Trabajo Práctico 4: Programación orientada a objetos II.
Materia: Programación 2
Estudiante: Etchegoyen Gabriel
Enlace a repositorio en GitHub: https://github.com/Gabriel071185/UTN-TUPaD-P2

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
Uso de this	Referencia a la instancia actual dentro de constructores y métodos

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().

Respuesta:

Solución en repositorio de GitHub: https://github.com/Gabriel071185/UTN-TUPaD-P2