Programación II - TP4: POO

Alumno: *Romero, Abel Tomás (Comisión 5)*
Link del repo de GitHub:
https://github.com/Tomu98/UTN-TUPaD-P2-TPs/tree/main/04%20POO

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.

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.

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()**`.

Clase `Empleado`:

```java
public class Empleado {

    // Atributos
    private int id;
    private String nombre;
    private String puesto;
    private double salario;
    private static int totalEmpleados = 0;
```

```java
    // Constructor que recibe todos los atributos como parámetros
    public Empleado(int id, String nombre, String puesto, double salario) {
        this.id = id;
        this.nombre = (nombre != null && !nombre.isEmpty()) ? nombre : "Fulanito";
        this.puesto = (puesto != null && !puesto.isEmpty()) ? puesto : "Desarrollador";
        this.salario = (salario >= 0) ? salario : 0.0;
        totalEmpleados++;
    }

    // Constructor que recibe solo nombre y puesto, asignando un id y salario por defecto
    public Empleado(String nombre, String puesto) {
        totalEmpleados++;
        this.id = totalEmpleados;   // ID automático
        this.nombre = (nombre != null && !nombre.isEmpty()) ? nombre : "Fulanito";
        this.puesto = (puesto != null && !puesto.isEmpty()) ? puesto : "Desarrollador";
        this.salario = 1000.0;
    }
```

```java
    // Métodos sobrecargados
    public double actualizarSalario(int porcentaje) {
        if (porcentaje > 0) {
            salario += salario * porcentaje / 100.0;
        }
        return salario;
    }

    public double actualizarSalario(double cantidadFija) {
        if (cantidadFija > 0) {
            salario += cantidadFija;
        }
        return salario;
    }

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

    // Método estático
    public static void mostrarTotalEmpleados() {
        System.out.println("Total de empleados: " + totalEmpleados + "\n");
    }
```

Clase `Main` para pruebas:

```java
public class Main {

    public static void main(String[] args) {

        // Inicio e instancias de la clase Empleado
        System.out.println("--- Sistema de Empleados ---\n");
        Empleado empl = new Empleado(12, "Abel", "Developer", 2500.0);
        Empleado emp2 = new Empleado("Cele", "Designer");
        Empleado emp3 = new Empleado(24, "Jere", "Tester", 2000.0);

        // Mostrar empleados creados y la cantidad total
        System.out.println("Empleados creados:");
        System.out.println(empl);
        System.out.println(emp2);
        System.out.println(emp3);
        Empleado.mostrarTotalEmpleados();

        // Probar métodos sobrecargados
        emp2.actualizarSalario(-20);     // ERROR: Porcentaje negativo
        emp2.actualizarSalario(20);      // Tipo de dato int: Aumenta un 20%
        emp3.actualizarSalario(-500.0);  // ERROR: Cantidad fija negativa
        emp3.actualizarSalario(500.0);   // Tipo de dato double: Aumenta $500
        System.out.println("Luego de aumentos:");
        System.out.println("- Aumento del 20% para Cele: " + emp2);
        System.out.println("- Aumento de $500.0 para Jere: " + emp3);

```

Output en consola:

```
run:
--- Sistema de Empleados ---

Empleados creados:
-> Empleado{id=12, nombre=Abel, puesto=Developer, salario=2500.0}
-> Empleado{id=2, nombre=Cele, puesto=Designer, salario=1000.0}
-> Empleado{id=24, nombre=Jere, puesto=Tester, salario=2000.0}
Total de empleados: 3

Luego de aumentos:
- Aumento del 20% para Cele: -> Empleado{id=2, nombre=Cele, puesto=Designer, salario=1200.0}
- Aumento de $500.0 para Jere: -> Empleado{id=24, nombre=Jere, puesto=Tester, salario=2500.0}
BUILD SUCCESSFUL (total time: 0 seconds)
```