CASO PRÁCTICO 4
PROGRAMACIÓN II

Profesor/es: Ramiro Huelpa
Alumno/a: Lauk, Karen

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:
    a. Utilizar this en los constructores para distinguir parámetros de atributos.
2. Constructores sobrecargados:
    a. Uno que reciba todos los atributos como parámetros.
    b. Otro que reciba solo nombre y puesto, asignando un id automático y un salario por defecto.
    c. Ambos deben incrementar totalEmpleados.
3. Métodos sobrecargados actualizarSalario:
    a. Uno que reciba un porcentaje de aumento.
    b. Otro que reciba una cantidad fija a aumentar.
4. Método toString():
    a. Mostrar id, nombre, puesto y salario de forma legible.
5. Método estático mostrarTotalEmpleados():
    a. Retornar el total de empleados creados hasta el momento.
6. Encapsulamiento en los atributos:
    a. Restringir el acceso directo a los atributos de la clase.
    b. 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:
    a. Instancie varios objetos usando ambos constructores.
    b. Aplique los métodos actualizarSalario() sobre distintos empleados.
    c. Imprima la información de cada empleado con toString().
    d. Muestre el total de empleados creados con mostrarTotalEmpleados().

Se modeló un sistema de gestión de empleados. Para ello, se creó una clase **Empleado** con atributos privados y encapsulados, constructores sobrecargados, métodos para actualizar el salario de dos formas diferentes, y un método estático para contabilizar el total de empleados creados.

Archivos:

1. **Clase Empleado:**

```
package casopractico4;

public class Empleados {
```

Clase que define la lógica y estructura.

- Atributos privados (encapsulamiento)

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

Protegemos los datos internos para que no puedan ser modificados directamente desde fuera de la clase. Evita alterar el estado de un objeto sin control.

- Atributo estático (compartido por todos los objetos)

```
private static int totalEmpleados = 0;
```

Este atributo pertenece a la clase en general, no a un objeto específico. Sirve de contador global de empleados registrados. Todos los objetos comparten este mismo valor.

- Constructor que recibe todos los atributos

```
public Empleados(int id, String nombre, String puesto, double salario) {
    this.id = id; // uso de this
    this.nombre = nombre;
    this.puesto = puesto;
    this.salario = salario;
    totalEmpleados++; // aumentar contador
}
```

Acceso controlado y seguro. Podemos observar el uso de this, para distinguir entre variables de instancia y parámetros.Y actualiza el contador de empleados.

- Constructor sobrecargado: solo recibe nombre y puesto

```
public Empleados(String nombre, String puesto) {
    this.id = totalEmpleados + 1; // asignación automática de id
    this.nombre = nombre;
    this.puesto = puesto;
    this.salario = 30000; // salario por defecto
    totalEmpleados++;
}
```

Sobrecarga = mismo nombre del constructor, distinta lista de parámetros. Esto permite crear empleados con menos datos.

➢ Métodos sobrecargados para actualizar salario

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

Se aplica dos formas de actualizar salario: usando porcentaje y usando un monto fijo.

➢ Getter y Setter de nombre

```
public String getNombre() {
    return nombre;
}
public void setNombre(String nombre) {
    this.nombre = nombre;
}
```

Acceso controlado al atributo nombre.

➢ Getter y Setter de salario

```
public void setSalario(double salario) {
    this.salario = salario;
}
```

Acceso controlado al atributo salario.

➢ Método toString para mostrar la información

```
@Override
public String toString() {
    return "Empleado [ID = " + id + ", Nombre = " + nombre +
           ", Puesto = " + puesto + ", Salario = " + salario + "]";
}
```

El método toString sirve para mostrar la información del objeto de forma legible.

➢ Método estático

```
public static int mostrarTotalEmpleados() {
    return totalEmpleados;
}
```

Se utiliza el modificador static en el método mostrarTotalEmpleados() para poder acceder al contador de empleados desde la clase directamente, sin necesidad de instanciar un objeto.

2. CasoPractico4

Contiene la clase principal con el método **main**. Se usa como ejecutor.

```
package casopractico4;

public class CasoPractico4 {

    public static void main(String[] args) {
```

➢ Creación de tres empleados como ejemplo

```
public static void main(String[] args) {
    Empleados empl = new Empleados(1, "Carlos Lopez", "Gerente", 80000);
    Empleados emp2 = new Empleados("Karen Lauk", "Analista");
    Empleados emp3 = new Empleados("Pedro Ruiz", "Desarrollador");
```

- emp1 = lleva el constructor completo (todos los atributos)
- emp2 y emp3 = tiene el constructor sobrecargado (solo nombre y puesto, salario por defecto)

➢ Métodos sobrecargados actualizarSalario

```
emp2.actualizarSalario(10);  // aumento del 10%
emp3.actualizarSalario(5000); // aumento fijo de 5000
```

- emp2 se le sube el salario un 10%
- emp3 se le sube un aumento fijo de 5000

➢ Mostrar información de empleados

```
System.out.println(empl.toString());
System.out.println(emp2.toString());
System.out.println(emp3.toString());
```

Se imprime la información de cada empleado usando el método toString(), que devuelve un texto descriptivo con el id, nombre, apellido, puesto y salario de los empleados.

➢ Mostrar total de empleados

```
System.out.println("Total de empleados creados: " +
                   Empleados.mostrarTotalEmpleados());
```

Llamamos al método static para saber cuántos empleados se han creado en total.

Resultado:

```
run:
Empleado [ID = 1, Nombre = Carlos Lopez, Puesto = Gerente, Salario = 80000.0]
Empleado [ID = 2, Nombre = Karen Lauk, Puesto = Analista, Salario = 30010.0]
Empleado [ID = 3, Nombre = Pedro Ruiz, Puesto = Desarrollador, Salario = 35000.0]
Total de empleados creados: 3
BUILD SUCCESSFUL (total time: 0 seconds)
```

Link de repositorio GitHub:
https://github.com/karenlauk/Programaci-n2