

## PROGRAMACIÓN II

### *Trabajo Práctico N.º 7: Herencia y polimorfismo*

Estudiante: Emilia Gómez Juárez

Objetivo:

Comprender y aplicar los conceptos de herencia y polimorfismo en la Programación Orientada a Objetos, reconociendo su importancia para la reutilización de código, la creación de jerarquías de clases y el diseño flexible de soluciones en Java.

1. Vehículos y herencia básica:

Clase base: Vehículo

```
public class Vehiculo {
12     protected String marca;
13     protected String modelo;
14
15     public Vehiculo(String marca, String modelo) {
16         this.marca = marca;
17         this.modelo = modelo;
18     }
19
20     public String getMarca() {
21         return marca;
22     }
23
24     public void setMarca(String marca) {
25         this.marca = marca;
26     }
27
28     public String getModelo() {
29         return modelo;
30     }
31
32     public void setModelo(String modelo) {
33         this.modelo = modelo;
34     }
35
36     public void mostrarInfo() {
37         System.out.println("Marca: " + this.marca
38             + "\nModelo: " + this.modelo);
39     }
40 }
```

## PROGRAMACIÓN II

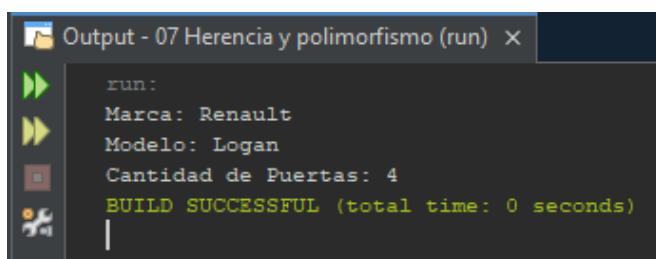
### Subclase Auto

```
11 public class Auto extends Vehiculo {
12     private int cantidadPuertas;
13
14     public Auto(String marca, String modelo, int cantidadPuertas) {
15         super(marca, modelo);
16         this.cantidadPuertas = cantidadPuertas;
17     }
18
19     public int getCantidadPuertas() {
20         return cantidadPuertas;
21     }
22
23     public void setCantidadPuertas(int cantidadPuertas) {
24         this.cantidadPuertas = cantidadPuertas;
25     }
26
27
28     @Override
29     public void mostrarInfo () {
30         System.out.println("Marca: " + this.marca
31                             + "\nModelo: " + this.modelo
32                             + "\nCantidad de Puertas: " + this.cantidadPuertas);
33     }
34 }
35
```

### Main

```
11 public class HerenciaBasica {
12     public static void main(String[] args) {
13         Vehiculo auto = new Auto("Renault", "Logan", 4);
14
15         auto.mostrarInfo();
16     }
17 }
```

### Resultado



```
Output - 07 Herencia y polimorfismo (run) x
run:
Marca: Renault
Modelo: Logan
Cantidad de Puertas: 4
BUILD SUCCESSFUL (total time: 0 seconds)
```

## PROGRAMACIÓN II

### 2. Figuras geométricas y métodos abstractos:

Clase abstracta: Figura

```
12  abstract class Figura {
13
14      protected String nombre;
15
16      public Figura(String nombre) {
17          this.nombre = nombre;
18      }
19
20      public String getNombre() {
21          return nombre;
22      }
23
24      public void setNombre(String nombre) {
25          this.nombre = nombre;
26      }
27
28      protected abstract double calcularArea();
29  }
30
```

Subclase: Círculo

```
11  public class Circulo extends Figura{
12      private double radio;
13
14      public Circulo(double radio, String nombre) {
15          super(nombre);
16          this.radio = radio;
17      }
18
19      public double getRadio() {
20          return radio;
21      }
22
23      public void setRadio(double radio) {
24          this.radio = radio;
25      }
26
27      @Override
28      protected double calcularArea() {
29          return Math.PI * Math.pow(this.radio, 2);
30      }
31  }
32
```

## PROGRAMACIÓN II

Subclase: Rectángulo

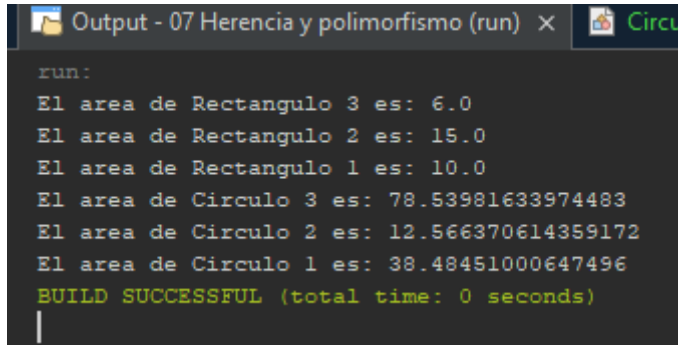
```
11 public class Rectangulo extends Figura{
12     private double base;
13     private double altura;
14
15     public Rectangulo(double base, double altura, String nombre) {
16         super(nombre);
17         this.base = base;
18         this.altura = altura;
19     }
20
21     public double getBase() {
22         return base;
23     }
24
25     public void setBase(double base) {
26         this.base = base;
27     }
28
29     public double getAltura() {
30         return altura;
31     }
32
33     public void setAltura(double altura) {
34         this.altura = altura;
35     }
36
37     @Override
38     protected double calcularArea() {
39         return base * altura;
40     }
41 }
```

Main

```
13 public class Figuras {
14     public static void main(String[] args) {
15         ArrayList <Figura> figuras = new ArrayList <>();
16
17         Figura c1 = new Circulo (3.5, "Circulo 1");
18         Figura c2 = new Circulo (2, "Circulo 2");
19         Figura c3 = new Circulo (5, "Circulo 3");
20
21         Figura r1 = new Rectangulo (2, 5, "Rectangulo 1");
22         Figura r2 = new Rectangulo (5, 3, "Rectangulo 2");
23         Figura r3 = new Rectangulo (1.5, 4, "Rectangulo 3");
24
25         figuras.add(r3);
26         figuras.add(r2);
27         figuras.add(r1);
28         figuras.add(c3);
29         figuras.add(c2);
30         figuras.add(c1);
31
32         for (Figura f : figuras){
33             System.out.println("El area de " + f.getNombre()
34                 + " es: " + f.calcularArea());
35         }
36     }
37 }
```

## PROGRAMACIÓN II

Resultado



```
run:
El area de Rectangulo 3 es: 6.0
El area de Rectangulo 2 es: 15.0
El area de Rectangulo 1 es: 10.0
El area de Circulo 3 es: 78.53981633974483
El area de Circulo 2 es: 12.566370614359172
El area de Circulo 1 es: 38.48451000647496
BUILD SUCCESSFUL (total time: 0 seconds)
```

### 3. Empleados y polimorfismo

Clase abstracta: Empleado

```
1  abstract class Empleado {
12     protected String nombre;
13     protected String apellido;
14     protected String id;
15     protected double sueldo;
16
17     public Empleado(String nombre, String apellido, String id) {
18         this.nombre = nombre;
19         this.apellido = apellido;
20         this.id = id;
21     }
22     public String getNombre() {
23         return nombre;
24     }
25     public void setNombre(String nombre) {
26         this.nombre = nombre;
27     }
28     public String getApellido() {
29         return apellido;
30     }
31     public void setApellido(String apellido) {
32         this.apellido = apellido;
33     }
34     public String getId() {
35         return id;
36     }
37     public void setId(String id) {
38         this.id = id;
39     }
```

## PROGRAMACIÓN II

```
40
41     public void mostrarInfo () {
42         System.out.println( "ID: " + this.id +
43                             "\nNombre: " + this.nombre +
44                             "\nApellido: " + this.apellido +
45                             "\nSueldo: " + this.sueldo);
46         System.out.println("-----\n");
47     }
48
49     protected abstract void calcularSueldo();
50 }
```

Subclase: Empleado temporal

```
11     public class EmpleadoTemporal extends Empleado {
12
13         private int diasTrabajados;
14         public static final double PAGO_POR_DIA = 40000.0;
15
16         public EmpleadoTemporal(int diasTrabajados, String nombre, String apellido, String id) {
17             super(nombre, apellido, id);
18             this.diasTrabajados = diasTrabajados;
19         }
20
21         public int getDiasTrabajados() {
22             return diasTrabajados;
23         }
24
25         public void setDiasTrabajados(int diasTrabajados) {
26             this.diasTrabajados = diasTrabajados;
27         }
28
29         @Override
30         protected void calcularSueldo() {
31             this.sueldo = diasTrabajados * PAGO_POR_DIA;
32         }
33     }
```

Subclase: Empleado de planta

```
11     public class EmpleadoPlanta extends Empleado {
12
13         private static double SUELDO_BASE = 1000000.0;
14         private int antiguedad;
15         private static double BONIFICACION_POR_ANIO = 100000.0;
16         private double descuentos;
17
18         public EmpleadoPlanta(int antiguedad, double descuentos, String nombre, String apellido, String id) {
19             super(nombre, apellido, id);
20             this.antiguedad = antiguedad;
21             this.descuentos = descuentos;
22         }
23
24         public static double getSUELDO_BASE() {
25             return SUELDO_BASE;
26         }
27
28         public static void setSUELDO_BASE(double SUELDO_BASE) {
29             EmpleadoPlanta.SUELDO_BASE = SUELDO_BASE;
30         }
31
32         public int getAntiguedad() {
33             return antiguedad;
34         }
35
36         public void setAntiguedad(int antiguedad) {
37             this.antiguedad = antiguedad;
38         }
39     }
```

## PROGRAMACIÓN II

```
40 public static double getBONIFICACION_POR_ANIO() {
41     return BONIFICACION_POR_ANIO;
42 }
43
44 public static void setBONIFICACION_POR_ANIO(double BONIFICACION_POR_ANIO) {
45     EmpleadoPlanta.BONIFICACION_POR_ANIO = BONIFICACION_POR_ANIO;
46 }
47
48 public double getDescuentos() {
49     return descuentos;
50 }
51
52 public void setDescuentos(double descuentos) {
53     this.descuentos = descuentos;
54 }
55
56 @Override
57 protected void calcularSueldo() {
58     this.sueldo = SUELDO_BASE + (antiguedad * BONIFICACION_POR_ANIO) - descuentos;
59 }
60
61 }
```

Main

```
13 public class Empleados {
14
15     public static void main(String[] args) {
16         ArrayList<Empleado> empleados = new ArrayList<>();
17
18
19         Empleado e1 = new EmpleadoPlanta(5, 10000, "Mirtha", "González", "P001");
20         Empleado e2 = new EmpleadoTemporal(15, "Carlos", "López", "T001");
21         Empleado e3 = new EmpleadoPlanta(10, 12000, "Ana", "Ruiz", "P002");
22         Empleado e4 = new EmpleadoTemporal(8, "Luis", "Pérez", "T002");
23
24         empleados.add(e1);
25         empleados.add(e2);
26         empleados.add(e3);
27         empleados.add(e4);
28
29         for (Empleado e : empleados){
30             e.calcularSueldo();
31             e.mostrarInfo();
32         }
33     }
34 }
```

Resultados

## PROGRAMACIÓN II

```
run:
ID: P001
Nombre: Mirtha
Apellido: González
Sueldo: 1490000.0
-----

ID: T001
Nombre: Carlos
Apellido: López
Sueldo: 600000.0
-----

ID: P002
Nombre: Ana
Apellido: Ruiz
Sueldo: 1988000.0
-----

ID: T002
Nombre: Luis
Apellido: Pérez
Sueldo: 320000.0
-----

BUILD SUCCESSFUL (total time: 0 seconds)
```

#### 4. Animales y comportamiento sobrescrito

Clase base: Animal

```
1  abstract class Animal {
12
13     protected String tipo;
14
15     public Animal(String tipo) {
16         this.tipo = tipo;
17     }
18
19     public String getTipo() {
20         return tipo;
21     }
22
23     public void setTipo(String tipo) {
24         this.tipo = tipo;
25     }
26
27     public abstract void hacerSonido();
28
29     public void describirAnimal() {
30         System.out.println("Soy un animal del tipo " + tipo);
31     }
32 }
```



## PROGRAMACIÓN II

Subclase: Perro

```
11 public class Perro extends Animal {
12     private String nombre;
13     private String color;
14     private String raza;
15     public Perro(String nombre, String color, String raza, String tipo) {
16         super(tipo);
17         this.nombre = nombre;
18         this.color = color;
19         this.raza = raza;
20     }
21     public String getNombre() {
22         return nombre;
23     }
24     public void setNombre(String nombre) {
25         this.nombre = nombre;
26     }
27     public String getColor() {
28         return color;
29     }
30     public void setColor(String color) {
31         this.color = color;
32     }
33     public String getRaza() {
34         return raza;
35     }
36     public void setRaza(String raza) {
37         this.raza = raza;
38     }
39     @Override
40     public void hacerSonido() {
41         System.out.println(nombre + " dice Guau guau");
42     }
43 }
```

Subclase: Gato

```
11 public class Gato extends Animal {
12     private String nombre;
13     private String color;
14     public Gato(String nombre, String color, String tipo) {
15         super(tipo);
16         this.nombre = nombre;
17         this.color = color;
18     }
19     public String getNombre() {
20         return nombre;
21     }
22     public void setNombre(String nombre) {
23         this.nombre = nombre;
24     }
25     public String getColor() {
26         return color;
27     }
28     public void setColor(String color) {
29         this.color = color;
30     }
31     @Override
32     public void hacerSonido() {
33         System.out.println(nombre + " dice Miaaaau");
34     }
35 }
```

## PROGRAMACIÓN II

Subclase: Vaca

```
11 public class Vaca extends Animal {
12     private String nombre;
13     private String color;
14
15     public Vaca(String nombre, String color, String tipo) {
16         super(tipo);
17         this.nombre = nombre;
18         this.color = color;
19     }
20
21     public String getNombre() {
22         return nombre;
23     }
24
25     public void setNombre(String nombre) {
26         this.nombre = nombre;
27     }
28
29     public String getColor() {
30         return color;
31     }
32
33     public void setColor(String color) {
34         this.color = color;
35     }
36
37     @Override
38     public void hacerSonido() {
39         System.out.println("La vaca " + nombre + " dice Muuuu");
40     }
41 }
```

Main

```
14 public class Animales {
15     public static void main(String[] args) {
16         List<Animal> animales = new ArrayList<>();
17
18         animales.add(new Perro("Firulaís", "Negro", "Labrador", "Perro"));
19         animales.add(new Gato("Mishi", "Gris", "Gato"));
20         animales.add(new Vaca("Lola", "Marrón", "Vaca"));
21
22         for (Animal a : animales) {
23             a.describirAnimal();
24             a.hacerSonido();
25             System.out.println("-----\n");
26         }
27     }
28 }
```

## PROGRAMACIÓN II

### Resultados

```
Soy un animal del tipo Perro
Firulaís dice Guau guau
-----

Soy un animal del tipo Gato
Mishi dice Miaaaau
-----

Soy un animal del tipo Vaca
La vaca Lola dice Muuuu
-----

BUILD SUCCESSFUL (total time: 0 seconds)
```

**REPOSITORIO**      **REMOTO:**      <https://github.com/GomezJEmilia/UTN-Programacion2-TPs-EmiliaGJ/tree/0415b46703a67c1f10b103dad9984a27109f5265/07%20Herencia%20y%20polimorfismo>