PROGRAMACIÓN II

Trabajo Práctico 7: Herencia y Polimorfismo en Java

Repositorio Github: https://github.com/AguP10/UTN-TUPaD-P2

Alumno: Palacios Fernando Agustin.

1. Vehículos y herencia básica

- **Clase base: Vehículo con atributos marca, modelo y método mostrarInfo()**
- **Subclase: Auto con atributo adicional cantidadPuertas, sobrescribe mostrarInfo()**
- **Tarea: Instanciar un auto y mostrar su información completa.**

Clase Vehiculo

```
class Vehiculo {

    protected String marca;
    protected String modelo;

    public Vehiculo(String marca, String modelo) {
        this.marca = marca;
        this.modelo = modelo;
    }

    public void mostrarInfo() {
        System.out.println("Marca: " + marca);
        System.out.println("Modelo: " + modelo);
    }
}
```

Subclase Auto

```
class Auto extends Vehiculo {

    private int cantidadPuertas;
```

```
    public Auto(String marca, String modelo, int cantidadPuertas) {
        super(marca, modelo);
        this.cantidadPuertas = cantidadPuertas;
    }

    @Override
    public void mostrarInfo() {
        super.mostrarInfo();
        System.out.println("Cantidad de puertas: " + cantidadPuertas);
    }
}
```

Main

```
public class Main {

    public static void main(String[] args) {
        Auto miAuto = new Auto("Toyota", "Corolla", 4);
        miAuto.mostrarInfo();
    }
}
```

2. Figuras geométricas y métodos abstractos

- **Clase abstracta: Figura con método calcularArea() y atributo nombre**
- **Subclases: Círculo y Rectángulo implementan el cálculo del área**
- **Tarea: Crear un array de figuras y mostrar el área de cada una usando polimorfismo.**

Clase Figura

```
abstract class Figura {

    protected String nombre;

    public Figura(String nombre) {
```

```
        this.nombre = nombre;
    }

    public abstract double calcularArea();

    public void mostrarArea() {
        System.out.println("Área del " + nombre + ": " + calcularArea());
    }
}
```

Subclase Circulo

```
class Circulo extends Figura {

    private double radio;

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

    @Override
    public double calcularArea() {
        return Math.PI * radio * radio;
    }
}
```

Subclase Rectángulo

```
class Rectangulo extends Figura {

    private double base;
    private double altura;
```

```
    public Rectangulo(double base, double altura) {
        super("Rectángulo");
        this.base = base;
        this.altura = altura;
    }

    @Override
    public double calcularArea() {
        return base * altura;
    }
}
```

Main

```
public class Main {

    public static void main(String[] args) {
        // Creamos un array de figuras
        Figura[] figuras = new Figura[3];
        figuras[0] = new Circulo(5);
        figuras[1] = new Rectangulo(4, 6);
        figuras[2] = new Circulo(2.5);

        for (Figura figura : figuras) {
            figura.mostrarArea();
        }
    }
}
```

3. Empleados y polimorfismo

● Clase abstracta: Empleado con método calcularSueldo()

● Subclases: EmpleadoPlanta, EmpleadoTemporal

● Tarea: Crear lista de empleados, invocar calcularSueldo() polimórficamente, usar instanceof para clasificar

Clase Empleado

```
abstract class Empleado {

    protected String nombre;

    public Empleado(String nombre) {
        this.nombre = nombre;
    }

    public abstract double calcularSueldo();

    public String getNombre() {
        return nombre;
    }
}
```

Subclase EmpleadoPlanta

```
class EmpleadoPlanta extends Empleado {

    private double sueldoBase;
    private double bonificacion;

    public EmpleadoPlanta(String nombre, double sueldoBase, double bonificacion) {
        super(nombre);
        this.sueldoBase = sueldoBase;
        this.bonificacion = bonificacion;
    }

    @Override
    public double calcularSueldo() {
        return sueldoBase + bonificacion;
    }
```

```
}
```

Subclase EmpleadoTemporal

```
class EmpleadoTemporal extends Empleado {

    private int diasTrabajados;
    private double pagoPorDia;

    public EmpleadoTemporal(String nombre, int diasTrabajados, double pagoPorDia) {
        super(nombre);
        this.diasTrabajados = diasTrabajados;
        this.pagoPorDia = pagoPorDia;
    }

    @Override
    public double calcularSueldo() {
        return diasTrabajados * pagoPorDia;
    }
}
```

Main

```
import java.util.ArrayList;

import java.util.List;
public class Main {

    public static void main(String[] args) {
        List<Empleado> empleados = new ArrayList<>();
        empleados.add(new EmpleadoPlanta("Ana", 150000, 20000));
        empleados.add(new EmpleadoTemporal("Luis", 20, 5000));

        for (Empleado e : empleados) {
```

```
            System.out.println(e.getNombre() + "$" + e.calcularSueldo());
        }
    }
}
```

4. Animales y comportamiento sobrescrito

- **Clase: Animal con método hacerSonido() y describirAnimal()**
- **Subclases: Perro, Gato, Vaca sobrescriben hacerSonido() con @Override**
- **Tarea: Crear lista de animales y mostrar sus sonidos con polimorfismo**

Clase Animal

```
class Animal {

    protected String nombre;

    public Animal(String nombre) {
        this.nombre = nombre;
    }

    public void hacerSonido() {
        System.out.println(nombre + " hace un sonido desconocido.");
    }

    public void describirAnimal() {
        System.out.println("El animal es " + nombre + ".");
    }
}
```

Subclase Perro

```
class Perro extends Animal {
```

```
    public Perro(String nombre) {
        super(nombre);
    }

    @Override
    public void hacerSonido() {
        System.out.println(nombre + "Ladrar");
    }
}
```

Subclase Gato

```
class Gato extends Animal {

    public Gato(String nombre) {
        super(nombre);
    }

    @Override
    public void hacerSonido() {
        System.out.println(nombre + "Maullar");
    }
}
```

Subclase Vaca

```
class Vaca extends Animal {

    public Vaca(String nombre) {
        super(nombre);
    }

    @Override
    public void hacerSonido() {
```

```
        System.out.println(nombre + "Mugir");
    }
}
```

Main

```
import java.util.ArrayList;

import java.util.List;

public class Main {

    public static void main(String[] args) {
        List<Animal> animales = new ArrayList<>();
        animales.add(new Perro("Fefo"));
        animales.add(new Gato("Michi"));
        animales.add(new Vaca("Lola"));

        System.out.println("Descripciones ");
        for (Animal a : animales) {
            a.describirAnimal();
        }

        System.out.println("\nSonidos");
        for (Animal a : animales) {
            a.hacerSonido();
        }
    }
}
```