UNIVERSIDAD TECNOLOGICA NACIONAL

3-9-2025

POO

Trabajo Práctico 4

Alumno:

- Ezequiel Alejandro Ventura

Materia: Programación II

Profesor: Ariel Enferrel

Tutor: Tomás Ferro

GitHub:

https://github.com/equimdq/Programacion2

Ezequiel Ventura
UTN

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

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 total

Empleados: 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().

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.

Resolución del ejercicio

Dificultad encontrada: Duplicación de ID´s

Al usar el constructor principal con ID manual junto con los constructores con ID´s automáticos, se generaban ID´ss duplicados. Esto ocurría porque el contador automático (contadorID) no se actualizaba al crear empleados con ID manual.

Solución encontrada:

Se implementó una lógica de sincronización dentro del constructor principal:

```
if (id >= contadorID) {
    contadorID = id + 1;
```

Esto garantiza que el próximo ID automático sea mayor al último ID manual asignado, evitando duplicaciones y manteniendo la lógica del programa.

Clase Empleado

Atributos y constructor principal

```java
package tp_4_poo;

public class Empleado {

    private int id;
    private String nombre;
    private String puesto;
    private double salario;

    private static int contadorID = 1;
    private static int totalEmpleados = 0;

    // Constructor principal con todos los atributos
    public Empleado(int id, String nombre, String puesto, double salario) {
        this.id = id;
        this.nombre = nombre;
        this.puesto = puesto;
        this.salario = salario;
        totalEmpleados++;

        // Sincroniza el contador automático si el ID manual es mayor para validar que no se repitan
        if (id >= contadorID) {
            contadorID = id + 1;
        }
    }
```

Métodos sobrecargados

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

public void actualizarSalario(int aumento) {
    this.salario += aumento;
}

public static int mostrarTotalEmpleados() {
    return totalEmpleados;
}

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

Main Empleado

```java
package tp_4_poo;



    public class MainEmpleado {

    public static void main(String[] args) {

        // Empleados creados con el constructor principal
        Empleado e1 = new Empleado(1, "María Gallardo", "Analista", 1550);
        // Asignamos ID5 para validar el funcionamiento del ID automático
        Empleado e2 = new Empleado(5, "Luis González", "Arquitecto de redes", 1200);

        // Empleados creados con constructor sobrecargado (ID automático)
        Empleado e3 = new Empleado("Ana Miranda", "Soporte Técnico");
        Empleado e4 = new Empleado("Carlos Armoa", "IT", 1000);
        Empleado e5 = new Empleado("Valeria Ochoa", "Diseñadora", 1500);

        // Aplicar aumentos
        e1.actualizarSalario(10.0); // Usamos un valor decimal (double) para invocar el método que aplica aumento porcentual
        e2.actualizarSalario(300);  // Método fijo
        e3.actualizarSalario(15.0);
        e4.actualizarSalario(200);
        e5.actualizarSalario(20.0);

        // Mostrar empleados
        System.out.println(e1);
        System.out.println(e2);
        System.out.println(e3);
        System.out.println(e4);
        System.out.println(e5);

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

```
run:
Empleado{id=1, nombre='Mar□a Gallardo', puesto='Analista', salario=1705.0}
Empleado{id=5, nombre='Luis Gonz□lez', puesto='Arquitecto de redes', salario=1500.0}
Empleado{id=6, nombre='Ana Miranda', puesto='Soporte T□cnico', salario=1150.0}
Empleado{id=7, nombre='Carlos Armoa', puesto='IT', salario=1200.0}
Empleado{id=8, nombre='Valeria Ochoa', puesto='Dise□adora', salario=1800.0}
Total de empleados creados: 5
BUILD SUCCESSFUL (total time: 0 seconds)
```