TECNICATURA UNIVERSITARIA
EN PROGRAMACIÓN
A DISTANCIA

PROGRAMACIÓN II

Trabajo Práctico 4: Programación Orientada a Objetos II

Alumno: Federico Iacono

Comisión 6

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

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()**.

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.
● 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.

TECNICATURA UNIVERSITARIA
EN PROGRAMACIÓN
A DISTANCIA

Link a repositorio: Link

Planificación y Consideraciones

Lo primero que hago es definir la clase Empleado con los atributos solicitados: id, nombre, puesto, y salario. Para el conteo global de empleados, declarar el atributo totalEmpleados como static.

En los constructores, utilizo la palabra clave this para diferenciar entre los parámetros de entrada y los atributos de la clase. Esto asegura que la asignación de valores sea correcta y evita conflictos de nombres.

Implementar dos constructores:

El primero recibirá todos los datos (id, nombre, puesto, salario).

El segundo, sobrecargado, solo aceptará el nombre y el puesto. Dentro de este constructor, asignaré un id automático (incrementando totalEmpleados antes de la asignación) y un salario por defecto. Ambos constructores incrementan el contador estático totalEmpleados para mantener un registro preciso.

Métodos sobrecargados actualizarSalario: Definir dos versiones de este método.

Una versión recibirá un double (el porcentaje) y recalculará el salario con un aumento porcentual.

La segunda versión aceptará un int (la cantidad fija) y simplemente sumará ese valor al salario actual.

Método toString(): Sobrescribir el método toString() para que devuelva una cadena de texto legible con la información del empleado (id, nombre, puesto, salario). Esto será muy útil para la depuración y para mostrar los datos en la consola.

Método estático mostrarTotalEmpleados(): Crear un método static que retornará el valor del atributo totalEmpleados. Como es un método estático, lo llamaré directamente desde la clase Main sin necesidad de crear una instancia de Empleado primero.

Clase de Prueba Main: Finalmente en la clase Main creo el método main para ejecutar mi código. Aquí instancio varios objetos Empleado utilizando los dos constructores. Luego aplico los dos métodos actualizarSalario a diferentes empleados para demostrar la sobrecarga. Por último imprimir la información de cada empleado con el método toString() y mostrar el recuento total de empleados usando el método estático.