

PROGRAMACIÓN II

Trabajo Práctico 7: Herencia y Polimorfismo en Java

Alumno:

Franco Sarrú

Link público de GitHub:

<https://github.com/fsarru/Programacion2.git>

OBJETIVO GENERAL

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.

MARCO TEÓRICO

Concepto	Aplicación en el proyecto
Herencia	Uso de `extends` para crear jerarquías entre clases, aprovechando el principio is-a.
Modificadores de acceso	Uso de private, protected y public para controlar visibilidad.
Constructores y super	Invocación al constructor de la superclase con super(...) para inicializar atributos.
Upcasting	Generalización de objetos al tipo de la superclase.
Instanceof	Comprobación del tipo real de los objetos antes de hacer conversiones seguras.

Downcasting	Especialización de objetos desde una clase general a una más específica.
Clases abstractas	Uso de abstract para definir estructuras base que deben ser completadas por subclases.
Métodos abstractos	Declaración de comportamientos que deben implementarse en las clases derivadas.
Polimorfismo	Uso de la sobrescritura de métodos (@Override) y llamada dinámica de métodos.
Herencia	Uso de `extends` para crear jerarquías entre clases, aprovechando el principio is-a.

1

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 u7_ejercicio1;

public class Main {

    public static void main(String[] args) {

        Auto miCoche = new Auto("Toyota", "Etios XS", 4);
        miCoche.mostrarInfo();

        Vehiculo miVehiculo = new Vehiculo("Nissan", "Sentra");
        miVehiculo.mostrarInfo();

    }

}
```

```
package u7_ejercicio1;

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);
    }
}
```

```
run:
Marca: Toyota
Modelo: Etios XS
Tipo: Auto
Cantidad de Puertas: 4
Marca: Nissan
Modelo: Sentra
BUILD SUCCESSFUL (total time: 8 seconds)
```

```
package u7_ejercicio1;

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("Tipo: Auto");
        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 u7_ejercicio2; // Mantengo tu paquete

public class Circulo extends Figura {

    private double radio;

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

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

    public void mostrarInfo() {
        mostrarNombre(); // Heredado de Figura
        System.out.println("Radio: " + radio);
        System.out.println("Area: " + calcularArea());
    }
}
```

```
package u7_ejercicio2; // Mantengo tu paquete

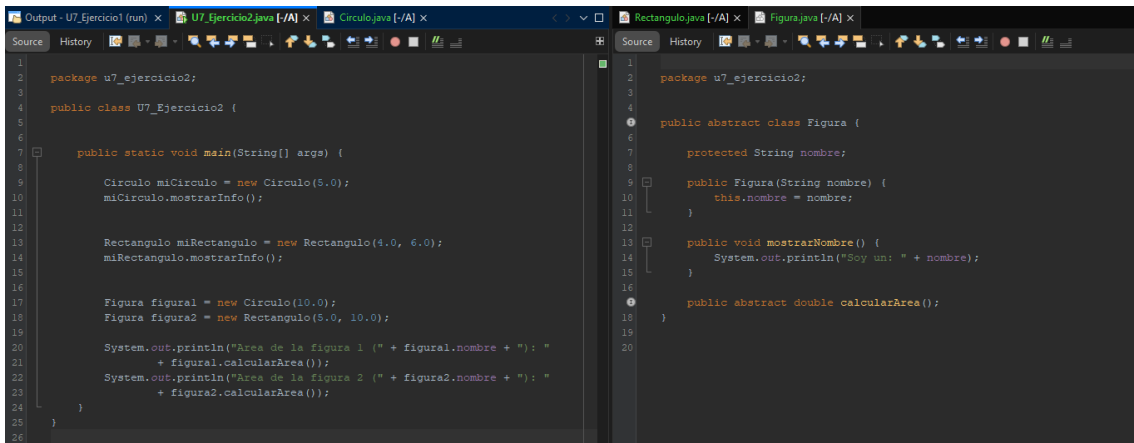
public class Rectangulo extends Figura {

    private double largo;
    private double ancho;

    public Rectangulo(double largo, double ancho) {
        super("Rectangulo");
        this.largo = largo;
        this.ancho = ancho;
    }

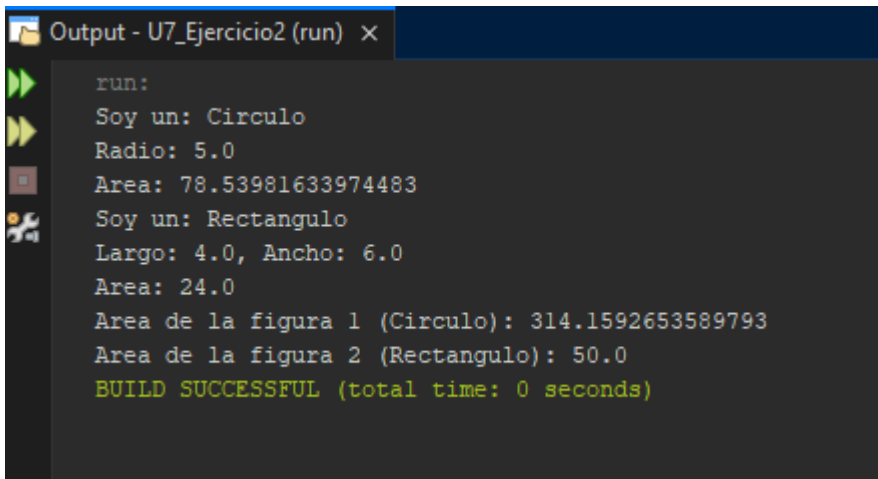
    @Override
    public double calcularArea() {
        return largo * ancho;
    }

    public void mostrarInfo() {
        mostrarNombre();
        System.out.println("Largo: " + largo + ", Ancho: " + ancho);
        System.out.println("Area: " + calcularArea());
    }
}
```



```
1 package u7_ejercicio2;
2
3 public class U7_Ejercicio2 {
4
5     public static void main(String[] args) {
6
7         Circulo miCirculo = new Circulo(5.0);
8         miCirculo.mostrarInfo();
9
10        Rectangulo miRectangulo = new Rectangulo(4.0, 6.0);
11        miRectangulo.mostrarInfo();
12
13        Figura figura1 = new Circulo(10.0);
14        Figura figura2 = new Rectangulo(5.0, 10.0);
15
16        System.out.println("Area de la figura 1 (" + figura1.nombre + "): "
17        + figura1.calcularArea());
18        System.out.println("Area de la figura 2 (" + figura2.nombre + "): "
19        + figura2.calcularArea());
20    }
21}
22
23
24
25
26
```

```
1 package u7_ejercicio2;
2
3 public abstract class Figura {
4
5     protected String nombre;
6
7     public Figura(String nombre) {
8         this.nombre = nombre;
9     }
10
11     public void mostrarNombre() {
12         System.out.println("Soy un: " + nombre);
13     }
14
15     public abstract double calcularArea();
16 }
17
18
19
20
```



```
run:
Soy un: Circulo
Radio: 5.0
Area: 78.53981633974483
Soy un: Rectangulo
Largo: 4.0, Ancho: 6.0
Area: 24.0
Area de la figura 1 (Circulo): 314.1592653589793
Area de la figura 2 (Rectangulo): 50.0
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

```
U7_Ejercicio3.java [-/A] x EmpleadoTempor.java [-/A] x
Source History
1 package u7_ejercicio3;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 public class U7_Ejercicio3 {
7
8     public static void main(String[] args) {
9
10         List<Empleado> nomina = new ArrayList<>();
11
12         nomina.add(new EmpleadoPlanta("Ana Perez", 101, 3000.0, 500.0));
13         nomina.add(new EmpleadoTemporal("Juan Gomez", 205, 20.0, 160));
14         nomina.add(new EmpleadoPlanta("Luis Castro", 102, 3500.0, 750.0));
15         nomina.add(new EmpleadoTemporal("Marta Diaz", 210, 25.0, 120));
16
17         for (Empleado emp : nomina) {
18             emp.mostrarInfoBasica();
19             System.out.println("Sueldo: " + emp.calcularSueldo());
20         }
21
22         int plantaCount = 0;
23         int temporalCount = 0;
24
25         for (Empleado emp : nomina) {
26
27             if (emp instanceof EmpleadoPlanta) {
28                 System.out.println("Clasificado: Empleado de Planta ");
29                 plantaCount++;
30             }
31             else if (emp instanceof EmpleadoTemporal) {
32                 System.out.println("Clasificado: Empleado Temporal ");
33                 temporalCount++;
34             }
35         }
36         System.out.println("\nResumen: Planta (" + plantaCount + "), "
37             + "Temporal (" + temporalCount + ")\n");
38     }
39 }
40
41
42
EmpleadoTempor.java [-/A] x
Source History
1 package u7_ejercicio3;
2
3 public class EmpleadoTemporal extends Empleado {
4
5     private double tarifaPorHora;
6     private int horasTrabajadas;
7
8     public EmpleadoTemporal(String nombre, int legajo,
9         double tarifaPorHora, int horasTrabajadas) {
10         super(nombre, legajo);
11         this.tarifaPorHora = tarifaPorHora;
12         this.horasTrabajadas = horasTrabajadas;
13     }
14
15     @Override
16     public double calcularSueldo() {
17         return tarifaPorHora * horasTrabajadas;
18     }
19 }
```

```
EmpleadoPlanta.java [-/A] x U7_Ejercicio3.java [-/A] x Empleado.java [-/A] x
Source History
1 package u7_ejercicio3;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 public class U7_Ejercicio3 {
7
8     public static void main(String[] args) {
9
10         List<Empleado> nomina = new ArrayList<>();
11
12         nomina.add(new EmpleadoPlanta("Ana Perez", 101, 3000.0, 500.0));
13         nomina.add(new EmpleadoTemporal("Juan Gomez", 205, 20.0, 160));
14         nomina.add(new EmpleadoPlanta("Luis Castro", 102, 3500.0, 750.0));
15         nomina.add(new EmpleadoTemporal("Marta Diaz", 210, 25.0, 120));
16
17         for (Empleado emp : nomina) {
18             emp.mostrarInfoBasica();
19             System.out.println("Sueldo: " + emp.calcularSueldo());
20         }
21
22         int plantaCount = 0;
23         int temporalCount = 0;
24
25         for (Empleado emp : nomina) {
26
27             if (emp instanceof EmpleadoPlanta) {
28                 System.out.println("Clasificado: Empleado de Planta ");
29                 plantaCount++;
30             }
31             else if (emp instanceof EmpleadoTemporal) {
32                 System.out.println("Clasificado: Empleado Temporal ");
33                 temporalCount++;
34             }
35         }
36         System.out.println("\nResumen: Planta (" + plantaCount + "), "
37             + "Temporal (" + temporalCount + ")\n");
38     }
39 }
40
41
42
Empleado.java [-/A] x
Source History
1 package u7_ejercicio3;
2
3 public abstract class Empleado {
4
5     protected String nombre;
6     protected int legajo;
7
8     public Empleado(String nombre, int legajo) {
9         this.nombre = nombre;
10         this.legajo = legajo;
11     }
12
13     public abstract double calcularSueldo();
14
15     public void mostrarInfoBasica() {
16         System.out.print("Nombre: " + nombre + " (Legajo: " + legajo + ")\n");
17     }
18 }
```

```
Output - U7_Ejercicio3 (run) x
run:
Nombre: Ana Perez (Legajo: 101)Sueldo: 3500.0
Nombre: Juan Gomez (Legajo: 205)Sueldo: 3200.0
Nombre: Luis Castro (Legajo: 102)Sueldo: 4250.0
Nombre: Marta Diaz (Legajo: 210)Sueldo: 3000.0
Clasificado: Empleado de Planta
Clasificado: Empleado Temporal
Clasificado: Empleado de Planta
Clasificado: Empleado Temporal

Resumen: Planta (2), dTemporal (2)

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

```
U7_Ejercicio4.java [-/A] x  Gato.java [-/A] x  Perro.java [-/A] x  Vaca.java [-/A] x
Source History
1 package u7_ejercicio4;
2
3 public class Gato extends Animal {
4
5     public Gato(String nombre) {
6         super(nombre, "Gato");
7     }
8
9     @Override
10    public void hacerSonido() {
11        System.out.println("Miau!");
12    }
13
14 }
```

```
Source History
1 package u7_ejercicio4;
2
3 public class Perro extends Animal {
4
5     public Perro(String nombre) {
6         super(nombre, "Perro");
7     }
8
9     @Override
10    public void hacerSonido() {
11        System.out.println("Gauau!");
12    }
13
14 }
```

```
Source History
1 package u7_ejercicio4;
2
3 public class Vaca extends Animal {
4
5     public Vaca(String nombre) {
6         super(nombre, "Vaca");
7     }
8
9     @Override
10    public void hacerSonido() {
11        System.out.println("Muuu!");
12    }
13
14 }
```

```
U7_Ejercicio4.java [-/A] x  Animal.java [-/A] x
Source History
1 package u7_ejercicio4;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 public class U7_Ejercicio4 {
7
8     public static void main(String[] args) {
9
10        List<Animal> granja = new ArrayList<>();
11
12        granja.add(new Perro("Rocko"));
13        granja.add(new Gato("Garfield"));
14        granja.add(new Vaca("Lola"));
15        granja.add(new Perro("Fatiga"));
16
17        for (Animal animal : granja) {
18
19            animal.describirAnimal();
20            animal.hacerSonido();
21
22        }
23
24    }
25 }
```

```
Source History
1 package u7_ejercicio4;
2
3 public class Animal {
4
5     protected String nombre;
6     protected String tipo;
7
8     public Animal(String nombre, String tipo) {
9         this.nombre = nombre;
10        this.tipo = tipo;
11    }
12
13    public void hacerSonido() {
14        System.out.println("El " + tipo + " hace un sonido genérico.");
15    }
16
17    public void describirAnimal() {
18        System.out.print("Nombre: " + nombre + " | Tipo: " + tipo + " | "
19        + "Sonido: ");
20    }
21
22 }
```

```
Output - U7_Ejercicio4 (run) x
run:
Nombre: Rocko | Tipo: Perro | Sonido: Gau!
Nombre: Garfield | Tipo: Gato | Sonido: Miau!
Nombre: Lola | Tipo: Vaca | Sonido: Muuu!
Nombre: Fatiga | Tipo: Perro | Sonido: Gau!
BUILD SUCCESSFUL (total time: 0 seconds)
```

CONCLUSIONES ESPERADAS

- Comprender el mecanismo de herencia y sus beneficios para la reutilización de código.
- Aplicar polimorfismo para lograr flexibilidad en el diseño de programas.
- Inicializar objetos correctamente usando **super** en constructores.
- Controlar el acceso a atributos y métodos con modificadores adecuados.
- Identificar y aplicar **upcasting**, **downcasting** y **instanceof** correctamente.
- Utilizar clases y métodos abstractos como base de jerarquías lógicas.
- Aplicar principios de diseño orientado a objetos en la implementación en Java.