Trabajo Práctico 7: Herencia y Polimorfismo en Java

Materia: Programación II

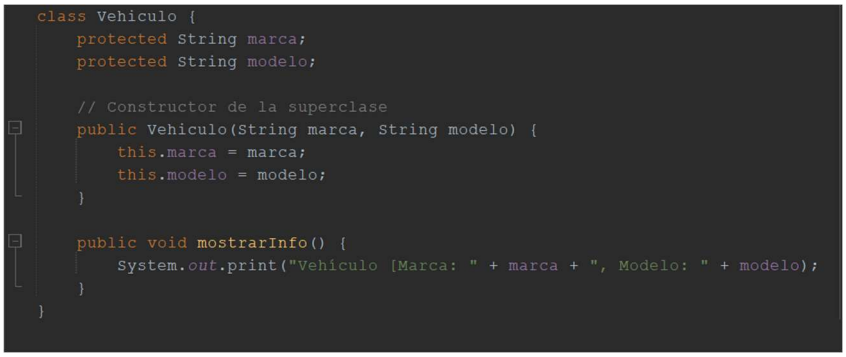
Nombre: Lopez Leandro

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

- Clase base: Vehículo con atributos marca, modelo y método mostrarInfo()

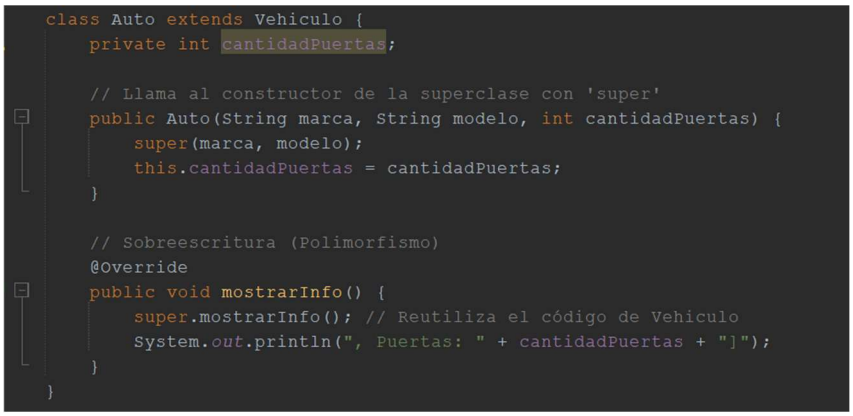


```java
class Vehiculo {
    protected String marca;
    protected String modelo;

    // Constructor de la superclase
    public Vehiculo(String marca, String modelo) {
        this.marca = marca;
        this.modelo = modelo;
    }

    public void mostrarInfo() {
        System.out.print("Vehículo [Marca: " + marca + ", Modelo: " + modelo);
    }
}
```

- Subclase: Auto con atributo adicional cantidadPuertas, sobrescribe mostrarInfo()



```java
class Auto extends Vehiculo {
    private int cantidadPuertas;

    // Llama al constructor de la superclase con 'super'
    public Auto(String marca, String modelo, int cantidadPuertas) {
        super(marca, modelo);
        this.cantidadPuertas = cantidadPuertas;
    }

    // Sobreescritura (Polimorfismo)
    @Override
    public void mostrarInfo() {
        super.mostrarInfo(); // Reutiliza el código de Vehiculo
        System.out.println(", Puertas: " + cantidadPuertas + "]");
    }
}
```

- Tarea: Instanciar un auto y mostrar su información completa.

```java
public class Principal {

    /**
     * @param args the command line arguments
     */

    public static void main(String[] args) {
        System.out.println("--- Kata 1: Vehículos ---");
        Auto miAuto = new Auto("Toyota", "Corolla", 4);
        miAuto.mostrarInfo();
    }
}
```

```
tput - TrabajoPractico7 (run)
run:
--- Kata 1: Veh?culos ---
Veh?culo [Marca: Toyota, Modelo: Corolla, Puertas: 4]
BUILD SUCCESSFUL (total time: 0 seconds)
```

2. Figuras geométricas y métodos abstractos

● Clase abstracta: Figura con método calcularArea() y atributo nombre

```java
abstract class Figura {
    protected String nombre;

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

    // Método abstracto: fuerza la implementación en subclases
    public abstract double calcularArea();

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

● Subclases: Círculo y Rectángulo implementan el cálculo del área

```java
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;
    }
}
```

```java
class Rectangulo extends Figura {
    private double ancho;
    private double alto;

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

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

● Tarea: Crear un array de figuras y mostrar el área de cada una usando polimorfismo.

```java
public class Principal {
    public static void main(String[] args) {
        System.out.println("\n--- Kata 2: Figuras Geométricas ---");

        List<Figura> figuras = new ArrayList<>();
        figuras.add(new Circulo(5.0)); // Upcasting
        figuras.add(new Rectangulo(4.0, 6.0));

        for (Figura figura : figuras) {
            // Polimorfismo: llama al método correcto
            System.out.printf("Área de %s: %.2f\n", figura.getNombre(), figura.calcularArea());
        }
    }
}
```

```
TrabajoPractico7.1 (run)
run:

--- Kata 2: Figuras Geom?tricas ---
?rea de C?rculo: 78,54
?rea de Rect?ngulo: 24,00
BUILD SUCCESSFUL (total time: 0 seconds)
```

3. Empleados y polimorfismo

● Clase abstracta: Empleado con método calcularSueldo()

```java
abstract class Empleado {
    protected String nombre;

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

    // Método abstracto para el cálculo del sueldo
    public abstract double calcularSueldo();

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

● Subclases: EmpleadoPlanta, EmpleadoTemporal

```java
class EmpleadoPlanta extends Empleado {
    private double sueldoBase;

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

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

```java
class EmpleadoTemporal extends Empleado {
    private double tarifaPorHora;
    private int horasTrabajadas;

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

    @Override
    public double calcularSueldo() {
        return tarifaPorHora * horasTrabajadas;
    }
}
```

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

```java
public class Principal {

    public static void main(String[] args) {
        System.out.println("\n--- Kata 3: Empleados y Polimorfismo ---");

        List<Empleado> nomina = new ArrayList<>();
        nomina.add(new EmpleadoPlanta("Ana López", 3500.00));
        nomina.add(new EmpleadoTemporal("Jorge Ruiz", 15.50, 160));

        for (Empleado emp : nomina) {
            System.out.printf("Sueldo de %s: $%.2f", emp.getNombre(), emp.calcularSueldo());

            // Uso de 'instanceof' para clasificar
            if (emp instanceof EmpleadoPlanta) {
                System.out.println(" (Tipo: Planta)");
            } else if (emp instanceof EmpleadoTemporal) {
                System.out.println(" (Tipo: Temporal)");
            }
        }
    }
}
```

```
put - TrabajoPractico7.2 (run)
run:

--- Kata 3: Empleados y Polimorfismo ---
Sueldo de Ana L?pez: $3500,00 (Tipo: Planta)
Sueldo de Jorge Ruiz: $2480,00 (Tipo: Temporal)
BUILD SUCCESSFUL (total time: 0 seconds)
```

4. Animales y comportamiento sobrescrito

● Clase: Animal con método hacerSonido() y describirAnimal()

```java
class Animal {
    protected String nombre;

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

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

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

● Subclases: Perro, Gato, Vaca sobrescriben hacerSonido() con @Override

```java
class Perro extends Animal {
    public Perro(String nombre) {
        super(nombre);
    }

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

```java
class Gato extends Animal {
    public Gato(String nombre) {
        super(nombre);
    }

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

```java
class Vaca extends Animal {
    public Vaca(String nombre) {
        super(nombre);
    }

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

- Tarea: Crear lista de animales y mostrar sus sonidos con polimorfismo

```java
public class Principal {

    public static void main(String[] args) {
        System.out.println("\n--- Kata 4: Animales y Comportamiento ---");

        List<Animal> granja = new ArrayList<>();
        granja.add(new Perro("Lassie"));
        granja.add(new Gato("Silvestre"));
        granja.add(new Vaca("Margarita"));

        for (Animal animal : granja) {
            // Polimorfismo: Llama al método 'hacerSonido()' específico
            animal.hacerSonido();
        }
    }
}
```

```
TrabajoPractico7.3 (run)
run:

--- Kata 4: Animales y Comportamiento ---
Lassie dice: ?Guau!
Silvestre dice: ?Miau!
Margarita dice: ?Muuu!
BUILD SUCCESSFUL (total time: 0 seconds)
```