PROGRAMACIÓN II

Trabajo Práctico 7: Herencia y Polimorfismo en Java

Alumno: Angelelli, Rodrigo Martin
Comisión No 7
Link GitHub: https://github.com/roangelelli/Programacion2_TUPAD_Angelelli_Rodrigo

Objetivo general: Comprender y aplicar los conceptos de herencia y polimorfismo en la Programación Orientada a Objetos, reconociendo su importancia para la reutilización de código, la creación de jerarquías de clases y el diseño flexible de soluciones en Java.

Caso Práctico: Desarrollar las siguientes Katas en Java aplicando herencia y polimorfismo. Se recomienda repetir cada kata para afianzar el concepto.

1. **Vehículos y herencia básica:**
    a. **Clase base:** Vehículo con atributos marca, modelo y método mostrarInfo()
    b. **Subclase**: Auto con atributo adicional cantidadPuertas, sobrescribe mostrarInfo()
    c. **Tarea**: Instanciar un auto y mostrar su información completa.

```java
package Ejercicio1;
public 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 + " | Modelo: " + modelo);
    }
}
```

```java
package Ejercicio1;
public class Auto extends Vehiculo {

    protected int cantidadPuertas;

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

    @Override
    public void mostrarInfo() {
        // Acceso directo gracias a protected
        System.out.println("Auto - Marca: " + marca
                + " | Modelo: " + modelo
                + " | Puertas: " + cantidadPuertas);
    }
}
```

```java
package Ejercicio1;
public class Main {
    public static void main(String[] args) {
        Auto auto1 = new Auto("Toyota", "Corolla", 4);

        System.out.println("Información del vehículo:");
        auto1.mostrarInfo();    }

}
```

```
run:
Información del vehículo:
Auto - Marca: Toyota | Modelo: Corolla | Puertas: 4
BUILD SUCCESSFUL (total time: 0 seconds)
```

2. **Figuras geométricas y métodos abstractos:**
    a. **Clase abstracta**: Figura con método calcularArea() y atributo nombre
    b. **Subclases**: Círculo y Rectángulo implementan el cálculo del área
    c. **Tarea**: Crear un array de figuras y mostrar el área de cada una usando polimorfismo.

```java
package Ejercicio2;
public abstract class Figura {

    protected String nombre;

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

    public String getNombre() {
        return nombre;
    }

    public abstract double calcularArea();
}
```

```java
package Ejercicio2;
public class Rectangulo extends Figura {

    private double ancho;
    private double alto;

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

    @Override
    public double calcularArea() {
        return ancho * alto;
    }
}
```

```java
package Ejercicio2;
public class Circulo extends Figura {

    private double radio;

    public Circulo(String nombre, double radio) {
        super(nombre);
        this.radio = radio;
    }

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

```java
package Ejercicio2;
public class Main {
    public static void main(String[] args) {

        // Array de la superclase
        Figura[] figuras = new Figura[3];

        figuras[0] = new Circulo("Círculo pequeño", 2.0);
        figuras[1] = new Rectangulo("Rectángulo mediano", 4.0, 3.0);
        figuras[2] = new Circulo("Círculo grande", 5.5);

        System.out.println("Áreas de las figuras:");

        for (Figura f : figuras) {
            System.out.println(f.getNombre() + " - Área: " + f.calcularArea());
        }
    }
}
```

```
run:
Áreas de las figuras:
Círculo pequeño - Área: 12.566370614359172
Rectángulo mediano - Área: 12.0
Círculo grande - Área: 95.03317777109123
BUILD SUCCESSFUL (total time: 0 seconds)
```

3. **Empleados y polimorfismo:**
    a. **Clase abstracta**: Empleado con método calcularSueldo()
    b. **Subclases**: EmpleadoPlanta, EmpleadoTemporal
    c. **Tarea**: Crear lista de empleados, invocar calcularSueldo() polimórficamente, usar instanceof para clasificar

```java
package Ejercicio3;
public abstract class Empleado {

    protected String nombre;

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

    public String getNombre() {...3 lines }
    public abstract double calcularSueldo();
}
```

```java
package Ejercicio3;
public class EmpleadoTemporal extends Empleado {

    protected int diasTrabajados;
    protected 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;
    }
}
```

```java
package Ejercicio3;
public class EmpleadoPlanta extends Empleado {

    protected double sueldoBase;
    protected double bonoAntiguedad;

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

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

```java
package Ejercicio3;
import java.util.ArrayList;

public class Main {

    public static void main(String[] args) {

        ArrayList<Empleado> empleados = new ArrayList<>();

        // Agregamos empleados de planta
        empleados.add(new EmpleadoPlanta("Ana", 300000, 50000));
        empleados.add(new EmpleadoPlanta("Luis", 280000, 30000));

        // Agregamos empleados temporales
        empleados.add(new EmpleadoTemporal("Carla", 20, 8000));
        empleados.add(new EmpleadoTemporal("Diego", 15, 7500));

        int cantPlanta = 0;
        int cantTemporales = 0;

        System.out.println("Listado de empleados y sueldos:");

        for (Empleado e : empleados) {
            // Polimorfismo: se llama al calcularSueldo() de la subclase correspondiente
            double sueldo = e.calcularSueldo();
            System.out.println("Empleado: " + e.getNombre() + " - Sueldo: " + sueldo);

            // Clasificación usando instanceof
            if (e instanceof EmpleadoPlanta) {
                cantPlanta++;
            } else if (e instanceof EmpleadoTemporal) {
                cantTemporales++;
            }
        }

        System.out.println("\nClasificación de empleados:");
        System.out.println("Empleados de planta: " + cantPlanta);
        System.out.println("Empleados temporales: " + cantTemporales);
    }
}
```

```
run:
Listado de empleados y sueldos:
Empleado: Ana - Sueldo: 350000.0
Empleado: Luis - Sueldo: 310000.0
Empleado: Carla - Sueldo: 160000.0
Empleado: Diego - Sueldo: 112500.0

Clasificación de empleados:
Empleados de planta: 2
Empleados temporales: 2
BUILD SUCCESSFUL (total time: 0 seconds)
```

4. **Animales y comportamiento sobrescrito:**
    a. **Clase**: Animal con método hacerSonido() y describirAnimal()
    b. **Subclases**: Perro, Gato, Vaca sobrescriben hacerSonido() con @Override
    c. **Tarea**: Crear lista de animales y mostrar sus sonidos con polimorfismo

```java
package Ejercicio4;
public class Animal {

    protected String nombre;

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

    public void hacerSonido() {
        System.out.println("El animal hace un sonido.");
    }

    public void describirAnimal() {
        System.out.println("Este es un animal llamado " + nombre);
    }

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

```java
package Ejercicio4;
public class Perro extends Animal {

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

    @Override
    public void hacerSonido() {
        System.out.println(nombre + " dice: ¡Guau guau!");
    }
}
```

```java
package Ejercicio4;
public class Gato extends Animal {

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

    @Override
    public void hacerSonido() {
        System.out.println(nombre + " dice: Miau miau");
    }
}
```

```java
package Ejercicio4;
public class Vaca extends Animal {

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

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

```java
package Ejercicio4;

import java.util.ArrayList;

public class Main {

    public static void main(String[] args) {

        ArrayList<Animal> animales = new ArrayList<>();

        animales.add(new Perro("Firulais"));
        animales.add(new Gato("Misu"));
        animales.add(new Vaca("Lola"));

        System.out.println("Sonidos de los animales:");

        for (Animal a : animales) {
            a.hacerSonido();
        }
    }
}
```

```
Sonidos de los animales:
Firulais dice: ¡Guau guau!
Misu dice: Miau miau
Lola dice: Muuuu
BUILD SUCCESSFUL (total time: 0 seconds)
```