

Trabajo Practico N°7  
de  
Programación II

Estudiante: Tobias Leiva  
Unidad: Herencia y Polimorfismo  
Universidad Tecnológica Nacional

## 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.

Clase Vehículo:

```
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);  
    }  
}
```

Clase Auto:

```
public class Auto extends Vehiculo{  
    private int cantidadPuertas;  
  
    public Auto(int cantidadPuertas, String marca, String modelo) {  
        super(marca, modelo);  
        this.cantidadPuertas = cantidadPuertas;  
    }  
  
    @Override  
    public void mostrarInfo(){
```

```

        System.out.println("Modelo: " + this.modelo);
        System.out.println("Marca: " + this.marca);
        System.out.println("Puertas: " + cantidadPuertas);
    }
}

```

Clase Principal:

```

public class Principal {
    public static void main(String[] args) {
        Auto auto1 = new Auto(4, "FORD", "FOCUS");
        auto1.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:

```

public abstract class Figura {
    private String nombre;

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

    public void calcularArea(){
    }
}

Clase Circulo:
public class Circulo extends Figura {
    private double radio;
}

```

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

@Override
public void calcularArea(){
    System.out.println("Area = " + Math.PI * Math.pow(radio, 2));
}

Clase Rectángulo:
public class Rectangulo extends Figura {
    private double base;
    private double altura;

    public Rectangulo(double base, double altura, String nombre) {
        super(nombre);
        this.base = base;
        this.altura = altura;
    }

    @Override
    public void calcularArea(){
        System.out.println("Area = " + base * altura);
    }

}

Clase Principal:
public class Principal {
    public static void main(String[] args) {
        ArrayList<Figura> figuras = new ArrayList<>();
        Rectangulo r1 = new Rectangulo(5.5, 4.0, "Rectangulo1");
    }
}
```

```

Rectangulo r2 = new Rectangulo(2, 4.0, "Rectangulo2");
Circulo c1 = new Circulo(10, "circulo1");
Circulo c2 = new Circulo(5, "circulo2");

figuras.add(r1);
figuras.add(r2);
figuras.add(c1);
figuras.add(c2);

for (Figura figura : figuras) {
    figura.calcularArea();
}

}
}

```

### 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:

```

public abstract class Empleado {

    private String nombre;
    private int id;

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

    public String getNombre() {
        return nombre;
    }
}

```

```
public double calcularSueldo(Empleado e) {  
    if (e instanceof EmpleadoPlanta) {  
        return 900000.0;  
    } else if (e instanceof EmpleadoTemporal) {  
        return 850000.0;  
    } else {  
        return 0;  
    }  
}  
  
Clase EmpleadoPlanta:  
public class EmpleadoPlanta extends Empleado {  
    public EmpleadoPlanta(String nombre, int id) {  
        super(nombre, id);  
    }  
}  
  
Clase EmpleadoTemporal:  
public EmpleadoTemporal(String nombre, int id) {  
    super(nombre, id);  
}  
}  
  
Clase Principal:  
public static void main(String[] args) {  
    ArrayList<Empleado> empleados = new ArrayList<>();  
  
    EmpleadoPlanta ep1 = new EmpleadoPlanta("Tobias", 1);  
    EmpleadoPlanta ep2 = new EmpleadoPlanta("Marco", 2);  
    EmpleadoTemporal et3 = new EmpleadoTemporal("Juliana", 3);  
    EmpleadoTemporal et4 = new EmpleadoTemporal("Martina", 4);
```

```

        empleados.add(ep1);
        empleados.add(ep2);
        empleados.add(et3);
        empleados.add(et4);

        for(Empleado e : empleados) {
            System.out.println("El empleado " + e.getNombre() + " cobra: " + e.calcularSueldo(e));
        }
    }
}

```

#### 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:

```

public abstract class Animal {

    protected String color;

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

    public void hacerSonido(){
    }

    public void describirAnimal(){
    }

}

```

Clase Perro:

```

public class Perro extends Animal{
    private String raza;
}

```

```
public Perro(String raza, String color) {  
    super(color);  
    this.raza = raza;  
}  
  
@Override  
public void hacerSonido(){  
    System.out.println("Guau, Guau");  
}  
  
@Override  
public void describirAnimal(){  
    System.out.println("La raza de este perro es: " + this.raza +  
"\nY su color es: " + super.color);  
}  
  
}  
Clase Gato:  
public class Gato extends Animal{  
  
    public Gato(String color) {  
        super(color);  
    }  
  
    @Override  
    public void hacerSonido(){  
        System.out.println("Miau, miau");  
    }  
    @Override  
    public void describirAnimal(){  
        System.out.println("El color de este gato es: " +  
super.color);  
    }  
}
```

```
    }  
}  
Clase Vaca:
```

```
public class Vaca extends Animal {
```

```
    public Vaca(String color) {  
        super(color);  
    }
```

```
    @Override  
    public void hacerSonido(){  
        System.out.println("Muuu, Muuu");  
    }  
    @Override  
    public void describirAnimal(){  
        System.out.println("El color de esta vaca es: " +  
super.color);  
    }
```

```
}
```

```
Clase Principal:
```

```
public class Principal {
```

```
    public static void main(String[] args) {  
        ArrayList<Animal> animales = new ArrayList<>();
```

```
        Perro p1 = new Perro("Pastor aleman", "Marron y negro");  
        Gato g1 = new Gato("Negro");  
        Gato g2 = new Gato("Marron claro");  
        Vaca v1 = new Vaca("Negra y blanca");
```

```
        animales.add(p1);  
        animales.add(g1);  
        animales.add(g2);  
        animales.add(v1);
```

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

Repositorio: <https://github.com/Tobias-L7/Trabajos-Practicos-P2.git>