

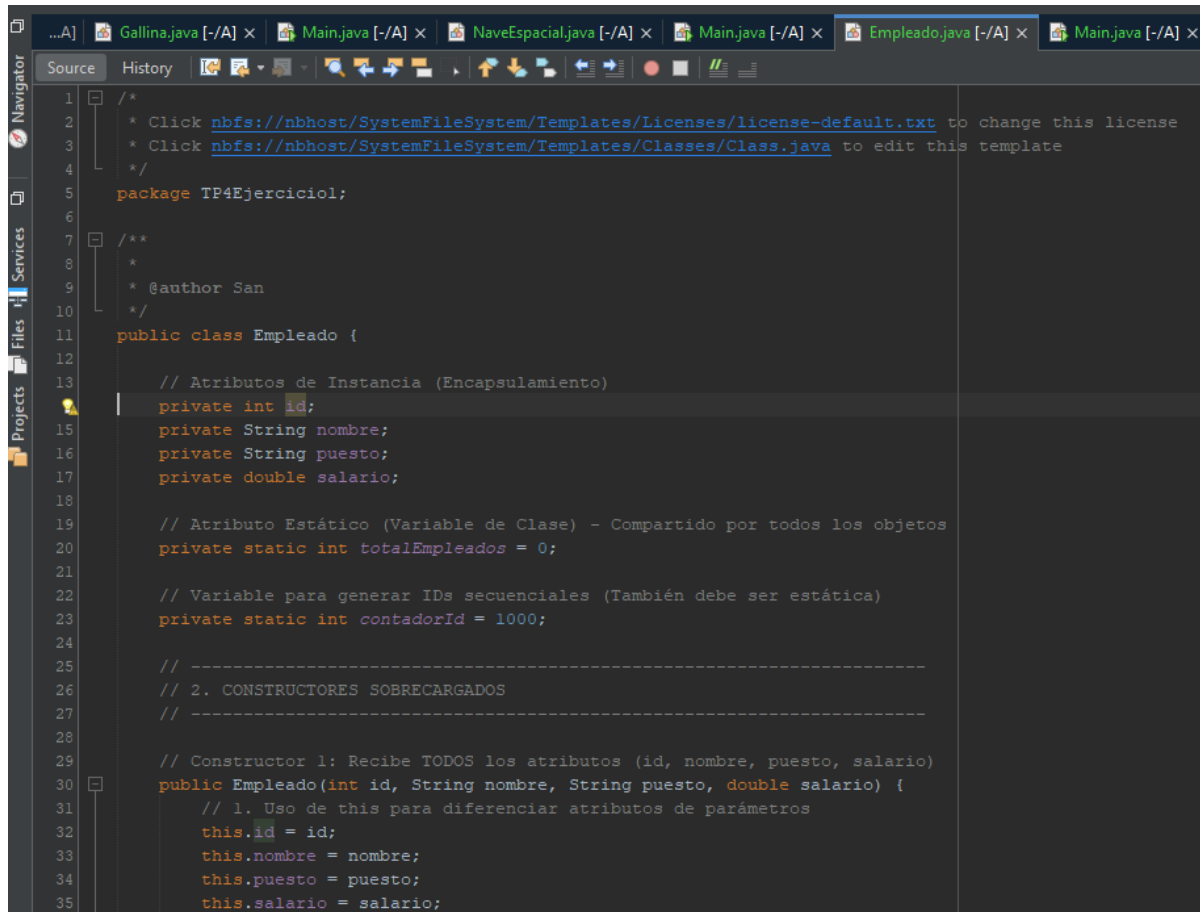
# PROGRAMACIÓN II

## Trabajo Práctico 4: Programación Orientada a Objetos II

Alumno Santiago Raúl Salinas

Caso Práctico:

**Clase: Empleado**



```
1  /*
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3   * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4   */
5   package TP4Ejercicio1;
6
7   /**
8    *
9    * @author San
10   */
11  public class Empleado {
12
13      // Atributos de Instancia (Encapsulamiento)
14      private int id;
15      private String nombre;
16      private String puesto;
17      private double salario;
18
19      // Atributo Estático (Variable de Clase) - Compartido por todos los objetos
20      private static int totalEmpleados = 0;
21
22      // Variable para generar IDs secuenciales (También debe ser estática)
23      private static int contadorId = 1000;
24
25      // -----
26      // 2. CONSTRUCTORES SOBRECARGADOS
27      // -----
28
29      // Constructor 1: Recibe TODOS los atributos (id, nombre, puesto, salario)
30      public Empleado(int id, String nombre, String puesto, double salario) {
31          // 1. Uso de this para diferenciar atributos de parámetros
32          this.id = id;
33          this.nombre = nombre;
34          this.puesto = puesto;
35          this.salario = salario;
```

```

35     this.salario = salario;
36
37     // Incrementa el contador global de empleados
38     totalEmpleados++;
39 }
40
41 // Constructor 2: Recibe solo nombre y puesto, asigna id automático y salario por defecto
42 public Empleado(String nombre, String puesto) {
43     // 1. Uso de this para diferenciar atributos de parámetros
44     // Asignación de ID automático y Salario por defecto
45     this.id = contadorId++; // Asigna el valor actual y luego lo incrementa para el próximo
46     this.nombre = nombre;
47     this.puesto = puesto;
48     this.salario = 35000.00; // Salario por defecto
49
50     // Incrementa el contador global de empleados
51     totalEmpleados++;
52 }
53
54 // -----
55 // 3. MÉTODOS SOBRECARGADOS (actualizarSalario)
56 // -----
57
58 // Método 1: Aumentar salario por porcentaje
59 public void actualizarSalario(double porcentajeAumento) {
60     if (porcentajeAumento > 0) {
61         double aumento = this.salario * (porcentajeAumento / 100);
62         this.salario += aumento;
63         System.out.printf("✓ Salario de %s actualizado. Aumento del %.2f%% (%$.2f)\n",
64             this.nombre, porcentajeAumento, aumento);
65     } else {
66         System.out.println("✗ El porcentaje de aumento debe ser positivo.");
67     }
68 }

```

```

69
70 // Método 2: Aumentar salario por cantidad fija
71 public void actualizarSalario(int cantidadFija) {
72     if (cantidadFija > 0) {
73         this.salario += cantidadFija;
74         System.out.printf("✓ Salario de %s actualizado. Aumento fijo de $%.2f\n",
75             this.nombre, (double)cantidadFija);
76     } else {
77         System.out.println("✗ La cantidad fija debe ser positiva.");
78     }
79 }
80
81 // -----
82 // 4. MÉTODO toString()
83 // -----
84
85 @Override
86 public String toString() {
87     return String.format(
88         "| ID: %d | Nombre: %s | Puesto: %s | Salario: $%.2f |",
89         this.id, this.nombre, this.puesto, this.salario
90     );
91 }
92
93 // -----
94 // 5. MÉTODO ESTÁTICO (mostrarTotalEmpleados)
95 // -----
96
97 // Se declara estático porque opera sobre el atributo estático (totalEmpleados)
98 public static int mostrarTotalEmpleados() {
99     return totalEmpleados;
100 }
101
102 // 6. GETTERS Y SETTERS (Solo los esenciales para el caso práctico)
103
104 public int getId() {
105     return id;
106 }

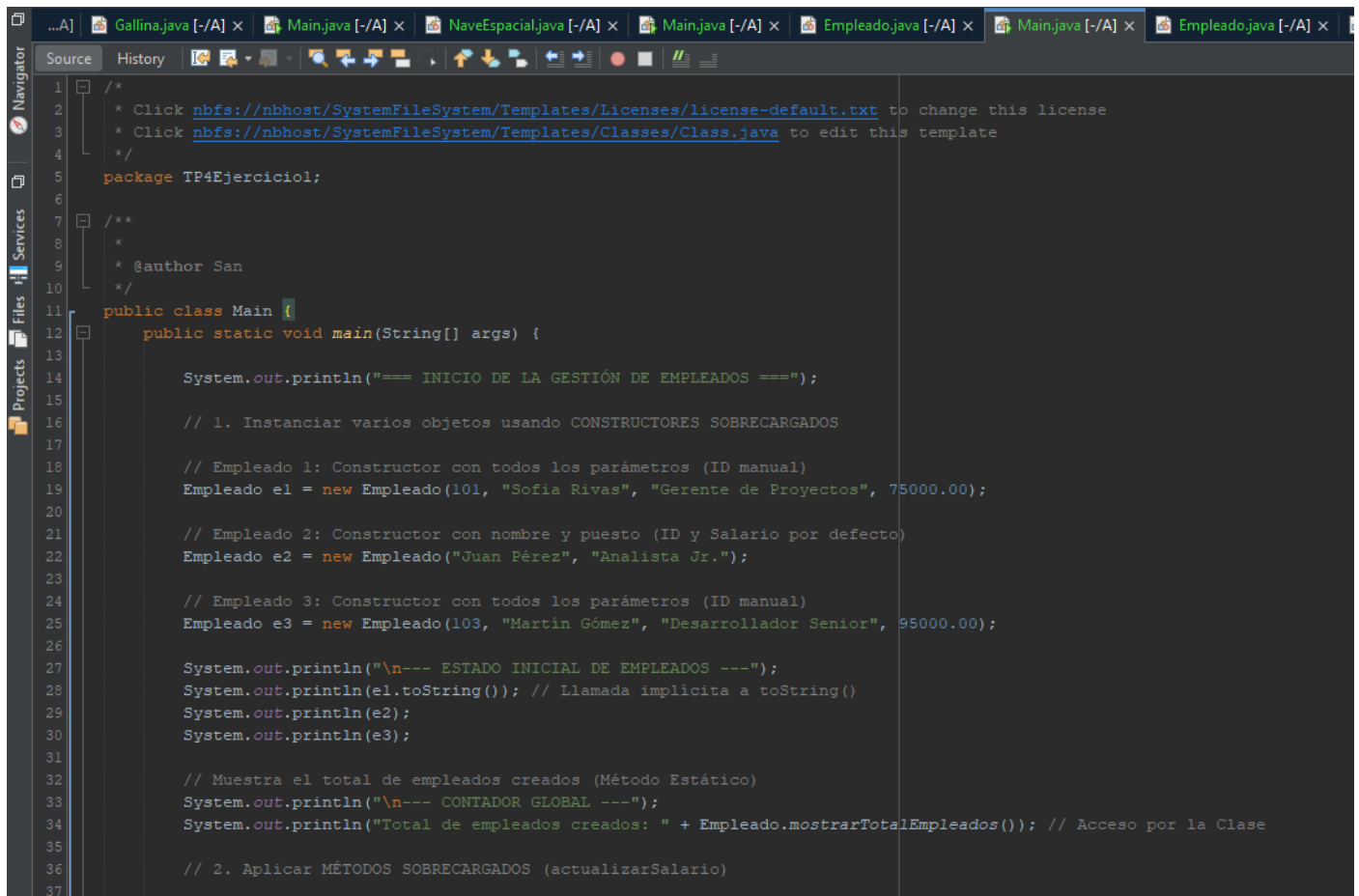
```

```

106     }
107
108     public String getNombre() {
109         return nombre;
110     }
111
112     public void setNombre(String nombre) {
113         this.nombre = nombre;
114     }
115
116     public String getPuesto() {
117         return puesto;
118     }
119
120     public void setPuesto(String puesto) {
121         this.puesto = puesto;
122     }
123
124     public double getSalario() {
125         return salario;
126     }
127 }

```

## Main:



```

1  /*
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4  */
5  package TP4Ejercicio1;
6
7  /**
8   *
9   * @author San
10  */
11  public class Main {
12      public static void main(String[] args) {
13
14          System.out.println("=== INICIO DE LA GESTIÓN DE EMPLEADOS ===");
15
16          // 1. Instanciar varios objetos usando CONSTRUCTORES SOBRECARGADOS
17
18          // Empleado 1: Constructor con todos los parámetros (ID manual)
19          Empleado e1 = new Empleado(101, "Sofia Rivas", "Gerente de Proyectos", 75000.00);
20
21          // Empleado 2: Constructor con nombre y puesto (ID y Salario por defecto)
22          Empleado e2 = new Empleado("Juan Pérez", "Analista Jr.");
23
24          // Empleado 3: Constructor con todos los parámetros (ID manual)
25          Empleado e3 = new Empleado(103, "Martín Gómez", "Desarrollador Senior", 95000.00);
26
27          System.out.println("\n--- ESTADO INICIAL DE EMPLEADOS ---");
28          System.out.println(e1.toString()); // Llamada implícita a toString()
29          System.out.println(e2);
30          System.out.println(e3);
31
32          // Muestra el total de empleados creados (Método Estático)
33          System.out.println("\n--- CONTADOR GLOBAL ---");
34          System.out.println("Total de empleados creados: " + Empleado.mostrarTotalEmpleados()); // Acceso por la Clase
35
36          // 2. Aplicar MÉTODOS SOBRECARGADOS (actualizarSalario)
37

```

```

35
36 // 2. Aplicar MÉTODOS SOBRECARGADOS (actualizarSalario)
37
38 System.out.println("\n--- APLICACIÓN DE AUMENTOS ---");
39
40 // Aumento al Empleado 1 por porcentaje
41 e1.actualizarSalario(5.5);
42
43 // Aumento al Empleado 2 por cantidad fija
44 e2.actualizarSalario(2500);
45
46 // Aumento al Empleado 3 por porcentaje
47 e3.actualizarSalario(10.0);
48
49 // 3. Imprimir el ESTADO FINAL (toString)
50
51 System.out.println("\n--- ESTADO FINAL DE EMPLEADOS ---");
52 System.out.println(e1);
53 System.out.println(e2);
54 System.out.println(e3);
55
56 // Muestra el total de empleados creados (Debe seguir siendo 3)
57 System.out.println("\n--- CONTADOR GLOBAL FINAL ---");
58 System.out.println("Total de empleados creados: " + Empleado.mostrarTotalEmpleados());
59 }
60

```

## Output:

```

run:
=== INICIO DE LA GESTIÓN DE EMPLEADOS ===

--- ESTADO INICIAL DE EMPLEADOS ---
| ID: 101 | Nombre: Sofia Rivas | Puesto: Gerente de Proyectos | Salario: $75000,00 |
| ID: 1000 | Nombre: Juan Pérez | Puesto: Analista Jr. | Salario: $35000,00 |
| ID: 103 | Nombre: Martón Gómez | Puesto: Desarrollador Senior | Salario: $95000,00 |

--- CONTADOR GLOBAL ---
Total de empleados creados: 3

--- APLICACIÓN DE AUMENTOS ---
? Salario de Sofia Rivas actualizado. Aumento del 5,50% ($4125,00)
? Salario de Juan Pérez actualizado. Aumento fijo de $2500,00
? Salario de Martón Gómez actualizado. Aumento del 10,00% ($9500,00)

--- ESTADO FINAL DE EMPLEADOS ---
| ID: 101 | Nombre: Sofia Rivas | Puesto: Gerente de Proyectos | Salario: $79125,00 |
| ID: 1000 | Nombre: Juan Pérez | Puesto: Analista Jr. | Salario: $37500,00 |
| ID: 103 | Nombre: Martón Gómez | Puesto: Desarrollador Senior | Salario: $104500,00 |

--- CONTADOR GLOBAL FINAL ---
Total de empleados creados: 3
BUILD SUCCESSFUL (total time: 0 seconds)

```