UNIVERSIDAD TECNOLOGICA NACIONAL

Trabajo Práctico 4

Programación Orientada a Objetos II

Programación II

Alumno

Alex Pedro Dauria

Fecha de Entrega

17 de Septiembre de 2025

Índice

Link al Repositorio en GitHub
https://github.com/Alex-Dauria/UTN-TUPaD-P2/tree/main/04%20POO

Ejercicio 1: Requerimientos

1. Uso de this

Se utilizó this en los constructores para referenciar los atributos de la instancia actual y evitar ambigüedades con los parámetros.

2. Constructores sobrecargados

Se implementaron dos constructores: uno que recibe todos los atributos y otro que recibe solo nombre y puesto, asignando un id automático y salario por defecto. Ambos incrementan el contador estático totalEmpleados.

```java
// Constructor completo
public Empleado(int id, String nombre, String puesto, double salario) {
    this.id = id;
    this.nombre = nombre;
    this.puesto = puesto;
    this.salario = salario;
    totalEmpleados++;
}

// Constructor simplificado
public Empleado(String nombre, String puesto) {
    this.id = totalEmpleados + 1;
    this.nombre = nombre;
    this.puesto = puesto;
    this.salario = 1000.0; // Salario por defecto
    totalEmpleados++;
}
```

3. Métodos sobrecargados actualizarSalario

Se implementaron dos versiones del método actualizarSalario: una que recibe un porcentaje de aumento y otra que recibe una cantidad fija.

```java
// Métodos sobrecargados para actualizar salario
public void actualizarSalario(double porcentaje) {
    this.salario += this.salario * (porcentaje / 100);
}

public void actualizarSalario(int aumentoFijo) {
    this.salario += aumentoFijo;
}
```

4. Método toString()

Se sobreescribió el método toString() para devolver una representación legible del estado del empleado.

```
// toString
@Override
public String toString() {
    return String.format("ID: %d | Nombre: %s | Puesto: %s | Salario: $%.2f", id, nombre, puesto, salario);
}
```

5. Método estático mostrarTotalEmpleados()

Se implementó un método estático que devuelve el total de empleados creados, usando el atributo estático totalEmpleados.

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

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

6. Encapsulamiento

Se encapsularon todos los atributos haciéndolos privados y se proporcionaron métodos getters y setters para acceder y modificar estos atributos de manera controlada.

```
private int id;
private String nombre;
private String puesto;
private double salario;
private static int totalEmpleados = 0;
```

```java
// 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; }
```

Ejercicio 2: Clase de Prueba (main)

```java
public class Main {
    public static void main(String[] args) {
        // Crear empleados
        Empleado emp1 = new Empleado(1, "Alex Dauria", "Programador", 2500.0);
        Empleado emp2 = new Empleado("Nicolas Gonzalez", "Ingeniero");
        Empleado emp3 = new Empleado("Marta Rodriguez", "Gerente");


        // Aplicar aumentos
        emp1.actualizarSalario(10.0); // 10% de aumento
        emp2.actualizarSalario(500);  // Aumento fijo

        // Mostrar información
        System.out.println("=== Empleados ===");
        System.out.println(emp1);
        System.out.println(emp2);
        System.out.println(emp3);

        // Mostrar total de empleados
        System.out.println("\nTotal de empleados: " + Empleado.mostrarTotalEmpleados());
    }
}
```

Salida en consola

```
run:
=== Empleados ===
ID: 1 | Nombre: Alex Dauria | Puesto: Programador | Salario: $2750,00
ID: 2 | Nombre: Nicolas Gonzalez | Puesto: Ingeniero | Salario: $1500,00
ID: 3 | Nombre: Marta Rodriguez | Puesto: Gerente | Salario: $1000,00

Total de empleados: 3
BUILD SUCCESSFUL (total time: 0 seconds)
```

Se implementaron correctamente todos los requisitos: this, constructores y métodos sobrecargados, encapsulamiento, miembro estático y toString(). La sobrecarga distingue tipos: un double para porcentaje (ej. actualizarSalario(10.0) → 10%) y un int para monto fijo (ej. actualizarSalario(500) → +500).