

RegNo: 23MCA1030

Name: Vinayak Kumar Singh

Java Programming Lab (PMCA502P)

Exercise 5e Types of Inheritance

1. Write a Java program that demonstrates multilevel inheritance. Define a superclass Animal with private attributes name and sound. Implement a constructor to initialize these attributes. Create a subclass Mammal inheriting from Animal with an additional attribute numLegs. Implement a constructor in the Mammal class that initializes all attributes, including those inherited from Animal. Finally, create another subclass Dog inheriting from Mammal with an additional attribute breed. Implement a constructor in the Dog class that initializes all attributes, including those inherited from both Animal and Mammal

Expected Input/Output:

Input:

Name: Max

Sound: Woof

Number of Legs: 4

Breed: Labrador

Output:

Name: Max

Sound: Woof

Number of Legs: 4

Breed: Labrador

Instructions:

Define the Animal class with private attributes name and sound, along with a constructor to initialize them.

Define the Mammal class inheriting from Animal with an additional private attribute numLegs.

Implement a constructor in the Mammal class that initializes all attributes, including those inherited from Animal.

Define the Dog class inheriting from Mammal with an additional private attribute breed.

Implement a constructor in the Dog class that initializes all attributes, including those inherited from both Animal and Mammal.

In the main method, create an instance of the Dog class, passing appropriate values to the constructor, and print the details of the dog, including name, sound, number of legs, and breed.

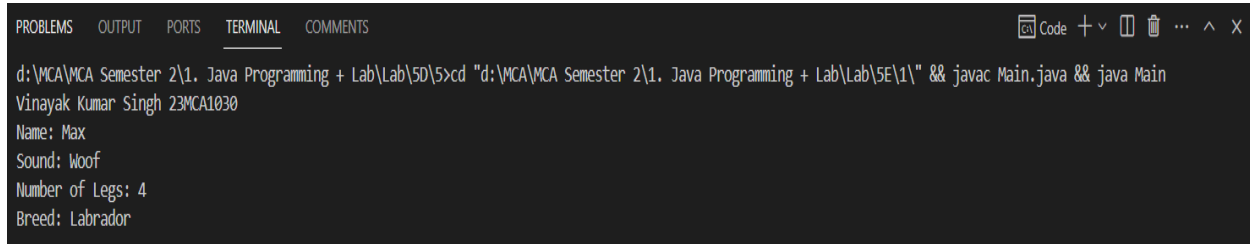
Code:

```
// Superclass Animal with private attributes name and sound
class Animal {
    private String name;
    private String sound;
    // Constructor to initialize name and sound
    Animal(String name, String sound) {
        this.name = name;
        this.sound = sound;
    }
    //Methods for name and sound
    public String getName() {
        return name;
    }
}
```

```
public String getSound() {  
    return sound;  
}  
}  
  
// Subclass Mammal inheriting from Animal with an additional private  
// attribute numLegs  
class Mammal extends Animal {  
    private int numLegs;  
    // Constructor to initialize all attributes, including those inherited from  
    // Animal  
    Mammal(String name, String sound, int numLegs) {  
        super(name, sound);  
        this.numLegs = numLegs;  
    }  
    // Method for numLegs  
    public int getNumLegs() {  
        return numLegs;  
    }  
}  
  
// Subclass Dog inheriting from Mammal with an additional private  
// attribute breed  
class Dog extends Mammal {  
    private String breed;  
    // Constructor to initialize all attributes, including those inherited from  
    // both Animal and Mammal
```

```
Dog(String name, String sound, int numLegs, String breed) {
    super(name, sound, numLegs);
    this.breed = breed;
}
// Method for breed
public String getBreed() {
    return breed;
}
}
// Main class
public class Main {
    public static void main(String[] args) {
        // Create an instance of the Dog class
        Dog dog = new Dog("Max", "Woof", 4, "Labrador");
        // Print the details of the dog
        System.out.println("Vinayak Kumar Singh 23MCA1030");
        System.out.println("Name: " + dog.getName());
        System.out.println("Sound: " + dog.getSound());
        System.out.println("Number of Legs: " + dog.getNumLegs());
        System.out.println("Breed: " + dog.getBreed());
    }
}
// Now you can run this program and see the details of the dog, including
name, sound, number of legs, and breed.
```

Output:



```
PROBLEMS OUTPUT PORTS TERMINAL COMMENTS
d:\MCA\MCA Semester 2\1. Java Programming + Lab\Lab\5D\5>cd "d:\MCA\MCA Semester 2\1. Java Programming + Lab\Lab\5E\1\" && javac Main.java && java Main
Vinayak Kumar Singh 23MCA1030
Name: Max
Sound: Woof
Number of Legs: 4
Breed: Labrador
```

2. Write a Java program that demonstrates hierarchy inheritance. Define a superclass Shape with private attributes color and filled. Implement a constructor to initialize these attributes. Create subclasses Circle, Rectangle, and Triangle inheriting from Shape. Each subclass should have specific attributes (radius for Circle, width and height for Rectangle, and base and height for Triangle) and methods to calculate their areas. Implement constructors in each subclass to initialize their specific attributes and call the superclass constructor to initialize color and filled.

Expected Input/Output:

Input:

Shape: Circle

Color: Red

Filled: true

Radius: 5

Output:

Shape: Circle

Color: Red

Filled: true

Area: 78.54

Instructions:

Define the Shape class with private attributes color and filled, along with a constructor to initialize them.

Define subclasses Circle, Rectangle, and Triangle inheriting from Shape.

Implement constructors in each subclass to initialize their specific attributes (radius for Circle, width and height for Rectangle, and base and height for Triangle) and call the superclass constructor to initialize color and filled.

Implement methods in each subclass to calculate their areas (getArea()).

In the main method, create an instance of one of the subclasses, passing appropriate values to the constructor, and print the details of the shape, including color, filled status, and area.

Code:

```
// Superclass Shape with private attributes color and filled
class Shape {
    private String color;
    private boolean filled;
    // Constructor to initialize color and filled
    Shape(String color, boolean filled) {
        this.color = color;
        this.filled = filled;
    }
    // Methods for color and filled
    public String getColor() {
```

```

        return color;
    }
    public boolean isFilled() {
        return filled;
    }
}

// Subclass Circle inheriting from Shape with specific attribute radius
class Circle extends Shape {
    private double radius;

    // Constructor to initialize attributes, including calling the superclass
    constructor
    Circle(String color, boolean filled, double radius) {
        super(color, filled);
        this.radius = radius;
    }

    // Method to calculate the area
    public double getArea() {
        return Math.PI * Math.pow(radius, 2);
    }
}

// Subclass Rectangle inheriting from Shape with attributes width and
height
class Rectangle extends Shape {
    private double width;
    private double height;

```

```
// Constructor to initialize attributes and call superclass constructor
Rectangle(String color, boolean filled, double width, double height) {
    super(color, filled);
    this.width = width;
    this.height = height;
}

// Method to calculate area
public double getArea() {
    return width * height;
}
}

// Subclass Triangle inheriting from Shape with specific attributes base
and height
class Triangle extends Shape {
    private double base;
    private double height;

    // Constructor to initialize attributes, including calling the superclass
    constructor

    Triangle(String color, boolean filled, double base, double height) {
        super(color, filled);
        this.base = base;
        this.height = height;
    }

    // Method to calculate area
    public double getArea() {
```

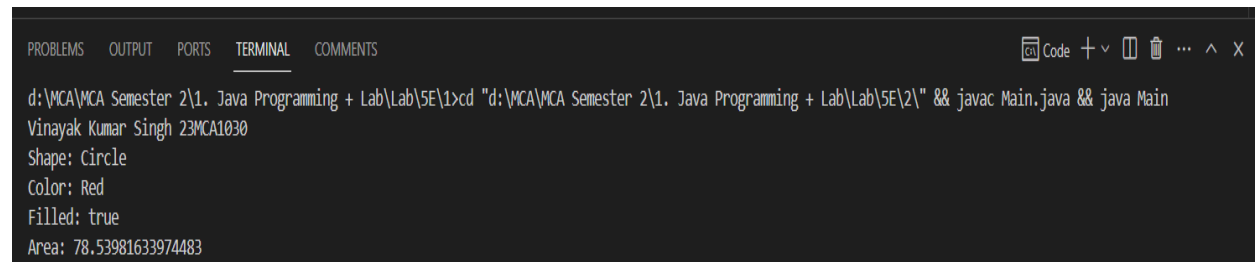


```

        return 0.5 * base * height;
    }
}
// Main class
public class Main {
    public static void main(String[] args) {
        // Create an instance of one of the subclasses i.e Circle
        Circle circle = new Circle("Red", true, 5.0);
        // Printing details of shape color, filled status, and area
        System.out.println("Vinayak Kumar Singh 23MCA1030");
        System.out.println("Shape: Circle");
        System.out.println("Color: " + circle.getColor());
        System.out.println("Filled: " + circle.isFilled());
        System.out.println("Area: " + circle.getArea());
    }
}

```

Output:



```

PROBLEMS  OUTPUT  PORTS  TERMINAL  COMMENTS
d:\MCA\MCA Semester 2\1. Java Programming + Lab\Lab\5E\1>cd "d:\MCA\MCA Semester 2\1. Java Programming + Lab\Lab\5E\2\" && javac Main.java && java Main
Vinayak Kumar Singh 23MCA1030
Shape: Circle
Color: Red
Filled: true
Area: 78.53981633974483

```

3. Write a Java program that demonstrates hybrid inheritance using only single inheritance. Define a superclass `Animal` with private attributes `name` and `sound`. Implement a constructor to initialize these attributes. Create subclasses `Mammal` and `Bird` inheriting from `Animal`. Each subclass should have specific attributes (`numLegs` for `Mammal` and `canFly` for `Bird`) and methods to display their characteristics. Finally, create a subclass `Bat` inheriting from both `Mammal` and `Bird`, demonstrating the combination of features from multiple parent classes.

Expected Input/Output:

Output:

Name: Bat

Sound: Squeak

Number of Legs: 2

Can Fly: true

Instructions:

Define the `Animal` class with private attributes `name` and `sound`, along with a constructor to initialize them.

Define subclasses `Mammal` and `Bird` inheriting from `Animal`.

Implement constructors in each subclass to initialize their specific attributes (`numLegs` for `Mammal` and `canFly` for `Bird`).

Implement methods in each subclass to display their characteristics (`displayCharacteristics()`).

Define a subclass `Bat` inheriting from both `Mammal` and `Bird`, demonstrating the combination of features from multiple parent classes.

In the main method, create an instance of the `Bat` class and print its characteristics.

Code:

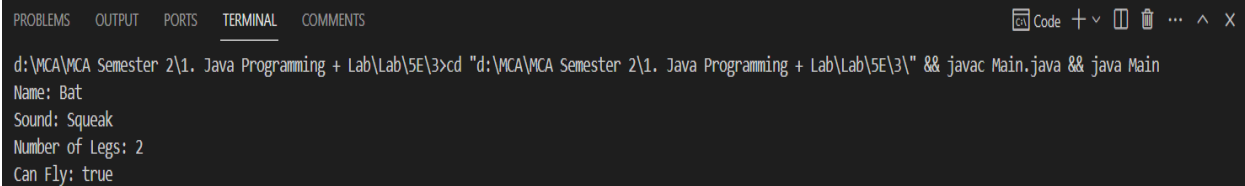
```
class Animal {
    private String name;
    private String sound;
    public Animal(String name, String sound) {
        this.name = name;
        this.sound = sound;
    }
    public void displayCharacteristics() {
        System.out.println("Name: " + name);
        System.out.println("Sound: " + sound);
    }
}

class Mammal extends Animal {
    private int numLegs;
    public Mammal(String name, String sound, int numLegs) {
        super(name, sound);
        this.numLegs = numLegs;
    }
    public void displayCharacteristics() {
        super.displayCharacteristics();
        System.out.println("Number of Legs: " + numLegs);
    }
}
```

```
}  
  
class Bird extends Animal {  
    private boolean canFly;  
  
    public Bird(String name, String sound, boolean canFly) {  
        super(name, sound);  
        this.canFly = canFly;  
    }  
  
    public void displayCharacteristics() {  
        super.displayCharacteristics();  
        System.out.println("Can Fly: " + canFly);  
    }  
}  
  
class Bat extends Mammal {  
    private boolean canFly;  
  
    public Bat(String name, String sound, int numLegs, boolean canFly) {  
        super(name, sound, numLegs);  
        this.canFly = canFly;  
    }  
  
    public void displayCharacteristics() {  
        super.displayCharacteristics();  
        System.out.println("Can Fly: " + canFly);  
    }  
}
```

```
public class Main {  
    public static void main(String[] args) {  
        Bat bat = new Bat("Bat", "Squeak", 2, true);  
        bat.displayCharacteristics();  
    }  
}
```

Output:



The screenshot shows a code editor with a terminal window. The terminal has tabs for PROBLEMS, OUTPUT, PORTS, TERMINAL, and COMMENTS. The TERMINAL tab is active, showing the command to compile and run the Java program. The output of the program is displayed below the command.

```
d:\MCA\MCA Semester 2\1. Java Programming + Lab\Lab\5E\3>cd "d:\MCA\MCA Semester 2\1. Java Programming + Lab\Lab\5E\3\" && javac Main.java && java Main  
Name: Bat  
Sound: Squeak  
Number of Legs: 2  
Can Fly: true
```