TP 4 – Programación Orientada a Objetos II

Objetivo:

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.

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.
- Asegúrate de que el método **toString()** sea claro y útil para depuración.
- Confirmá que el contador **totalEmpleados** se actualiza correctamente en cada constructor.

```
package tp4ejer1;


import java.io.PrintStream;

import java.nio.charset.StandardCharsets;


class Empleado {

    private int id;

    private String nombre;
```

```
    private String puesto;
    private double salario;

    private static int totalEmpleados = 0;

    public Empleado(int id, String nombre, String puesto, double salario) {
        this.id = id;
        this.nombre = nombre;
        this.puesto = puesto;
        this.salario = salario;
        totalEmpleados++;
    }

    public Empleado(String nombre, String puesto) {
        this.id = totalEmpleados + 1;
        this.nombre = nombre;
        this.puesto = puesto;
        this.salario = 30000.0;
        totalEmpleados++;
    }

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

    public void actualizarSalario(int cantidadFija) {
        this.salario += cantidadFija;
```

```
    }

    public int getId() {
        return 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) {
```

```
        if (salario >= 0) {
            this.salario = salario;
        }
    }

    @Override
    public String toString() {
        return "Empleado{" + "id=" + id + ", nombre=" + nombre + ", puesto=" + puesto + ",
salario=" + salario + '}';
    }

    public static int mostrarTotalEmpleados() {
        return totalEmpleados;
    }
}


public class TP4Ejer1 {

    public static void main(String[] args) {
        System.setOut(new PrintStream(System.out, true, StandardCharsets.UTF_8));

        Empleado e1 = new Empleado(101, "Ana López", "Gerente", 60000);
        Empleado e2 = new Empleado(102, "Carlos Pérez", "Analista", 45000);

        Empleado e3 = new Empleado("Lucía Gómez", "Programadora");
        Empleado e4 = new Empleado("Jorge Torres", "Diseñador");
```

```
        e1.actualizarSalario(10.0);   // +10% a Ana
        e2.actualizarSalario(5000);   // +5000 a Carlos
        e3.actualizarSalario(15.0);   // +15% a Lucía
        e4.actualizarSalario(2000);   // +2000 a Jorge

        System.out.println(e1);
        System.out.println(e2);
        System.out.println(e3);
        System.out.println(e4);

        System.out.println("\nTotal de empleados creados: " +
Empleado.mostrarTotalEmpleados());
    }

}
```

OUTPUT:

```
run:
Empleado{id=101, nombre=Ana López, puesto=Gerente, salario=66000.0}
Empleado{id=102, nombre=Carlos Pérez, puesto=Analista, salario=50000.0}
Empleado{id=3, nombre=Lucía Gómez, puesto=Programadora, salario=34500.0}
Empleado{id=4, nombre=Jorge Torres, puesto=Diseñador, salario=32000.0}

Total de empleados creados: 4
BUILD SUCCESSFUL (total time: 0 seconds)
```