.out.println("Employee Details:");
        emp.display();

        Manager mgr = new Manager("ruthin", 456788, "AI&DS");
        System.out.println("\nManager Details:");
        mgr.display();
    }
}
```

```
java -cp /tmp/Z1SzXdIrPf/Main
Employee Details:
Name: akhil
Salary: 50000.0

Manager Details:
Name: ruthin
Salary: 456788.0
Department: AI&DS

=== Code Execution Successful ===
```

2. Create a superclass Person with properties name and age, and a method displayInfo(). Create a subclass Student that adds a property studentId and overrides the displayInfo() method. Use the super keyword to call the superclass method.

```java
class Person {
    String name;
    int age;
    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }
    public void displayInfo() {
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
    }
}
class Student extends Person {
    String studentId;
    public Student(String name, int age, String studentId) {
        super(name, age);
        this.studentId = studentId;
    }
    public void displayInfo() {
        super.displayInfo();
        System.out.println("Student ID: " + studentId);
    }
}
public class Main {
    public static void main(String[] args) {
        Person person = new Person("AKHIL", 20);
        System.out.println("Person Details:");
        person.displayInfo();
        Student student = new Student("RUTHIN", 19, "192324112");
        System.out.println("\nStudent Details:");
        student.displayInfo();
    }
}
```

```
java -cp /tmp/o0qh69CYLG/Main
Person Details:
Name: AKHIL
Age: 20

Student Details:
Name: RUTHIN
Age: 19
Student ID: 192324112

=== Code Execution Successful ===
```

3. Create a class Vehicle with a method move(). Create subclasses Car and Bicycle, each

overriding the move() method to provide specific implementations. Demonstrate the use of overridden methods.

```java
class Vehicle {
    public void move() {
        System.out.println("The vehicle is moving.");
    }
}
class Car extends Vehicle {
    public void move() {
        System.out.println("The car is moving");
    }
}
class Bicycle extends Vehicle {
    public void move() {
        System.out.println("The bicycle is moving");
    }
}
public class Main {
    public static void main(String[] args) {
        Vehicle vehicle = new Vehicle();
        vehicle.move();
        Car car = new Car();
        car.move();
        Bicycle bicycle = new Bicycle();
        bicycle.move();
    }
}
```

```
java -cp /tmp/ufNYO3gsaU/Main
The vehicle is moving.
The car is moving
The bicycle is moving

=== Code Execution Successful ===
```

4. Design a class hierarchy for shapes in Java. Include an abstract class Shape with methods calculateArea() and calculatePerimeter(). Implement subclasses such as Circle, Rectangle, and Triangle that extend Shape and provide specific implementations for area and perimeter calculations.

```java
abstract class Shape {
    abstract double calculateArea();
    abstract double calculatePerimeter();
}
class Circle extends Shape {
    double radius;
    public Circle(double radius) {
        this.radius = radius;
    }
    double calculateArea() {
        return Math.PI * radius * radius;
    }
    double calculatePerimeter() {
        return 2 * Math.PI * radius;
    }
}
class Rectangle extends Shape {
    double length, width;
    public Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }
    double calculateArea() {
        return length * width;
        }
    double calculatePerimeter() {
        return 2 * (length + width);
    }
}
class Triangle extends Shape {
    double b,h;
    public Triangle(double b,double h) {
        this.b=b;
        this.h=h;
        }
    double calculateArea() {
        return 0.5*b*h;
    }
    double calculatePerimeter() {
        return b+h;
    }
}
public class Main {
    public static void main(String[] args) {
        Shape circle = new Circle(5);
        Shape rectangle = new Rectangle(4, 6);
        Shape triangle = new Triangle(3, 4);
        System.out.println("Circle Area: " + circle.calculateArea());
        System.out.println("Circle Perimeter: " + circle.calculatePerimeter());
        System.out.println("Rectangle Area: " + rectangle.calculateArea());
        System.out.println("Rectangle Perimeter: " + rectangle.calculatePerimeter());
        System.out.println("Triangle Area: " + triangle.calculateArea());
        System.out.println("Triangle Perimeter: " + triangle.calculatePerimeter());
    }
```

```
java -cp /tmp/NHxjOwmTY3/Main
Circle Area: 78.53981633974483
Circle Perimeter: 31.41592653589793
Rectangle Area: 24.0
Rectangle Perimeter: 20.0
Triangle Area: 6.0
Triangle Perimeter: 7.0

=== Code Execution Successful ===
```