

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
Uso de this	Referencia a la instancia actual dentro de constructores y métodos

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.

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 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.

TAREAS A REALIZAR

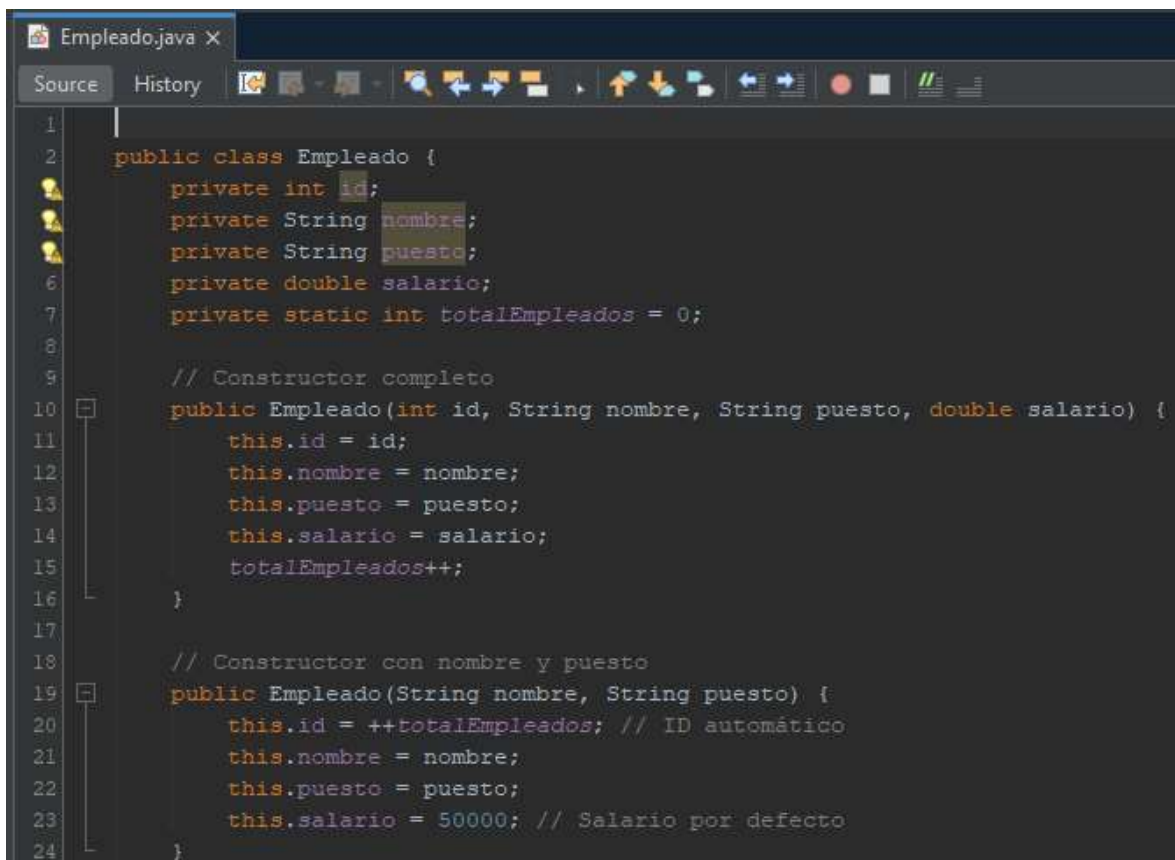
1. Implementar la clase Empleado aplicando todos los puntos anteriores.
2. Crear una clase de prueba con método main que:
 - Instancie varios objetos usando ambos constructores.
 - Aplique los métodos **actualizarSalario()** sobre distintos empleados.
 - Imprima la información de cada empleado con **toString()**.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

CONCLUSIONES ESPERADAS

- Comprender el uso de **this** para acceder a atributos de instancia.
- Aplicar constructores sobrecargados para flexibilizar la creación de objetos.
- 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.



```
1
2 public class Empleado {
3     private int id;
4     private String nombre;
5     private String puesto;
6     private double salario;
7     private static int totalEmpleados = 0;
8
9     // Constructor completo
10    public Empleado(int id, String nombre, String puesto, double salario) {
11        this.id = id;
12        this.nombre = nombre;
13        this.puesto = puesto;
14        this.salario = salario;
15        totalEmpleados++;
16    }
17
18    // Constructor con nombre y puesto
19    public Empleado(String nombre, String puesto) {
20        this.id = ++totalEmpleados; // ID automático
21        this.nombre = nombre;
22        this.puesto = puesto;
23        this.salario = 50000; // Salario por defecto
24    }
```

```

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

// Método sobrecargado: cantidad fija
public void actualizarSalario(int aumentoFijo) {
    this.salario += aumentoFijo;
}

// Método toString
@Override
public String toString() {
    return "ID: " + id + ", Nombre: " + nombre + ", Puesto: " + puesto + ", Salario: $" + salario;
}

// Método estático
public static int mostrarTotalEmpleados() {
    return totalEmpleados;
}
}

```

>MAIN

```

* @author veron
*/

public class MainEmpleados {
    public static void main(String[] args) {
        // Crear empleados con ambos constructores
        Empleado e1 = new Empleado(1, "Juan Lopez", "Gerente", 80000);
        Empleado e2 = new Empleado("Juan Perez", "Analista");
        Empleado e3 = new Empleado("Veronica Guirin", "Desarrollador");

        // Aplicar métodos sobrecargados
        e2.actualizarSalario(10); // aumento del 10%
        e3.actualizarSalario(5000); // aumento fijo

        // Mostrar información
        System.out.println(e1);
        System.out.println(e2);
        System.out.println(e3);

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

```

>Salida:

Output - Practico 4 (run) ×

```
run:
ID: 1, Nombre: Juan Lopez, Puesto: Gerente, Salario: $80000.0
ID: 2, Nombre: Juan Perez, Puesto: Analista, Salario: $50010.0
ID: 3, Nombre: Veronica Guirin, Puesto: Desarrollador, Salario: $55000.0
Total de empleados creados: 3
BUILD SUCCESSFUL (total time: 0 seconds)
```