1. Consider a scenario where you have a base class Vehicle with properties like make, model, and methods like startEngine() and stopEngine(). Now, you have a derived class Car that extends the Vehicle class. The Car class has an additional property numDoors and a method drive().

Code :

class Vehicle {

String make;

String model;

void startEngine() {

System.out.println("Engine started.");

}

void stopEngine() {

System.out.println("Engine stopped.");

}

}

class Car extends Vehicle {

int numDoors;

void drive() {

System.out.println("Car is in motion.");

}

}

1. Consider a scenario where you have three classes - Animal, Mammal, and Dog. The Animal class has properties like name and eat(). The Mammal class extends Animal and has an additional property furColor and a method giveBirth(). Finally, the Dog class extends Mammal and has an additional property breed and a method bark().

Code:

class Animal {

String name;

void eat() {

System.out.println("Animal is eating.");

}

}

class Mammal extends Animal {

String furColor;

void giveBirth() {

System.out.println("Mammal is giving birth.");

}

}

class Dog extends Mammal {

String breed;

void bark() {

System.out.println("Dog is barking.");

}

}

1. Consider a scenario where you have a base class Shape with properties like color and a method draw(). Now, you have three derived classes - Circle, Rectangle, and Triangle - each extending the Shape class. Each derived class has its own properties (radius for Circle, length and width for Rectangle, and base and height for Triangle) and overrides the draw() method to provide their own implementation.

Code :

// Base class

class Shape {

String color;

void draw() {

System.out.println("Drawing a shape.");

}

}

// Derived class 1

class Circle extends Shape {

double radius;

@Override

void draw() {

System.out.println("Drawing a circle.");

}

}

// Derived class 2

class Rectangle extends Shape {

double length;

double width;

@Override

void draw() {

System.out.println("Drawing a rectangle.");

}

}

// Derived class 3

class Triangle extends Shape {

double base;

double height;

@Override

void draw() {

System.out.println("Drawing a triangle.");

}

}

1. Create a base class Shape with a method named calculateArea() that prints "Calculating area in the base class." Now, create two derived classes, Circle and Rectangle, both extending the Shape class. Override the calculateArea() method in each derived class to provide specific implementations for calculating the area of a circle and a rectangle.

Code :

// Base class

class Shape {

// Method to calculate area

void calculateArea() {

System.out.println("Calculating area in the base class.");

}

}

// Derived class 1

class Circle extends Shape {

// Additional property

double radius;

// Override method to calculate area of a circle

@Override

void calculateArea() {

double area = Math.PI \* radius \* radius;

System.out.println("Calculating area of a circle: " + area);

}

}

// Derived class 2

class Rectangle extends Shape {

// Additional properties

double length;

double width;

// Override method to calculate area of a rectangle

@Override

void calculateArea() {

double area = length \* width;

System.out.println("Calculating area of a rectangle: " + area);

}

}

public class Main {

public static void main(String[] args) {

// Demonstrating method overriding

Shape shape1 = new Circle();

shape1.calculateArea(); // Calls the overridden method in Circle

Shape shape2 = new Rectangle();

shape2.calculateArea(); // Calls the overridden method in Rectangle

}

}