Assignment - 8

Object-Oriented Programming in Java

Name: Kamithkar Vinod
Course: PG DAC AUGUST 2025
Form No: 250500480
Date: 19-09-2025

Problem 1: Abstract Class

Task: Create an abstract class Shape with an abstract method double area(). Then, create two subclasses, Circle and Rectangle, that extend Shape and provide implementations for the area method. Write a main method to create instances of Circle and Rectangle, and display their areas.

Code:

```
// abstract class
package shape;

public abstract class Shape {
        public abstract String area();
}

// Rectangle class extends Shape

package shape;

public class Rectangle extends Shape {
        private double length;
        private double breadth;



        public Rectangle(double length, double breadth) {
                this.length = length;
                this.breadth = breadth;
        }

        @Override
        public String area() {
                double a = length * breadth;
```

```
                return "Area of Rectangle: " + a;
        }
}

// Circle extends Shape

package shape;

public class Circle extends Shape {
        private double radius;
        private static double pi = 3.14;

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

        @Override
        public String area() {
                double c = pi * radius * radius;
                return "Area of Circle: " + c;
        }
}

// Main

package shape;

public class Main {

        public static void main(String[] args) {
                Shape rec = new Rectangle(10, 20);

                System.out.println(rec.area());

                Shape cir = new Circle(14);

                System.out.println(cir.area());

        }

}
```

Output: —

```
Area of Rectangle: 200.0
Area of Circle: 615.44
```

Problem 2: Abstract Class

Task: Create an abstract class Animal with an abstract method void sound(). Then, create three subclasses, Dog, Cat, and Cow, each implementing the sound method with their respective sounds. Write a main method to create instances of Dog, Cat, and Cow, and invoke the sound method on each instance.

Code:

```java
// abstract class

package animal;

public abstract class Animal {
        public abstract void sound();
}

// dog class extends animal class
package animal;

public class Dog extends Animal {

        @Override
        public void sound() {
                System.out.println("Bow Bow");
        }
}

// cat class extends animal class
package animal;

public class Cat extends Animal {

        @Override
        public void sound() {
                System.out.println("Meow Meow");
        }
}

// cow class extends animal class
package animal;

public class Cow extends Animal {

```

```
        @Override
        public void sound() {
                System.out.println("Amba Amba");
        }
}

// Main class
package animal;

public class Main {

        public static void main(String[] args) {
                Animal dog = new Dog();
                Animal cat = new Cat();
                Animal cow = new Cow();

                dog.sound();
                cat.sound();
                cow.sound();
        }
}
```

Output: —

```
Bow Bow
Meow Meow
Amba Amba
```

Problem 3: Abstract Class

Task: Create an abstract class Appliance with fields for brand and power consumption, and an abstract method void turnOn(). Create three subclasses, WashingMachine, Refrigerator, and Microwave, each providing their own implementation of the turnOn method. Write a main method to create instances of WashingMachine, Refrigerator, and Microwave, and invoke the turnOn method on each instance to display brand and power consumed.

Code: —

```
package appliance;

public abstract class Appliance {

```

```
5       public abstract void turnON(String brand, double
            powerConsumption);
6
7   }
8
9   /////
10
11  package appliance;
12
13  public class WashingMachine extends Appliance {
14
15      @Override
16      public void turnON(String brand, double powerConsumption) {
17          System.out.println("Brand Name: " + brand);
18          System.out.println("Power Consumption: " +
                powerConsumption+ " units");
19      }
20  }
21
22  /////
23
24  package appliance;
25
26  public class Refrigerator extends Appliance {
27
28      @Override
29      public void turnON(String brand, double powerConsumption) {
30          System.out.println("Brand Name: " + brand);
31          System.out.println("Power Consumption: " +
                powerConsumption + " units");
32      }
33  }
34
35  //////
36
37  package appliance;
38
39  public class Microwave extends Appliance {
40
41      @Override
42      public void turnON(String brand, double powerConsumption) {
43          System.out.println("Brand Name: " + brand);
44          System.out.println("Power Consumption: " +
                powerConsumption+ " units");
45      }
46  }
47
48  /////
49
50  package appliance;
51
```

```java
public class Main {

    public static void main(String[] args) {
        Appliance wm = new WashingMachine();
        Appliance r = new Refrigerator();
        Appliance mw = new Microwave();

        System.out.println("Washing Machine");
        wm.turnON("Whirlpool", 143);

        System.out.println("Refrigerator");
        r.turnON("Samsung", 200);

        System.out.println("Microwave");
        mw.turnON("Agaro", 244);
    }
}
```

Output: —

```
Washing Machine
Brand Name: Whirlpool
Power Consumption: 143.0 units
Refrigerator
Brand Name: Samsung
Power Consumption: 200.0 units
Microwave
Brand Name: Agaro
Power Consumption: 244.0 units
```

Problem 4: Interface

Task: Create an interface Animal with methods makeSound() and eat(). Implement this interface in two classes Dog and Cat.

Code: —

```java
package animal;

public interface Animal {
    void makesound();
    void eat();
}

////

package animal;

public class Dog implements Animal {
```

```
13
14      @Override
15      public void makesound() {
16          System.out.println("Dog sounds Bow Bow");
17      }
18
19      @Override
20      public void eat() {
21          System.out.println("Dog eats bones");
22      }
23  }
24
25  ////
26
27  package animal;
28
29  public class Cat implements Animal {
30
31      @Override
32      public void makesound() {
33          System.out.println("Cat sounds Meow Meow");
34      }
35
36      @Override
37      public void eat() {
38          System.out.println("Cat eats Milk");
39      }
40  }
41
42  ////
43
44  package animal;
45
46  public class Main {
47
48      public static void main(String[] args) {
49          Animal dog = new Dog();
50          Animal cat = new Cat();
51
52          dog.eat();
53          cat.eat();
54
55          System.out.println("----------------");
56
57          dog.makesound();
58          cat.makesound();
59      }
60  }
```

Output: —

```
Dog eats bones
Cat eats Milk
---------------
Dog sounds Bow Bow
Cat sounds Meow Meow
```

Problem 5: Interface

Task: Create an interface Vehicle with a default method startEngine() that prints "Engine started". Implement this interface in the class Car and override the startEngine() method.

Code:

```java
package vehicle;

public interface Vehicle {
        default void startEngine () {
                System.out.println("Engine Started.");
        }
}

/////

package vehicle;

public class Car implements Vehicle {
        @Override
        public void startEngine() {
                Vehicle.super.startEngine();
        }

        public static void main(String[] args) {
                Vehicle v = new Car();
                v.startEngine();
        }
}
```

Output:

```
Engine Started.
```

Problem 6: Interface

Task: Interface Inheritance - Create an interface Shape with methods draw() and calculateArea(). Create another interface Colorful that extends Shape and adds a method fillColor(). Implement these interfaces in the class Circle.

Code:

```
package inheritanceInterface;

public interface Shape {
        void draw();
        double calculateArea();
}

//////

package inheritanceInterface;

public interface Colorful extends Shape {

        void fillColor(String color);
}

//////

package inheritanceInterface;

public class Circle implements Colorful {

        private double radius;

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

        @Override
        public void draw() {
                System.out.println("Draw the circle with radius =
                    " + radius);
        }

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

        @Override
        public void fillColor (String color) {
                System.out.println("Fill the circle with color "
                   + color);
        }
```

```
43
44          public static void main(String[] args) {
45                  Circle c = new Circle(14);
46
47                  System.out.println("Access with Circle reference"
                        );
48                  c.fillColor("Blue");
49                  System.out.println("Area of Circle: " + c.
                        calculateArea());
50                  c.draw();
51
52                  System.out.println("Access with Colorful
                        reference");
53
54                  Colorful cl = new Circle(14);
55
56                  System.out.println("Area of Circle: " + cl.
                        calculateArea());
57                  cl.draw();
58                  cl.fillColor("Yellow");
59
60                  System.out.println("Access with Shape reference")
                        ;
61                  Shape s = new Circle(20);
62
63                  System.out.println("Area of Circle: " + s.
                        calculateArea());
64                  s.draw();
65          }
66
67  }
```

Output: —

```
Access with Circle reference
Fill the circle with color Blue
Area of Circle: 615.7521601035994
Draw the circle with radius = 14.0
Access with Colorful reference
Area of Circle: 615.7521601035994
Draw the circle with radius = 14.0
Fill the circle with color Yellow
Access with Shape reference
Area of Circle: 1256.6370614359173
Draw the circle with radius = 20.0
```

Problem 7: Interface

Task: Interface with Multiple Implementations - Create an interface Payment with a method pay(). Implement this interface in two classes CreditCardPayment and Paypal-

Payment. Write a PaymentProcessor class that uses the Payment interface to process payments.

Code:

```java
// Payment.java
interface Payment {
    void pay(double amount);
}

// CreditCardPayment.java
class CreditCardPayment implements Payment {
    private String cardNumber;

    public CreditCardPayment(String cardNumber) {
        this.cardNumber = cardNumber;
    }

    @Override
    public void pay(double amount) {
        System.out.println("Paid " + amount + " using Credit Card
            : " + cardNumber);
    }
}

// PaypalPayment.java
class PaypalPayment implements Payment {
    private String email;

    public PaypalPayment(String email) {
        this.email = email;
    }

    @Override
    public void pay(double amount) {
        System.out.println("Paid " + amount + " using PayPal
            account: " + email);
    }
}

// PaymentProcessor.java
class PaymentProcessor {
    private Payment payment;

    // Inject dependency (any Payment implementation)
    public PaymentProcessor(Payment payment) {
        this.payment = payment;
    }

    public void processPayment(double amount) {
        payment.pay(amount);
    }
```

```java
}

// Main.java
public class Main {
    public static void main(String[] args) {
        // Credit Card Payment
        Payment creditCardPayment = new CreditCardPayment("1234-5678-9012-3456");
        PaymentProcessor processor1 = new PaymentProcessor(creditCardPayment);
        processor1.processPayment(1500.00);

        // PayPal Payment
        Payment paypalPayment = new PaypalPayment("user@example.com");
        PaymentProcessor processor2 = new PaymentProcessor(paypalPayment);
        processor2.processPayment(750.50);
    }
}
```

Output: —

```
Paid 1500.0 using Credit Card: 1234-5678-9012-3456
Paid 750.5 using PayPal account: user@example.com
```

Problem 8: Anonymous Inner Class

Task: Anonymous Inner Class Implementing an Interface - Create an interface Greeting with a method sayHello(). Write a method generateGreeting() in another class that uses an anonymous inner class to implement the Greeting interface and prints a greeting message

Code: —

```java
// Greeting.java
interface Greeting {
    void sayHello();
}

// GreetingGenerator.java
class GreetingGenerator {
    public void generateGreeting() {
        // Anonymous Inner Class implementing Greeting
        Greeting greeting = new Greeting() {
            @Override
            public void sayHello() {
                System.out.println("Hello! Welcome to the world of Anonymous Inner Classes in Java.");
```

```
            }
        };

        // Call the method
        greeting.sayHello();
    }
}

// Main.java
public class Main {
    public static void main(String[] args) {
        GreetingGenerator generator = new GreetingGenerator();
        generator.generateGreeting();
    }
}
```

Output: —

```
Hello! Welcome to the world of Anonymous Inner Classes in Java.
```

Problem 9: Anonymous Inner Class

Task: Anonymous Inner Class Extending an Abstract Class - Create an abstract class Shape with an abstract method draw(). Write a method createShape() in another class that uses an anonymous inner class to extend Shape and provides an implementation for the draw() method.

Code: —

```
import java.util.Scanner;

// Abstract class
abstract class Shape {
    abstract void draw();
}

// Factory class
class ShapeFactory {
    public void createShape(String shapeType) {
        Shape shape;

        switch (shapeType.toLowerCase()) {
            case "circle":
                shape = new Shape() {
                    @Override
                    void draw() {
                        System.out.println("Drawing a Circle
                            using Anonymous Inner Class.");
                    }
```

```
                };
                break;

            case "rectangle":
                shape = new Shape() {
                    @Override
                    void draw() {
                        System.out.println("Drawing a Rectangle using Anonymous Inner Class.");
                    }
                };
                break;

            case "triangle":
                shape = new Shape() {
                    @Override
                    void draw() {
                        System.out.println("Drawing a Triangle using Anonymous Inner Class.");
                    }
                };
                break;

            default:
                shape = new Shape() {
                    @Override
                    void draw() {
                        System.out.println("Unknown Shape. Cannot draw!");
                    }
                };
        }

        // Call method
        shape.draw();
    }
}

// Main class
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter shape (Circle / Rectangle / Triangle): ");
        String input = sc.nextLine();

        ShapeFactory factory = new ShapeFactory();
        factory.createShape(input);

        sc.close();
    }
```

```
}
```

Output: —

```
Enter shape (Circle / Rectangle / Triangle): Rectangle
Drawing a Rectangle using Anonymous Inner Class.

Enter shape (Circle / Rectangle / Triangle): Hexagon
Unknown Shape. Cannot draw!
```

Problem 10: Anonymous Inner Class

Task: Anonymous Inner Class Extending a Regular Class - Create a class Printer with a method printMessage(). Write a method createPrinter() in another class that uses an anonymous inner class to extend Printer and overrides the printMessage() method.

Code: —

```java
// Regular class
class Printer {
    void printMessage() {
        System.out.println("Default Printer Message.");
    }
}

// Factory class
class PrinterFactory {
    public void createPrinter() {
        // Anonymous Inner Class extending Printer
        Printer printer = new Printer() {
            @Override
            void printMessage() {
                System.out.println("Overridden Message: Printing
                    using Anonymous Inner Class!");
            }
        };

        // Call the method
        printer.printMessage();
    }
}

// Main class
public class Main {
    public static void main(String[] args) {
        PrinterFactory factory = new PrinterFactory();
        factory.createPrinter();
    }
}
```

Output: —

```
Overridden Message: Printing using Anonymous Inner Class!
```