Informe Técnico: Sistema de Gestión de Empleados

Luis Carlos Cortez Guzmán

Introducción

El presente informe detalla la creación de un sistema para gestionar diferentes tipos de empleados en una empresa, utilizando la programación orientada a objetos en Java. El sistema maneja empleados de tiempo completo, por horas y temporales, cada uno con características y beneficios específicos.

Estructura del Programa

El programa está compuesto por una clase padre `Empleado` y tres clases hijas: `EmpleadoTiempoCompleto`, `EmpleadoPorHoras` y `EmpleadoTemporal`. Cada clase hija extiende la funcionalidad de la clase padre añadiendo atributos y métodos específicos.

Clase Padre: Empleado

La clase `Empleado` contiene los atributos y métodos comunes a todos los tipos de empleados.

Atributos:

- `nombre` (String): Nombre del empleado.

- `id` (int): Identificador único del empleado.

- `salario` (double): Salario base del empleado.

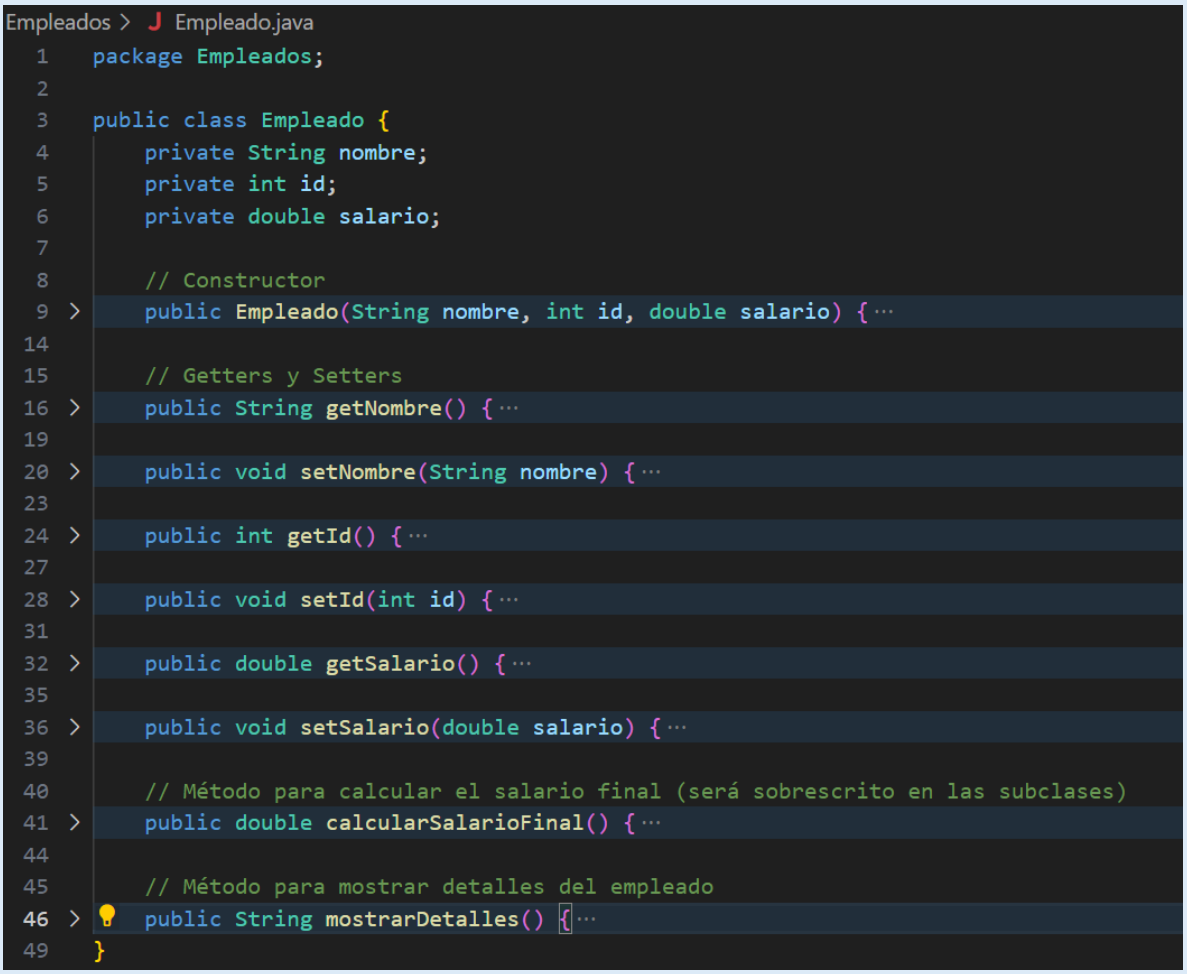
Métodos:

- Constructor: Inicializa los atributos `nombre`, `id` y `salario`.

- Getters y Setters: Métodos para acceder y modificar cada uno de los atributos.

- calcularSalarioFinal(): Método que devuelve el salario final (será sobrescrito en las subclases).

- mostrarDetalles(): Método que devuelve la información básica del empleado.



EmpleadoTiempoCompleto

Atributo adicional:

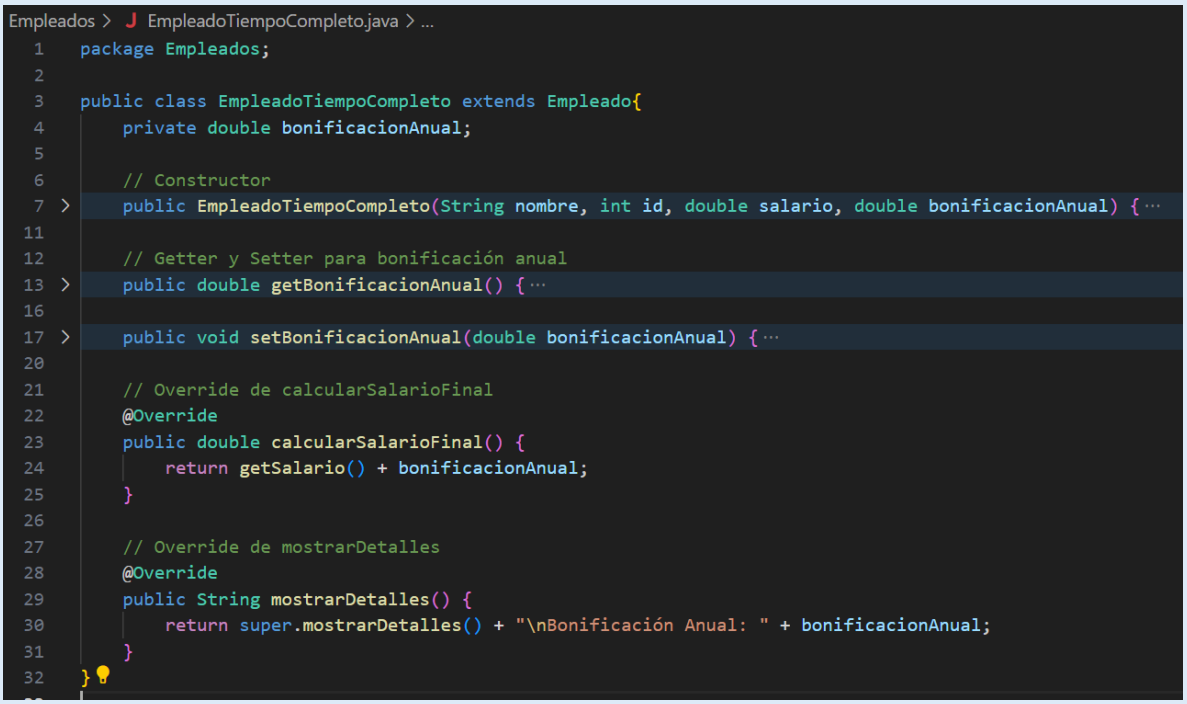
- `bonificacionAnual` (double): Bonificación anual del empleado.

Métodos:

- Constructor: Inicializa los atributos de la clase padre y el atributo `bonificacionAnual`.

- calcularSalarioFinal(): Incluye la bonificación anual en el cálculo del salario final.

- mostrarDetalles(): Incluye la bonificación anual en la información mostrada.



EmpleadoPorHoras

Atributos adicionales:

- `horasTrabajadas` (int): Número de horas trabajadas.

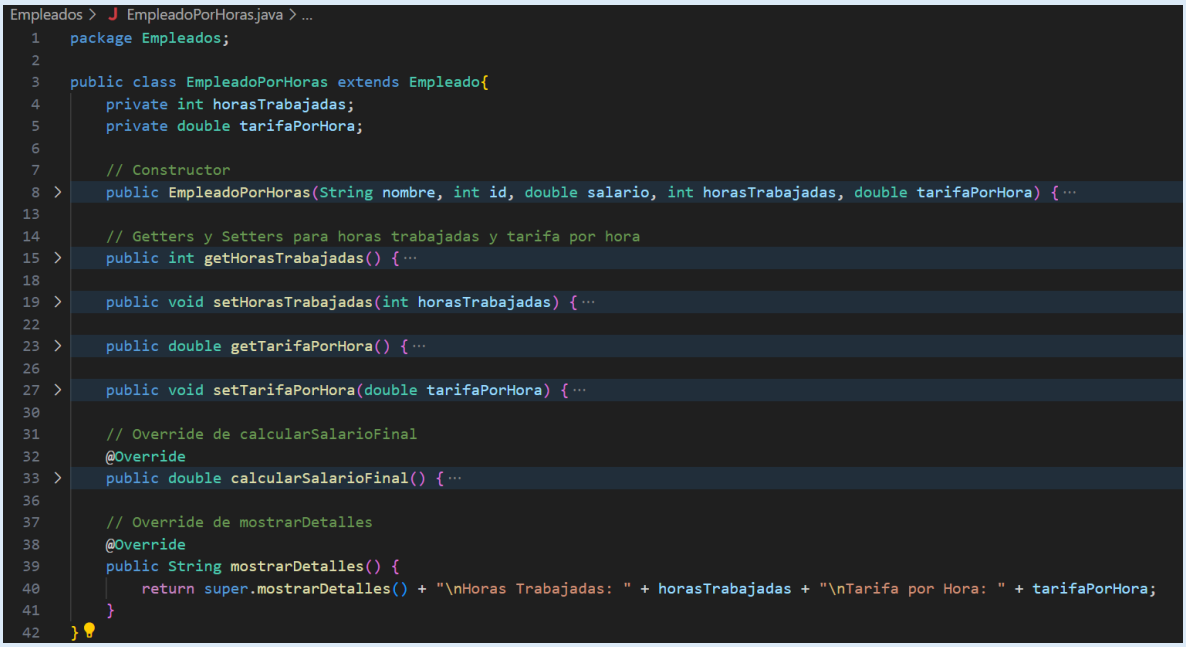
- `tarifaPorHora` (double): Tarifa por hora trabajada.

Métodos:

- Constructor: Inicializa los atributos de la clase padre, `horasTrabajadas` y `tarifaPorHora`.

- calcularSalarioFinal(): Calcula el salario en base a las horas trabajadas y la tarifa por hora.

- mostrarDetalles(): Incluye las horas trabajadas y la tarifa por hora en la información mostrada.



EmpleadoTemporal

Atributos adicionales:

- `fechaInicio` (String): Fecha de inicio del contrato.

- `fechaFin` (String): Fecha de fin del contrato.

Métodos:

- Constructor: Inicializa los atributos de la clase padre, `fechaInicio` y `fechaFin`.

- calcularSalarioFinal(): Considera un bono al finalizar el contrato.

- mostrarDetalles(): Incluye la fecha de inicio y fin en la información mostrada.

Empleados > EmpleadoTemporal.java > ...

```java
package Empleados;

public class EmpleadoTemporal extends Empleado{
    private String fechaInicio;
    private String fechaFin;

    // Constructor
    public EmpleadoTemporal(String nombre, int id, double salario, String fechaInicio, String fechaFin) {...

    // Getters y Setters para fecha de inicio y fin
    public String getFechaInicio() {...

    public void setFechaInicio(String fechaInicio) {...

    public String getFechaFin() {...

    public void setFechaFin(String fechaFin) {...

    // Override de calcularSalarioFinal (considerando un bonus al finalizar el contrato)
    @Override
    public double calcularSalarioFinal() {...

    // Override de mostrarDetalles
    @Override
    public String mostrarDetalles() {...
}
```

Clase Principal Main

La clase `Main` se utiliza para crear instancias de cada tipo de empleado y mostrar sus detalles y salario final.

Main.java > ...

```java
import Empleados.*;
public class Main {
    Run | Debug
    public static void main(String[] args) {
        EmpleadoTiempoCompleto empTiempoCompleto = new EmpleadoTiempoCompleto(nombre:"Luis", id:1, salario:2500, bonificacionAnual:5000);
        EmpleadoPorHoras empPorHoras = new EmpleadoPorHoras(nombre:"Carlos", id:2, salario:0, horasTrabajadas:100, tarifaPor…20);
        EmpleadoTemporal empTemporal = new EmpleadoTemporal(nombre:"Cortez", id:3, salario:2000, fechaInicio:"2024-01-01", fechaFin:"2024-

        System.out.println(empTiempoCompleto.mostrarDetalles());
        System.out.println("Salario Final: " + empTiempoCompleto.calcularSalarioFinal());
        System.out.println();

        System.out.println(empPorHoras.mostrarDetalles());
        System.out.println("Salario Final: " + empPorHoras.calcularSalarioFinal());
        System.out.println();

        System.out.println(empTemporal.mostrarDetalles());
        System.out.println("Salario Final: " + empTemporal.calcularSalarioFinal());
    }
}
```

Explicación de la Ejecución del Programa

1. Instanciación de Objetos:

  - Se crean instancias de `EmpleadoTiempoCompleto`, `EmpleadoPorHoras` y `EmpleadoTemporal` utilizando los constructores definidos en cada clase.

  - Ejemplo: `EmpleadoTiempoCompleto empTiempoCompleto = new EmpleadoTiempoCompleto("Carlos López", 1, 50000, 5000);`

2. Llamada a Métodos:

  - Se llama al método `mostrarDetalles()` para cada instancia creada.

  - Ejemplo: `System.out.println(empTiempoCompleto.mostrarDetalles());`

3. Salida del Programa:

  - El método `mostrarDetalles()` de cada clase hija sobrescribe el método de la clase padre `Empleado` para incluir información específica (bonificaciones, horas trabajadas, tarifas, fechas).

  - Se imprime la información detallada de cada empleado, mostrando el nombre, ID, salario, y los atributos adicionales según el tipo de empleado.

  - También se llama al método `calcularSalarioFinal()` para calcular y mostrar el salario final considerando las bonificaciones y horas trabajadas.

Ejemplo de Salida

```
Nombre: Luis
ID: 1
Salario: 2500.0
Bonificación Anual: 5000.0
Salario Final: 7500.0

Nombre: Carlos
ID: 2
Salario: 0.0
Horas Trabajadas: 100
Tarifa por Hora: 20.0
Salario Final: 2000.0

Nombre: Cortez
ID: 3
Salario: 2000.0
Fecha de Inicio: 2024-01-01
Fecha de Fin: 2024-06-30
Salario Final: 2200.0
```

Conclusión

Este sistema de gestión de empleados permite manejar eficientemente la información de diferentes tipos de empleados, utilizando la herencia y el polimorfismo en Java. La estructura del programa es extensible, lo que facilita la adición de nuevos tipos de empleados en el futuro si es necesario.