Week 8 - S8 - Core OOP - Abstract Class And Interface - Assignment Problem(HW)

Name: Ramesh Harisabapathi Chettiar

Date of Submission:07/10/25

QNO1→

Create an abstract class Shape with abstract methods area() and perimeter(). Provide a concrete method displayInfo().

Create subclasses Circle and Rectangle that implement the abstract methods. Test the implementation by creating objects and displaying results.

Hints:

- Use abstract keyword for Shape class.
- Implement area() and perimeter() in subclasses.
- Call displayInfo() from subclass objects.

Shape.java

Shape.java > Shape

```java
public abstract class Shape {
    public abstract double area();
    public abstract double perimeter();

    public void displayInfo() {
        System.out.println("This is a shape.");
    }
}
```

Rectangle.java

```
Rectangle.java > Rectangle
public class Rectangle extends Shape {
    private double width;
    private double height;

    public Rectangle(double width, double height) {
        this.width = width;
        this.height = height;
    }

    @Override
    public double area() {
        return width * height;
    }

    @Override
    public double perimeter() {
        return 2 * (width + height);
    }
}
```

Circle.java

```
Circle.java > Circle
public class Circle extends Shape {
    private double radius;

    public Circle(double radius) {
        this.radius = radius;
    }

    @Override
    public double area() {
        return Math.PI * radius * radius;
    }

    @Override
    public double perimeter() {
        return 2 * Math.PI * radius;
    }
}
```

Main.java

```
Main.java > Main
public class Main {
    Run main | Debug main
    public static void main(String[] args) {
        // Create a Circle object
        Circle circle = new Circle(5.0);
        System.out.println("--- Circle Information ---");
        circle.displayInfo();
        System.out.println("Area: " + circle.area());
        System.out.println("Perimeter: " + circle.perimeter());
        System.out.println();

        // Create a Rectangle object
        Rectangle rectangle = new Rectangle(4.0, 6.0);
        System.out.println("--- Rectangle Information ---");
        rectangle.displayInfo();
        System.out.println("Area: " + rectangle.area());
        System.out.println("Perimeter: " + rectangle.perimeter());
    }
}
```

OUTPUT→

```
PS C:\Users\Ramesh\Personal Folders\MISCELLANEOUS\ENTRANCE EXAMS\SRM\SEMESTERS\SEMESTER-3\JAVA-STEP\Weeks\Week 8\Ass
ignment HW\Program1> cd "c:\Users\Ramesh\Personal Folders\MISCELLANEOUS\ENTRANCE EXAMS\SRM\SEMESTERS\SEMESTER-3\JAVA
-STEP\Weeks\Week 8\Assignment HW\Program1\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
--- Circle Information ---
This is a shape.
Area: 78.53981633974483
Perimeter: 31.41592653589793

--- Rectangle Information ---
This is a shape.
Area: 24.0
Perimeter: 20.0
```

QNO2→

Create an interface Playable with methods play() and pause().

Create two classes MusicPlayer and VideoPlayer that implement this interface.

Demonstrate polymorphism by storing objects in a Playable reference and invoking methods.

Hints:

- Use interface keyword.
- Implement both methods in each class.
- Use Playable ref = new MusicPlayer(); to test polymorphism.

Playable.java

```
Playable.java
public interface Playable {
    void play();
    void pause();
}
```

MusicPlayer.java

```
MusicPlayer.java > MusicPlayer
public class MusicPlayer implements Playable {
    @Override
    public void play() {
        System.out.println("Music is now playing.");
    }

    @Override
    public void pause() {
        System.out.println("Music is paused.");
    }
}
```

VideoPlayer.java

```java
VideoPlayer.java > VideoPlayer
public class VideoPlayer implements Playable {
    @Override
    public void play() {
        System.out.println("Video is now playing.");
    }

    @Override
    public void pause() {
        System.out.println("Video is paused.");
    }
}
```

Main.java

```java
Main.java > Main
public class Main {

    public static void main(String[] args) {
        // Demonstrate polymorphism with a Playable reference for a MusicPlayer object
        Playable player1 = new MusicPlayer();
        System.out.println("--- Using MusicPlayer ---");
        player1.play();
        player1.pause();
        System.out.println(); // Newline for readability

        // Reuse the same Playable reference for a VideoPlayer object
        Playable player2 = new VideoPlayer();
        System.out.println("--- Using VideoPlayer ---");
        player2.play();
        player2.pause();
    }
}
```

OUTPUT→

```
PS C:\Users\Ramesh\Personal Folders\MISCELLANEOUS\ENTRANCE EXAMS\SRM\SEMESTERS\SEMESTER-3\JAVA-STEP\Weeks\Week 8\Ass
ignment HW\Program2> cd "c:\Users\Ramesh\Personal Folders\MISCELLANEOUS\ENTRANCE EXAMS\SRM\SEMESTERS\SEMESTER-3\JAVA
-STEP\Weeks\Week 8\Assignment HW\Program2\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
--- Using MusicPlayer ---
Music is now playing.
Music is paused.

--- Using VideoPlayer ---
Video is now playing.
Video is paused.
```

QNO3→

Create an abstract class Vehicle with abstract method start() and a concrete method stop().

Create an interface Fuel with method refuel().

Create class Car that extends Vehicle and implements Fuel. Test all methods.

Hints:

- Use abstract class for Vehicle.
- Implement refuel() from Fuel interface in Car.
- Show method calls of start(), stop(), and refuel().

Fuel.java

```java
public interface Fuel {
    // Interface method for refueling.
    void refuel();
}
```

Vehicle.java

```java
public abstract class Vehicle {
    // Abstract method to be implemented by subclasses.
    public abstract void start();

    // Concrete method with a default implementation.
    public void stop() {
        System.out.println("The vehicle has stopped.");
    }
}
```

Car.java

```java
Car.java > Car
public class Car extends Vehicle implements Fuel {
    @Override
    public void start() {
        System.out.println("The car's engine has started.");
    }

    @Override
    public void refuel() {
        System.out.println("The car is being refueled.");
    }
}
```

Main.java

```java
Main.java > Main
public class Main {

    public static void main(String[] args) {
        // Create an instance of the Car class.
        Car myCar = new Car();

        // Call the abstract method inherited from Vehicle.
        myCar.start();

        // Call the concrete method inherited from Vehicle.
        myCar.stop();

        // Call the method implemented from the Fuel interface.
        myCar.refuel();
    }
}
```

OUTPUT→

```
PS C:\Users\Ramesh\Personal Folders\MISCELLANEOUS\ENTRANCE EXAMS\SRM\SEMESTERS\SEMESTER-3\JAVA-STEP\Weeks\Week 8\Ass
ignment HW\Program3> cd "c:\Users\Ramesh\Personal Folders\MISCELLANEOUS\ENTRANCE EXAMS\SRM\SEMESTERS\SEMESTER-3\JAVA
-STEP\Weeks\Week 8\Assignment HW\Program3\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
The car's engine has started.
The vehicle has stopped.
The car is being refueled.
```

QNO4→

Create an interface Animal with method eat().

Create another interface Pet that extends Animal and adds method play().

Create a class Dog that implements Pet. Demonstrate interface inheritance in action.

Hints:

- Use interface Pet extends Animal.
- Dog must implement both eat() and play().
- Create object of Dog and test.

Animal.java

```java
public interface Animal {
    void eat();
}
```

Pet.java

```java
public interface Pet extends Animal {
    void play();
}
```

Dog.java

```java
public class Dog implements Pet {
    @Override
    public void eat() {
        System.out.println("The dog is eating its food.");
    }

    @Override
    public void play() {
        System.out.println("The dog is playing with a ball.");
    }
}
```

Main.java

```
Main.java > Main
public class Main {

    public static void main(String[] args) {
        // Create an instance of the Dog class.
        Dog myDog = new Dog();

        // The Dog class can use methods from both the Animal and Pet interfaces.
        myDog.eat();
        myDog.play();
    }
}
```

OUTPUT→

```
PS C:\Users\Ramesh\Personal Folders\MISCELLANEOUS\ENTRANCE EXAMS\SRM\SEMESTERS\SEMESTER-3\JAVA-STEP\Weeks\Week 8\Ass
ignment HW\Program4> cd "c:\Users\Ramesh\Personal Folders\MISCELLANEOUS\ENTRANCE EXAMS\SRM\SEMESTERS\SEMESTER-3\JAVA
-STEP\Weeks\Week 8\Assignment HW\Program4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
The dog is eating its food.
The dog is playing with a ball.
```