# PROGRAMACIÓN II

## Trabajo Práctico 7: Herencia y Polimorfismo en Java

Alumno: Mauro Zavatti – legajo 1215

GitHub: https://github.com/maurozavatti/UTN-TUPaD--P2

### 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.

```
package tp7.herencia.polimorfismo.Vehiculo;


public class VehiculosMain {
   public static void main(String[] args) {
      Auto miAuto = new Auto("Toyota", "Corolla", 4);
      miAuto.mostrarInfo();
   }
}
package tp7.herencia.polimorfismo.Vehiculo;


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);
        System.out.println("Modelo: " + modelo);
    }
}
package tp7.herencia.polimorfismo.Vehiculo;

public 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);
    }
}
```

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.

```
package tp7.herencia.polimorfismo.FigurasGeometricas;



public class FGeoMain {
   public static void main(String[] args) {
      // Array
      Figura[] figuras = new Figura[3];
      figuras[0] = new Circulo(5.0);
      figuras[1] = new Rectangulo(4.0, 6.0);
      figuras[2] = new Circulo(2.5);



      for (Figura f : figuras) {
         f.mostrarArea();  // Cada figura usa su propio método calcularArea()
      }
}
}
package tp7.herencia.polimorfismo.FigurasGeometricas;

abstract class Figura {
   protected String nombre;

   public Figura(String nombre) {
```

```
        this.nombre = nombre;
    }

    // Método abstracto
    public abstract double calcularArea();

    public void mostrarArea() {
        System.out.println("El área del " + nombre + " es: " + calcularArea());
    }
}
package tp7.herencia.polimorfismo.FigurasGeometricas;

class Circulo extends Figura {
    private double radio;

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

    @Override
    public double calcularArea() {
        return Math.PI * Math.pow(radio, 2);
    }
}
package tp7.herencia.polimorfismo.FigurasGeometricas;
```

```
public 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;
    }
}
```

### 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

```
package tp7.herencia.polimorfismo.Empleados;


import java.util.ArrayList;
```

```
public class EmpleadosMain {
  public static void main(String[] args) {

    // Crear lista de empleados
    ArrayList<Empleado> empleados = new ArrayList<>();
    empleados.add(new EmpleadoPlanta("Mauro", 250000));
    empleados.add(new EmpleadoTemporal("Valentina", 80, 1800));
    empleados.add(new EmpleadoPlanta("Francisco", 300000));
    empleados.add(new EmpleadoTemporal("Rodrigo", 60, 1500));

    // Mostrar sueldos usando polimorfismo
    for (Empleado e : empleados) {
      e.mostrarSueldo();

      // Clasificar con instanceof
      if (e instanceof EmpleadoPlanta) {
        System.out.println("→ Es un empleado de planta.\n");
      } else if (e instanceof EmpleadoTemporal) {
        System.out.println("→ Es un empleado temporal.\n");
      }
    }
  }
}
package tp7.herencia.polimorfismo.Empleados;

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

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

    // Método abstracto (debe implementarse en las subclases)
    public abstract double calcularSueldo();

    public void mostrarSueldo() {
        System.out.println(nombre + " gana $" + calcularSueldo());
    }
}
package tp7.herencia.polimorfismo.Empleados;

public class EmpleadoTemporal extends Empleado {
    private int horasTrabajadas;
    private double valorHora;

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

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

```
}
```

```
package tp7.herencia.polimorfismo.Empleados;

public class EmpleadoPlanta extends Empleado {
   private double sueldoMensual;

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

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

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

```
package tp7.herencia.polimorfismo.Animales;

import java.util.ArrayList;

public class AnimalesMain {
```

```
    public static void main(String[] args) {

        // Lista de animales
        ArrayList<Animal> animales = new ArrayList<>();
        animales.add(new Perro("Firulais"));
        animales.add(new Gato("Michi"));
        animales.add(new Vaca("Lola"));

        // Mostrar comportamiento con polimorfismo
        for (Animal a : animales) {
            a.describirAnimal();
            a.hacerSonido();

            // Clasificación usando instanceof
            if (a instanceof Perro) {
                System.out.println("→ Es un perro.\n");
            } else if (a instanceof Gato) {
                System.out.println("→ Es un gato.\n");
            } else if (a instanceof Vaca) {
                System.out.println("→ Es una vaca.\n");
            }
        }
    }
}
package tp7.herencia.polimorfismo.Animales;
```

```
public class Animal {
    protected String nombre;

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

    // Método que será sobrescrito
    public void hacerSonido() {
        System.out.println("El animal hace un sonido.");
    }

    public void describirAnimal() {
        System.out.println("Este es un animal llamado " + nombre + ".");
    }
}
package tp7.herencia.polimorfismo.Animales;

public class Perro extends Animal {

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

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

```
}
```

```
package tp7.herencia.polimorfismo.Animales;

public class Gato extends Animal {

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

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

```
package tp7.herencia.polimorfismo.Animales;

public class Vaca extends Animal {

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

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