Core Java Mock Test - 3

Q1. 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.

Sol:-

```
class Animal {
    public void makeSound() {
        System.out.println("Sounds of Animal");
    }
}

class Dog extends Animal {
    public void makeSound() {
        System.out.println("The dog barks");
    }
}

class Cat extends Animal {
    public void makeSound() {
        System.out.println("The cat meows");
    }
}

class Cow extends Animal {
    public void makeSound() {
        System.out.println("The cow moos");
    }
}

public class MockTestQue6 {

    public static void main(String[] args) {
        Animal an = new Animal();
        an.makeSound();

        Animal dog = new Dog();
        dog.makeSound();

        Animal cat = new Cat();
        cat.makeSound();

        Animal cow = new Cow();
        cow.makeSound();
    }
}
```

Q2. 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.

Sol:-

```java
abstract class Shape {
    public abstract double calculateArea();
}

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

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

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

class Circle extends Shape {
    private double radius;

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

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

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

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

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

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

public class MockTestQue7 {
    public static void main(String[] args) {
        ShapeCalculator calculator = new ShapeCalculator();

        Rectangle rectangle = new Rectangle(5, 10);
        calculator.printArea(rectangle);
```

```
        Circle circle = new Circle(3);
         calculator.printArea(circle);

        Triangle triangle = new Triangle(4, 6);
         calculator.printArea(triangle);
    }
}
```

Q3. 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.

Sol:-

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

    public String getName() {
        return name;
    }

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

    public int getAge() {
        return age;
    }

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

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }
}

public class MockTestQue8 {
    public static void main(String[] args) {
        Person person = new Person();

        person.setName("Pankaj Gola");
        person.setAge(20);
        person.setAddress("611 Faridabad");

        System.out.println("Name: " + person.getName());
        System.out.println("Age: " + person.getAge());
        System.out.println("Address: " + person.getAddress());
    }
}
```

Q4. 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.

Sol:-

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

class Circle implements Drawable {
    public void draw() {
        System.out.println("Drawing a circle");
    }
}

class Rectangle implements Drawable {
    public void draw() {
        System.out.println("Drawing a rectangle");
    }
}

public class MockTestQue9 {
    public static void main(String[] args) {
        Drawable circle = new Circle();
        Drawable rectangle = new Rectangle();

        circle.draw();
        rectangle.draw();
    }
}
```