PROGRAMACION II.
Modulo 4: POO.

Trabajo Práctico modulo 4: POO.

Alumno: LEPKA AGUSTIN
Comisión: 13

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().

Código resultante para la tarea a realizar:

Clase main (PruebaEmpleado):

```
package pruebaempleado;


public class PruebaEmpleado {


    public static void main(String[] args) {


        //se crean los empleados con ambos constructores
        Empleado empleado1 = new Empleado(509, "Perez", "Analista",
70000);
        Empleado empleado2 = new Empleado("Lopez", "Programador");
//id automático y salario por defecto
        Empleado empleado3 = new Empleado("Diaz", "Diseñadora");

        //se llama los metodos para actualizar salario
        empleado1.actualizarSalario(25);  //aumento del 25%
        empleado2.actualizarSalario(5000, true);  //aumento de 100 000
fijo
        empleado3.actualizarSalario(7); //aumento del 7%

        //se muestra la informacion de cada objeto
        System.out.println(empleado1.toString());
        System.out.println(empleado2.toString());
        System.out.println(empleado3.toString());

        //se imprime la cantidad total de empleados
        System.out.println("El total de empleados es: " +
Empleado.mostrarTotalEmpleados());




    }

}
```

Clase Empleado:

```
package pruebaempleado;


public class Empleado {

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

    //atributo static con contador
    private static int totalEmpleados = 0;

    //constructor 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++; //incremento del contador
    }

    //constructor sobrecargado (solo nombre y puesto)
    public Empleado(String nombre, String puesto) {
        this.id = totalEmpleados + 1; //id generado automatico
        this.nombre = nombre;
        this.puesto = puesto;
        this.salario = 50000.0; //salario default
        totalEmpleados++;//incremento del contador
    }

    //sobrecarga de metodos para actualizar salario

    //aumento de salario por porcentaje
    public void actualizarSalario(double porcentaje) {

        this.salario += this.salario * porcentaje / 100.0;
    }

    //aumento de salario por cantidad fija
    public void actualizarSalario(double cantidad, boolean esFijo) {
        if (esFijo) {

            this.salario += cantidad;
        }

        else

        {
            actualizarSalario(cantidad);//llama al aumento por
porcentaje si es false
        }
    }

    //getters y setters

    //devuelve el id
    public int getId() {

        return id;
    }

    //devuelve el nombre
    public String getNombre() {

        return nombre;
    }


    //establece el nombre
    public void setNombre(String nombre) {

        this.nombre = nombre;
    }
```

```java
    //devuelve el puesto
    public String getPuesto() {

        return puesto;
    }

    //establece el puesto
    public void setPuesto(String puesto) {

        this.puesto = puesto;
    }

    //devuelve el salario
    public double getSalario() {

        return salario;
    }

    //establece el salario
    public void setSalario(double salario) {

        this.salario = salario;
    }

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

    //metodo static que devuelve la cantidad de empleados
    public static int mostrarTotalEmpleados() {
        return totalEmpleados;
    }


}
```