PROGRAMACIÓN II: PROGRAMACIÓN ORIENTADA A OBJETOS II

TRABAJO PRÁCTICO N°4

Resumen

Este trabajo práctico resume conceptos clave de la programación orientada a objetos en Java: uso de this, constructores y sobrecarga, métodos sobrecargados y toString() para representar objetos. También aborda atributos y métodos estáticos y la importancia del encapsulamiento para proteger los datos internos.

Olima, Nicolás Pablo

nicolima200@gmail.com

Link Repo GitHub

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.
- Asegúrate 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.
- 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.

Clase Principal:

```
package tp4.poo.ii;

public class Principal {

    public static void main(String[] args) {
        System.out.println("Empleados: "+Empleado.mostrarTotalEmpleados()); // Verificamos el total de empleados
        System.out.println("--------------------------------------------------------------------------");

        Empleado emp1 = new Empleado(58,"Jorge","Presidente",2000); //Creamos una instancia de la clase Empleado con 4 argumentos
        System.out.println(emp1.toString()); // Mostramos la instancia 'emp1'
        emp1.actualizarSalario(4150); // Aumentamos en una suma fija el salario con el método actualizarSalario()
        System.out.println(emp1); // Mostramos de nuevo la info de la instancia
        System.out.println("Empleados: "+Empleado.mostrarTotalEmpleados()); // Verificamos el total de empleados

        System.out.println("--------------------------------------------------------------------------");

        Empleado emp2 = new Empleado("Ariel","CEO"); // Utilizamos el constructor con 2 argumentos (nombre y puesto)
        System.out.println(emp2);
        emp2.actualizarSalario(1);
        System.out.println(emp2);
        System.out.println("Empleados: "+Empleado.mostrarTotalEmpleados());

        System.out.println("--------------------------------------------------------------------------");

        Empleado emp3 = new Empleado(60, "Pepe","Chofer", 1000);
        System.out.println(emp3);
        emp3.actualizarSalario(50.00);// Aumentamos en un porcentaje el salario con el método actualizarSalario()
        System.out.println(emp3);
        System.out.println("Empleados: "+Empleado.mostrarTotalEmpleados());

    }

}
```

Clase Empleado

```
package tp4.poo.ii;
public 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) {
        totalEmpleados++;
        this.id = id;
        this.nombre = nombre;
        this.puesto = puesto;
        this.salario = salario;
    }

    public Empleado(String nombre, String puesto){

        this(9999, nombre, puesto, 2500.00);
    }

    public void actualizarSalario(double porcentaje){
        if (porcentaje>0){
            this.salario+=salario*(porcentaje/100);
        }
    }

    public void actualizarSalario(int aumento){
        if (aumento>0){
            this.salario+=aumento;
        }
    }
```

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

    public static int mostrarTotalEmpleados(){
        return totalEmpleados;
    }

}
```

Salida por consola: