

<https://github.com/Coman-Che/Programacion-II/tree/main/M%C3%B3dulo%204>

PROGRAMACIÓN II

Trabajo Práctico 4: Programación Orientada a Objetos II

OBJETIVO GENERAL

Comprender y aplicar conceptos de Programación Orientada a Objetos en Java, incluyendo el uso de **this**, constructores, sobrecarga de métodos, encapsulamiento y miembros estáticos, para mejorar la modularidad, reutilización y diseño del código.

MARCO TEÓRICO

Concepto	Aplicación en el proyecto
Uso de this	Referencia a la instancia actual dentro de constructores y métodos
Constructores y sobrecarga	Inicialización flexible de objetos con múltiples formas de instanciación
Métodos sobrecargados	Definición de varias versiones de un método según los parámetros recibidos
toString()	Representación legible del estado de un objeto para visualización y depuración
Atributos estáticos	Variables compartidas por todas las instancias de una clase
Métodos estáticos	Funciones de clase invocadas sin instanciar objetos
Encapsulamiento	Restringir el acceso directo a los atributos de una clase

Caso Práctico

Sistema de Gestión de Empleados

Modelar una clase **Empleado** que represente a un trabajador en una empresa. Esta clase debe incluir constructores sobrecargados, métodos sobrecargados y el uso

de atributos aplicando encapsulamiento y métodos estáticos para llevar control de los objetos creados.

CLASE EMPLEADO

Atributos:

- **int id**: Identificador único del empleado.
- **String nombre**: Nombre completo.
- **String puesto**: Cargo que desempeña.
- **double salario**: Salario actual.
- **static int totalEmpleados**: Contador global de empleados creados.

REQUERIMIENTOS

1. Uso de this:
 - Utilizar **this** en los constructores para distinguir parámetros de atributos.
2. Constructores sobrecargados:
 - Uno que reciba todos los atributos como parámetros.
 - Otro que reciba solo nombre y puesto, asignando un id automático y un salario por defecto.
 - Ambos deben incrementar **totalEmpleados**.
3. Métodos sobrecargados **actualizarSalario**:
 - Uno que reciba un porcentaje de aumento.
 - Otro que reciba una cantidad fija a aumentar.
4. Método **toString()**:
 - Mostrar id, nombre, puesto y salario de forma legible.
5. Método estático **mostrarTotalEmpleados()**:
 - Retornar el total de empleados creados hasta el momento.
6. Encapsulamiento en los atributos:
 - Restringir el acceso directo a los atributos de la clase.
 - Crear los métodos Getters y Setters correspondientes.

TAREAS A REALIZAR

1. Implementar la clase Empleado aplicando todos los puntos anteriores.
2. Crear una clase de prueba con método main que:
 - Instancie varios objetos usando ambos constructores.
 - Aplique los métodos **actualizarSalario()** sobre distintos empleados.
 - Imprima la información de cada empleado con **toString()**.
 - Muestre el total de empleados creados con **mostrarTotalEmpleados()**.

CONSEJOS

- Usá **this** en los constructores para evitar errores de asignación.

- Probá distintos escenarios para validar el comportamiento de los métodos sobrecargados.
- Asegurate de que el método **toString()** sea claro y útil para depuración.
- Confirmá que el contador **totalEmpleados** se actualiza correctamente en cada constructor.

Rta:

```
package jofre.alan.programación.pkg2.tp4;
```

```
public class Empleado {
```

```
    private int id;  
    private String nombre;  
    private String puesto;  
    private double salario;
```

```
    // Atributo estático compartido por todas las instancias  
    private static int totalEmpleados = 0;
```

```
    // ----- CONSTRUCTORES -----
```

```
    // Constructor completo
```

```
    public Empleado(int id, String nombre, String puesto, double salario) {  
        this.id = id;           // uso de this para diferenciar atributos  
        this.nombre = nombre;  
        this.puesto = puesto;  
        this.salario = salario;  
        totalEmpleados++;      // incrementar contador global  
    }
```

```
    // Constructor sobrecargado (solo nombre y puesto)
```

```
    public Empleado(String nombre, String puesto) {  
        this.id = ++totalEmpleados; // asigna id automático  
        this.nombre = nombre;  
        this.puesto = puesto;  
        this.salario = 30000;      // salario por defecto  
        // OJO: acá ya sumamos 1 al totalEmpleados al asignar el id  
        // No se suma de nuevo, porque ya lo hicimos en la línea anterior  
    }
```

```
    // ----- MÉTODOS SOBRECARGADOS -----
```

```
    // Aumentar salario por porcentaje
```

```
    public void actualizarSalario(double porcentaje) {  
        this.salario += this.salario * (porcentaje / 100);  
    }
```

```
// Aumentar salario por cantidad fija
public void actualizarSalario(int cantidad) {
    this.salario += cantidad;
}

// ----- GETTERS & SETTERS -----
public int getId() {
    return id;
}

public void setId(int id) { // si se quiere permitir modificarlo
    this.id = id;
}

public String getNombre() {
    return nombre;
}

public void setNombre(String nombre) {
    this.nombre = nombre;
}

public String getPuesto() {
    return puesto;
}

public void setPuesto(String puesto) {
    this.puesto = puesto;
}

public double getSalario() {
    return salario;
}

public void setSalario(double salario) {
    this.salario = salario;
}

// ----- MÉTODOS ESTÁTICOS -----
public static int mostrarTotalEmpleados() {
    return totalEmpleados;
}

// ----- toString() -----
@Override
```

```
public String toString() {
    return "Empleado [ID: " + id
        + ", Nombre: " + nombre
        + ", Puesto: " + puesto
        + ", Salario: $" + salario + "];"
}

}

package jofre.alan.programación.pkg2.tp4;

public class Main_TP4 {

    public static void main(String[] args) {

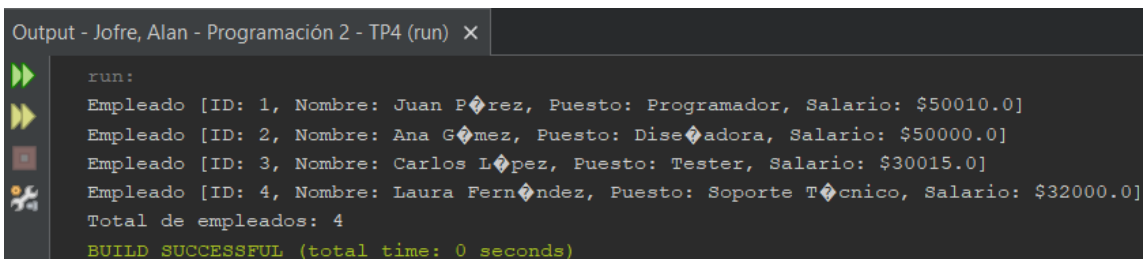
        // Crear empleados con constructor completo
        Empleado emp1 = new Empleado(1, "Juan Pérez", "Programador", 50000);
        Empleado emp2 = new Empleado(2, "Ana Gómez", "Diseñadora", 45000);

        // Crear empleados con constructor sobrecargado
        Empleado emp3 = new Empleado("Carlos López", "Tester");
        Empleado emp4 = new Empleado("Laura Fernández", "Soporte Técnico");

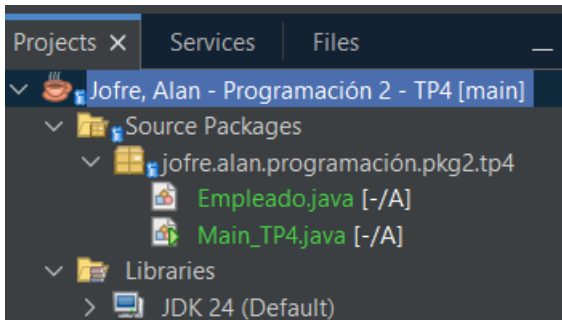
        // Aplicar aumentos de salario
        emp1.actualizarSalario(10); // aumento 10%
        emp2.actualizarSalario(5000); // aumento fijo $5000
        emp3.actualizarSalario(15); // aumento 15%
        emp4.actualizarSalario(2000); // aumento fijo $2000

        // Mostrar empleados
        System.out.println(emp1);
        System.out.println(emp2);
        System.out.println(emp3);
        System.out.println(emp4);

        // Mostrar total de empleados creados
        System.out.println("Total de empleados: " + Empleado.mostrarTotalEmpleados());
    }
}
```



```
Output - Jofre, Alan - Programación 2 - TP4 (run) X
run:
Empleado [ID: 1, Nombre: Juan Pérez, Puesto: Programador, Salario: $50010.0]
Empleado [ID: 2, Nombre: Ana Gómez, Puesto: Diseñadora, Salario: $50000.0]
Empleado [ID: 3, Nombre: Carlos López, Puesto: Tester, Salario: $30015.0]
Empleado [ID: 4, Nombre: Laura Fernández, Puesto: Soporte Técnico, Salario: $32000.0]
Total de empleados: 4
BUILD SUCCESSFUL (total time: 0 seconds)
```



CONCLUSIONES ESPERADAS

- Comprender el uso de **this** para acceder a atributos de instancia.
- Aplicar constructores sobrecargados para flexibilizar la creación de objetos.
- Aplicar encapsulamiento en los atributos.
- Implementar métodos con el mismo nombre y distintos parámetros.
- Representar objetos con **toString()** para mejorar la depuración.
- Diferenciar y aplicar atributos y métodos estáticos en Java.
- Reforzar el diseño modular y reutilizable mediante el paradigma orientado a objetos.