Mock Test 2 – Java – PPT

1. Create a superclass called Animal with a method makeSound() that prints the sound made by the animal. Implement subclasses Dog, Cat, and Cow that inherit from the Animal class. Implement the makeSound() method in each subclass to print the sound made by a dog, cat, and cow, respectively.

Answer:

```
// Animal superclass
class Animal {
    void makeSound() {
        System.out.println("The animal makes a sound");
    }
}

// Dog subclass
class Dog extends Animal {
    @Override
    void makeSound() {
        System.out.println("The dog barks");
    }
}

// Cat subclass
class Cat extends Animal {
    @Override
    void makeSound() {
        System.out.println("The cat meows");
    }
}

// Cow subclass
class Cow extends Animal {
    @Override
    void makeSound() {
        System.out.println("The cow moos");
    }
}

// Main class to demonstrate the implementation
public class Main {
    public static void main(String[] args) {
        Animal animal = new Animal();
```

```
        animal.makeSound(); // Output: The animal makes a sound

        Animal dog = new Dog();
        dog.makeSound();       // Output: The dog barks

        Animal cat = new Cat();
        cat.makeSound();       // Output: The cat meows

        Animal cow = new Cow();
        cow.makeSound();       // Output: The cow moos
    }
}
```

2. Create a superclass called Shape with an abstract method calculateArea() that returns the area of the shape. Implement subclasses Rectangle, Circle, and Triangle that inherit from the Shape class. Implement the calculateArea() method in each subclass to calculate and return the area of a rectangle, circle, and triangle, respectively. Then, create a class called ShapeCalculator with a method printArea(Shape shape) that accepts an object of type Shape and prints its area. Demonstrate polymorphism by passing instances of different subclasses to the printArea() method.

Answer:

```
// Shape superclass
abstract class Shape {
    abstract double calculateArea();
}

// Rectangle subclass
class Rectangle extends Shape {
    private double length;
    private double width;

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

    @Override
    double calculateArea() {
        return length * width;
    }
}
```

```
// Circle subclass
class Circle extends Shape {
    private double radius;

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

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

// Triangle subclass
class Triangle extends Shape {
    private double base;
    private double height;

    public Triangle(double base, double height) {
        this.base = base;
        this.height = height;
    }

    @Override
    double calculateArea() {
        return 0.5 * base * height;
    }
}

// ShapeCalculator class
class ShapeCalculator {
    void printArea(Shape shape) {
        double area = shape.calculateArea();
        System.out.println("Area: " + area);
    }
}

// Main class to demonstrate polymorphism
public class Main {
    public static void main(String[] args) {
        ShapeCalculator calculator = new ShapeCalculator();
```

```
        Rectangle rectangle = new Rectangle(5, 3);
        Circle circle = new Circle(2);
        Triangle triangle = new Triangle(4, 6);

        calculator.printArea(rectangle); // Output: Area: 15.0
        calculator.printArea(circle);       // Output: Area: 12.566370614359172
        calculator.printArea(triangle);    // Output: Area: 12.0
    }
}
```

3. Create a class called Person with private properties like name, age, and address. Provide public getter and setter methods for these properties. Write a program that creates an instance of the Person class, sets values for its properties using the setter methods, and displays the values using the getter methods.

Anawer:

```
class Person {
    private String name;
    private int age;
    private String address;

    // Getter methods
    public String getName() {
        return name;
    }

    public int getAge() {
        return age;
    }

    public String getAddress() {
        return address;
    }

    // Setter methods
    public void setName(String name) {
        this.name = name;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public void setAddress(String address) {
```

```
        this.address = address;
    }
}

public class Main {
    public static void main(String[] args) {
        Person person = new Person();

        // Set values using setter methods
        person.setName("John Doe");
        person.setAge(25);
        person
```

4. Create an interface called Drawable with a method draw() that has no implementation. Implement this interface in classes Circle and Rectangle. Write a program that creates objects of Circle and Rectangle and calls the draw() method on each object.

Answer:

```
// Drawable interface
interface Drawable {
    void draw();
}

// Circle class implementing the Drawable interface
class Circle implements Drawable {
    @Override
    public void draw() {
        System.out.println("Drawing a circle");
    }
}

// Rectangle class implementing the Drawable interface
class Rectangle implements Drawable {
    @Override
    public void draw() {
        System.out.println("Drawing a rectangle");
    }
}

// Main class to demonstrate the implementation
public class Main {
    public static void main(String[] args) {
        Circle circle = new Circle();
        circle.draw();          // Output: Drawing a circle

        Rectangle rectangle = new Rectangle();
```

```
        rectangle.draw();       // Output: Drawing a rectangle
    }
}
```