/\*  
\* To change this license header, choose License Headers in Project Properties.  
\* To change this template file, choose Tools | Templates  
\* and open the template in the editor.  
\*/  
package semana3;

/\*\*  
\*  
\* @author DELL  
\*/  
import java.util.Scanner;

// Clase principal que contiene el método main y los ejercicios  
public class Semana3 {

    // Método principal que se ejecuta al iniciar el programa  
    public static void main(String[] args) {  
        Scanner scanner = new Scanner(System.in);

        // Mostrar el menú de opciones para el usuario  
        System.out.println("Ejercicios de Programación Orientada a Objetos - Semana 3");  
        System.out.println("Seleccione el número del ejercicio a ejecutar:");  
        System.out.println("1. Encapsulamiento");  
        System.out.println("2. Herencia");  
        System.out.println("3. Polimorfismo");  
        System.out.println("4. Abstracción");  
        System.out.println("5. Composición");  
        System.out.print("Ingrese su opción (1-5): ");

        // Leer la opción seleccionada por el usuario  
        int opcion = scanner.nextInt();  
        System.out.println();

        // Ejecutar el ejercicio correspondiente según la opción ingresada  
        switch (opcion) {  
            case 1:  
                encapsulamiento();  
                break;  
            case 2:  
                herencia();  
                break;  
            case 3:  
                polimorfismo();  
                break;  
            case 4:  
                abstraccion();  
                break;  
            case 5:  
                composicion();  
                break;  
            default:  
                System.out.println("Opción no válida. Por favor ingrese un número del 1 al 5.");  
        }

        scanner.close(); // Cerrar el lector de entrada  
    }

    // EJERCICIO 1: Encapsulamiento  
    public static void encapsulamiento() {

        // Clase interna que representa una cuenta bancaria  
        class CuentaBancaria {  
            // El atributo 'saldo' está encapsulado (privado)  
            private double saldo;

            // Constructor que inicializa el saldo con un valor inicial  
            public CuentaBancaria(double saldoInicial) {  
                saldo = saldoInicial;  
            }

            // Método público para depositar dinero  
            public void depositar(double monto) {  
                if (monto > 0) {  
                    saldo += monto;  
                }  
            }

            // Método público para retirar dinero  
            public void retirar(double monto) {  
                if (monto > 0 && monto <= saldo) {  
                    saldo -= monto;  
                }  
            }

            // Método público para consultar el saldo actual  
            public double obtenerSaldo() {  
                return saldo;  
            }  
        }

        // Crear una nueva cuenta con un saldo inicial  
        CuentaBancaria cuenta = new CuentaBancaria(100);

        // Realizar un depósito  
        cuenta.depositar(50);

        // Realizar un retiro  
        cuenta.retirar(30);

        // Mostrar el saldo final  
        System.out.println("Saldo final de la cuenta: $" + cuenta.obtenerSaldo());  
    }

    // EJERCICIO 2: Herencia  
    public static void herencia() {

        // Clase base o superclase  
        class Animal {  
            public void hablar() {  
                System.out.println("Este animal emite un sonido.");  
            }  
        }

        // Clase derivada que hereda de Animal  
        class Perro extends Animal {  
            // Se sobreescribe el método hablar para personalizar el comportamiento  
            @Override  
            public void hablar() {  
                System.out.println("El perro dice: guau guau.");  
            }  
        }

        // Crear un objeto de la clase Perro y llamar al método hablar  
        Perro miPerro = new Perro();  
        miPerro.hablar();  
    }

    // EJERCICIO 3: Polimorfismo  
    public static void polimorfismo() {

        // Interfaz que define un comportamiento común para las figuras  
        interface Figura {  
            double calcularArea();  
        }

        // Implementación de la interfaz para un círculo  
        class Circulo implements Figura {  
            private double radio;

            public Circulo(double radio) {  
                this.radio = radio;  
            }

            public double calcularArea() {  
                return Math.PI \* radio \* radio;  
            }  
        }

        // Implementación de la interfaz para un rectángulo  
        class Rectangulo implements Figura {  
            private double ancho;  
            private double alto;

            public Rectangulo(double ancho, double alto) {  
                this.ancho = ancho;  
                this.alto = alto;  
            }

            public double calcularArea() {  
                return ancho \* alto;  
            }  
        }

        // Crear figuras y calcular sus áreas usando la misma interfaz  
        Figura figura1 = new Circulo(3.0);        // Área de círculo  
        Figura figura2 = new Rectangulo(4.0, 5.0); // Área de rectángulo

        // Mostrar los resultados  
        System.out.println("Área del círculo: " + figura1.calcularArea());  
        System.out.println("Área del rectángulo: " + figura2.calcularArea());  
    }

    // EJERCICIO 4: Abstracción  
    public static void abstraccion() {

        // Clase abstracta que define un comportamiento general  
        abstract class Vehiculo {  
            public abstract void moverse(); // Método abstracto sin implementación  
        }

        // Clase concreta que extiende la abstracción  
        class Automovil extends Vehiculo {  
            // Implementación del método abstracto  
            public void moverse() {  
                System.out.println("El automóvil se desplaza por carretera.");  
            }  
        }

        // Crear un objeto de tipo Automovil usando referencia de tipo Vehiculo  
        Vehiculo miVehiculo = new Automovil();  
        miVehiculo.moverse(); // Llamar al método implementado  
    }

    // EJERCICIO 5: Composición  
    public static void composicion() {

        // Clase Motor con un método para encenderlo  
        class Motor {  
            public void encender() {  
                System.out.println("Motor encendido correctamente.");  
            }  
        }

        // Clase Moto que contiene un objeto Motor (relación de composición)  
        class Moto {  
            private Motor motor; // Componente interno

            // Constructor que crea un nuevo motor al construir la moto  
            public Moto() {  
                motor = new Motor();  
            }

            // Método que utiliza el motor para arrancar la moto  
            public void arrancar() {  
                motor.encender();  
                System.out.println("La motocicleta está lista para funcionar.");  
            }  
        }

        // Crear una moto y arrancarla  
        Moto miMoto = new Moto();  
        miMoto.arrancar();  
    }  
}