PROGRAMACIÓN II

Trabajo Práctico 4: Programación Orientada a Objetos II

ALUMNO:
Franco Sarrú
Link público de GitHub:
https://github.com/fsarru/Programacion2.git

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()**.

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.
- Aplicar encapsulamiento en los atributos.
- 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.

Sarru_Unidad4.java × Empleado.java ×

Source History

```java
/*● 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.*/

package sarru_unidad4;


public class Empleado {


//Atributos - Privados
private int id; //Identificador único del empleado.
private String nombre; //Nombre completo.
private String puesto; //Cargo que desempeña.
private double salario; //Salario actual.

//Atributos Estáticos
private static int totalEmpleados = 0; //Contador global de empleados creados.
private static int ultimoId = 0;

// Sobrecarga de constructores

// 1er constructor - 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++;
        if (id > ultimoId) {
            ultimoId = id; // Para no repetir ID
        }

        }

 // 2do constructor - atributo nomre y puesto. Salario predeterminado
 public Empleado(String nombre, String puesto) {
     this.id = ++ultimoId;
     this.nombre = nombre;
     this.puesto = puesto;
     this.salario = 1000.00;
     totalEmpleados++;
```

```java
    // Método sobrecargado para actulizar salario por un porcentaje
    public void actualizarSalario(double porcentaje) {
        this.salario += this.salario * (porcentaje / 100);
    }

    // Método sobrecargado para actulizar salario por una suma fija
    public void actualizarSalario(int aumento) {
        this.salario += aumento;
    }


    //Getters y setters
    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getNombre() {
        return nombre;
    }

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

    public String getPuesto() {
        return puesto;
    }

    public void setPuesto(String puesto) {
        this.puesto = puesto;
    }

    public double getSalario() {
        return salario;
    }

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

```java
    public static int getTotalEmpleados() {
        return totalEmpleados;
    }

    public static void setTotalEmpleados(int totalEmpleados) {
        Empleado.totalEmpleados = totalEmpleados;
    }


    // Mostar total de empleados creados
    public static int mostrarTotalEmpleados() {
        return totalEmpleados;
    }

    // toString

    @Override
    public String toString() {
        return "El empleado a consultar es {" + "id=" + id + ", nombre=" + nombre + ", puesto=" + puesto + ", salario=" + salario + '}';
    }


}
```

Sarru_Unidad4.java × Empleado.java ×

Source History

```java
package sarru_unidad4;


public class Sarru_Unidad4 {

    public static void main(String[] args) {
        //creación de empleados con constructores

        //Constructor 1:
        Empleado empl = new Empleado(1, "Carlos Perez", "Gerente", 1800);

        // Validación constante de los empleados creados
        System.out.println("Total de empleados: " + Empleado.mostrarTotalEmpleados());

        //Constructor 2:
        Empleado emp2 = new Empleado("Martina Gonzalez", "Jefe");

        // Validación constante de los empleados creados
        System.out.println("Total de empleados: " + Empleado.mostrarTotalEmpleados());

         Empleado emp3 = new Empleado("Rafael Gomez", "Operador");



    // toString de cada empleado

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

    // Actualizar salarios con métodos sobrecargados
    empl.actualizarSalario(200);
    emp2.actualizarSalario(30.0);
    emp3.actualizarSalario(100);

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

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

```
41
42      System.out.println("Total de empleados: " + Empleado.mostrarTotalEmpleados());}
43      }
44
45
```

sarru_unidad4.Sarru_Unidad4 > main >

Output ×

franc - C:\Users\franc × | Sarru_Unidad4 (run) ×

```
run:
Total de empleados: 1
Total de empleados: 2
El empleado a consultar es {id=1, nombre=Carlos Perez, puesto=Gerente, salario=1800.0}
El empleado a consultar es {id=2, nombre=Martina Gonzalez, puesto=Jefe, salario=1000.0}
El empleado a consultar es {id=3, nombre=Rafael Gomez, puesto=Operador, salario=1000.0}
El empleado a consultar es {id=1, nombre=Carlos Perez, puesto=Gerente, salario=2000.0}
El empleado a consultar es {id=2, nombre=Martina Gonzalez, puesto=Jefe, salario=1300.0}
El empleado a consultar es {id=3, nombre=Rafael Gomez, puesto=Operador, salario=1100.0}
Total de empleados: 3
BUILD SUCCESSFUL (total time: 0 seconds)
```