**PROGRAMACIÓN II**

**Trabajo Práctico 7: Herencia y Polimorfismo en Java**

**Caso Práctico 1**

**Alumna: Eugenia Demarchi**

**Comisión: 7**

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()**

```
public class Vehiculo {
private String marca;
private 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);
}

   public String getMarca() {
      return marca;
   }

   public String getModelo() {
      return modelo;
   }
}
```

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

```
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(){
   super.mostrarInfo();
   System.out.println("Cantidad puertas: " + cantidadPuertas);
}
}
```

**● Tarea: Instanciar un auto y mostrar su información completa.**

```
public class Principal {

   public static void main(String[] args) {
Auto a = new Auto(2, "Peugeot", "306");
Auto b = new Auto(4, "Honda", "Civic");
a.mostrarInfo();
b.mostrarInfo();
   }
}
```

**2. Figuras geométricas y métodos abstractos**

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

```
public abstract class Figura {
private String nombre;

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

   }

   public String getNombre() {
      return nombre;
   }
public abstract double calcularArea();
}
```

● **Subclases: Círculo y Rectángulo implementan el cálculo del área**

```
public class Circulo extends Figura {
   private double radio;

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

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

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

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

```
public class Principal {
     public static void main(String[] args) {
ArrayList<Figura> figuras = new ArrayList<>();

figuras.add(new Rectangulo(3, 4, "Rectangulo"));
figuras.add(new Circulo(3, "Circulo"));

  for (Figura figura: figuras) {
         System.out.println("Area de " + figura.getNombre() + " " +  figura.calcularArea());
      }
}
}
```

**3. Empleados y polimorfismo**

**● Clase abstracta: Empleado con método calcularSueldo()**

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

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

   public String getNombre() {
      return nombre;
   }

   public abstract double calcularSueldo();
}
```

**● Subclases: EmpleadoPlanta, EmpleadoTemporal**

```
public class EmpleadoPlanta extends Empleado{
private double sueldo;

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

  @Override
  public double calcularSueldo(){
     return sueldo;
  }

public class EmpleadoTemporal extends Empleado{
private double sueldoPorHora;
private int cantidadHorasTrabajadas;

   public EmpleadoTemporal(double sueldoPorHora, int cantidadHorasTrabajadas, String
nombre) {
      super(nombre);
      this.sueldoPorHora = sueldoPorHora;
      this.cantidadHorasTrabajadas = cantidadHorasTrabajadas;
   }
```

```
  @Override
  public double calcularSueldo(){
     return sueldoPorHora * cantidadHorasTrabajadas;
  }
}
```

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

```
public class Principal {
     public static void main(String[] args) {
ArrayList<Empleado> empleados = new ArrayList<>();

empleados.add(new EmpleadoPlanta(2000.0,"Jorge"));
empleados.add(new EmpleadoTemporal(70, 200, "Luisa"));


for (Empleado e : empleados) {
   String tipo = (e instanceof EmpleadoPlanta) ? "Planta" : "Temporal";
   System.out.println(tipo + " - " + e.getNombre() + " - Sueldo: " + e.calcularSueldo());
}
}
}
```

**4. Animales y comportamiento sobrescrito**

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

```
public abstract class Animal {
private String nombre;

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

   public String getNombre() {
      return nombre;
   }

public abstract String hacerSonido();

public String describirAnimal(){
```

```
 return nombre;
}

}
```

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

```
public class Perro extends Animal{

   public Perro(String nombre) {
      super(nombre);
   }
@Override
public String hacerSonido(){
   return "Guau";
}
}

public class Gato extends Animal{

   public Gato(String nombre) {
      super(nombre);
   }
@Override
public String hacerSonido(){
   return "Miau";
}
}

public class Vaca extends Animal{

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

@Override
public String hacerSonido(){
   return "Muuu";
}
}
```

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

```
public class Principal {
public static void main(String[] args) {
   ArrayList <Animal> animales = new ArrayList<>();
   animales.add(new Perro("Nino"));
   animales.add(new Vaca("Lola"));
```

```
    animales.add(new Gato("Luna"));

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