

PROGRAMACIÓN II

Trabajo Práctico 7: Herencia y Polimorfismo en Java

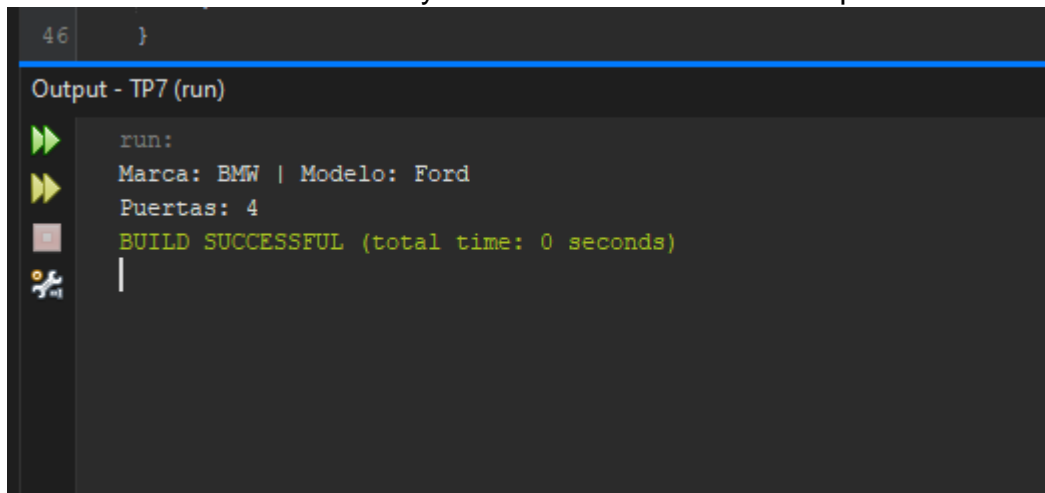
Alumno: Mauro Gaspar
Comisión: 4

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()
- Subclase: Auto con atributo adicional cantidadPuertas, sobrescribe mostrarInfo()
- Tarea: Instanciar un auto y mostrar su información completa.



```
46    }  
  
Output - TP7 (run)  
  
run:  
Marca: BMW | Modelo: Ford  
Puertas: 4  
BUILD SUCCESSFUL (total time: 0 seconds)
```

```
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);
    }
}

class Auto extends Vehiculo {
    private int cantidadPuertas;

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

    @Override
    public void mostrarInfo() {
        super.mostrarInfo();
        System.out.println("Puertas: " + cantidadPuertas);
    }
}

// Instanciar un auto y mostrar su información completa
public class MainVehiculo {
    public static void main(String[] args) {
        Auto al = new Auto("BMW", "Ford", 4);
        al.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.

```
abstract class Figura {  
    protected String nombre;  
  
    public Figura(String nombre) {  
        this.nombre = nombre;  
    }  
  
    public abstract double calcularArea();  
  
    public void mostrarInfo() {  
        // La implementación de calcularArea() se resolverá en tiempo de ejecución  
        System.out.println("Figura: " + nombre + " | Área: " + calcularArea());  
    }  
}  
  
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;  
    }  
}  
  
class Rectangulo extends Figura {  
    private double base, altura;  
  
    public Rectangulo(double base, double altura) {  
        super("Rectángulo");  
        this.base = base;  
        this.altura = altura;  
    }  
}
```

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

public class MainFiguras {

    public static void main(String[] args) {
        Figura[] figuras = {
            new Circulo(10),
            new Rectangulo(8, 14)
        };

        for (Figura f : figuras) {
            f.mostrarInfo(); // Polimorfismo: llama al método correcto
        }
    }
}
```

ut - TP7 (run)

```
run:
Figura: Circulo | Área: 314.1592653589793
Figura: Rectángulo | Área: 112.0
BUILD SUCCESSFUL (total time: 0 seconds)
```

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

```
abstract class Empleado {
    protected String nombre;

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

    public abstract double calcularSueldo();

    public void mostrarTipo() {
        System.out.println("Empleado: " + nombre);
    }
}

class EmpleadoPlanta extends Empleado {
    private double salarioBase;
    private double bono;

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

    @Override
    public double calcularSueldo() {
        return salarioBase + bono;
    }
}

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;
    }
}
```

```
public class MainEmpleados {  
    public static void main(String[] args) {  
  
        // Se declara una lista genérica que solo puede contener referencias de  
        ArrayList<Empleado> empleados = new ArrayList<>();  
  
        // Se produce el Upcasting implicito.  
        empleados.add(new EmpleadoPlanta("Loana", 60000, 5000));  
        // La lista almacena referencias de la superclase(Empleado),  
        // pero los objetos reales son los de la subclases  
        empleados.add(new EmpleadoTemporal("Mauro", 25, 1800));  
  
        for (Empleado e : empleados) {  
            // Al llamar a e.calcularSueldo(), ocurre el Polimorfismo por Sobres  
            System.out.println("Empleado: " + e.nombre + " | Sueldo: " + e.calcu  
  
            // Se utiliza el operador instanceof para verificar la clase real de  
            if (e instanceof EmpleadoPlanta) {  
                System.out.println("→ Es empleado de planta");  
            } else if (e instanceof EmpleadoTemporal) {  
                System.out.println("→ Es empleado temporal");  
            }  
        }  
    }  
}
```

out - TP7 (run)

```
run:  
Empleado: Loana | Sueldo: 65000.0  
→ Es empleado de planta  
Empleado: Mauro | Sueldo: 45000.0  
→ Es empleado temporal  
BUILD SUCCESSFUL (total time: 0 seconds)
```

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

```
class Animal {  
    protected String nombre;  
  
    public Animal(String nombre) {  
        this.nombre = nombre;  
    }  
  
    public void hacerSonido() {  
        System.out.println("Sonido genérico...");  
    }  
  
    public void describirAnimal() {  
        System.out.println("Soy un animal llamado " + nombre);  
    }  
}  
  
class Perro extends Animal {  
    public Perro(String nombre) { super(nombre); }  
  
    @Override  
    public void hacerSonido() {  
        System.out.println("Guau guau!");  
    }  
}  
  
class Gato extends Animal {  
    public Gato(String nombre) { super(nombre); }  
  
    @Override  
    public void hacerSonido() {  
        System.out.println("Miau miau!");  
    }  
}  
  
class Vaca extends Animal {  
    public Vaca(String nombre) { super(nombre); }  
  
    @Override  
    public void hacerSonido() {  
        System.out.println("Muuu!");  
    }  
}
```

```
public class MainAnimales {  
  
    public static void main(String[] args) {  
        // El tipo del arreglo es Animal (la clase base o superclase) animal  
        Animal[] animales = {  
            new Perro("Samy"),  
            new Gato("Laira"),  
            new Vaca("Vanina")  
        };  
  
        // El programa itera sobre cada elemento () en el arreglo.  
        // Cada variable es tratada como un objeto de tipo en el código fuente  
        for (Animal a : animales) {  
            a.describirAnimal();  
            a.hacerSonido(); // Polimorfismo  
        }  
    }  
}
```

ut - TP7 (run)

```
run:  
Soy un animal llamado Samy  
Guau guau!  
Soy un animal llamado Laira  
Miau miau!  
Soy un animal llamado Vanina  
Muuu!  
BUILD SUCCESSFUL (total time: 0 seconds)
```

LINK:

<https://github.com/27mau/UTN-Programacion-2/tree/main/TP7>