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.

```java
public class Empleado {  7 usages
    private int id;  4 usages
    private String nombre;  5 usages
    private String puesto;  5 usages
    private double salario;  8 usages
    private static int totalEmpleados = 0;  3 usages

    private static int siguienteId = 1;  3 usages
    private static final double SALARIO_BASE = 100000;  1 usage

    public Empleado(int id, String nombre, String puesto, double salario){  1 usage
        this.id = id;
        this.nombre = nombre;
        this.puesto = puesto;
        this.salario = salario;
        totalEmpleados++;
        if (id >= siguienteId){
            siguienteId = id + 1;
        }

    }
    public Empleado(String nombre, String puesto){  2 usages
        this.id = siguienteId;
        this.nombre = nombre;
        this.puesto = puesto;
        this.salario = SALARIO_BASE;
        totalEmpleados++;
    }

    public void actualizarSalario(double porcentaje){  no usages
        this.salario += this.salario * (porcentaje / 100);
    }

    public void actualizarSalario(int cantidadFija){  3 usages
        this.salario += cantidadFija;
    }

    @Override
    public String toString(){
        return String.format("Empleado{id=%d, nombre='%s', puesto='%s', salario=%.2f}", id, nombre, puesto, salario);
    }

    public int getId() { return id; }  no usages
    public String getNombre() { return nombre; }  no usages
    public void setNombre(String nombre) { this.nombre = nombre; }  no usages
    public String getPuesto() { return puesto; }  no usages
    public void setPuesto(String puesto) { this.puesto = puesto; }  no usages
    public double getSalario() { return salario; }  no usages
    public void setSalario(double salario) { this.salario = salario; }  no usages

    @Contract(pure = true)
    public static int mostrarTotalEmpleados() {  1 usage
        return totalEmpleados;
    }

}
```

```java
public class Gestion_Empleados {
    public static void  main(String[]args){

        Empleado e1 = new Empleado( id: 100,  nombre: "Ana Pérez",  puesto: "Desarrolladora",  salario: 500000);
        Empleado e2 = new Empleado( nombre: "Luis Gómez",  puesto: "QA");
        Empleado e3 = new Empleado( nombre: "Sofía Díaz",  puesto: "Soporte");

        e1.actualizarSalario( cantidadFija: 10);
        e2.actualizarSalario( cantidadFija: 20000);
        e3.actualizarSalario( cantidadFija: 5);

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

        System.out.println("Total empleados: " + Empleado.mostrarTotalEmpleados());
    }
}
```

https://github.com/fabian24cf/Programacion2.git