PARADIGMAS DE PROGRAMACIÓN

Práctica 01

Kevin Sebastián Frias García

Sep 24, 2024

LIS - 501

Sistema de gestión de empleados

Instrucción

Crear un sistema de gestión de empleados en Java que utilice los cuatro pilares de la POO: encapsulación, herencia, polimorfismo y abstracción.

Clase Empleado

El primer pilar a utilizar es la encapsulación de atributos. De ese forma se asegura la integridad de estos datos al hacer que solo se puedan acceder desde la clase. Se crea el constructor y se crean los métodos para obtener estos datos y modificar de forma segura.

```java
public class Empleado {
    private String nombre;
    private int ID;
    private double salario;

    public Empleado(String nombre, int ID, double salario) {
        this.nombre = nombre;
        this.ID = ID;
        this.salario = salario;
    }

    public String getNombre() {
        return nombre;
    }

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

    public int getID() {
        return ID;
    }

    public void setID(int ID) {
        this.ID = ID;
    }

    public double getSalario() {
        return salario;
    }

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

Herencia

Prosiguiendo, ahora utilizamos el pilar herencia para transmitir desde una clase padre a dos clases hijas que llamaremos: 1- EmpleadoTiempoCompleto y 2- EmpleadoTiempoParcial.

Clase EmpleadoTiempoCompleto

```java
public class EmpleadoTiempoCompleto extends Empleado{
    private int horasTrabajadas;

    public EmpleadoTiempoCompleto(String nombre, int ID,
                                  double salario, int horasTrabajadas) {
        super(nombre, ID, salario);
        this.horasTrabajadas = horasTrabajadas;
    }

    public int getHorasTrabajadas() {
        return horasTrabajadas;
    }

    public void setHorasTrabajadas(int horasTrabajadas) {
        this.horasTrabajadas = horasTrabajadas;
    }

    /* @Override
    public double calcularSalario() {
        return getSalario() + horasTrabajadas;
    }
    */
}
```

Se ha añadido un atributo llamado "horasTrabajadas" de tipo "int" y he optado por hacer la primera clase abstracta para poder hacer un método abstracto para definir un método para calcular el salario mensual en función del salario base y las horas trabajadas.

Clase EmpleadoTiempoParcial

Repetimos el proceso, no obstante, tenemos una variable llamada "horasPorSemana" y modificamos el método de calcular salario en función a este atributo y al salario por hora.

```java
public class EmpleadoTiempoParcial extends Empleado{
    private int horasPorSemana;

    public EmpleadoTiempoParcial(String nombre, int ID, double salario,
                                 int horasPorSemana) {
        super(nombre, ID, salario);
        this.horasPorSemana = horasPorSemana;
    }

    public int getHorasPorSemana() {
        return horasPorSemana;
    }

    public void setHorasPorSemana(int horasPorSemana) {
        this.horasPorSemana = horasPorSemana;
    }

    @Override
    public double calcularSalario() {
        return horasPorSemana * getSalario();
    }
}
```

Polimorfismo

¿Qué es?
Capacidad de un objeto para adoptar distintas formas o cambiar de comportamiento, incluso apariencia pero su implementación de la interfaz es la misma(heredado de la superclase).

Al implementar un método en común con la clase padre "Empleado" obtenemos que ambas clases hijo puedan calcular el salario correspondiente según el tipo de empleado utilizando una misma interfaz y obtener el salario adecuado.

Abstracción

Creamos la interfaz "IEmpleado" para declarar un método sin implementación en las clases hijas donde:

- "EmpleadoTiempoCompleto": 5% sobre el cálculo del salario
- "EmpleadoTiempoParcial": 3% sobre el cálculo del salario

Interfaz IEmpleado

```
public interface IEmpleado {
    double calcularIncentivo();
}
```

Implementación

Para llevar a cabo la implementación de la interfaz IEmpleado es necesario indicar en las clases hija con la palabra reservada "implements" que están heredando métodos u atributos de la interfaz. Luego con la anotación "@Override" definimos que estamos sobre escribiendo un método que se implementa en la interfaz principal de esa forma podemos reutilizar código utilizando principios de la POO.

```
// Clase EmpleadoTiempoParcial
public class EmpleadoTiempoParcial extends Empleado implements IEmpleado
// Clase EmpleadoTiempoCompleto
public class EmpleadoTiempoCompleto extends Empleado implements IEmpleado

// Se implementa el método con las especifícaciones dadas
@Override
    public double calcularIncentivo() {
        return 0.03 * calcularSalario();
    }

 @Override
    public double calcularIncentivo() {
        return 0.05 * calcularSalario();
    }
```

De igual forma sobre escribimos el método para imprimir "toString" de la clase "object" dentro de Java. De esta manera le damos un mejor formato de salida.

```
// Clase EmpleadoTiempoCompleto
@Override
    public String toString() {
        return "El empleado a tiempo completo: " +getNombre()+
        " tiene un salario de: $"+calcularSalario()+"\n"
                + "con un incentivo de $"+calcularIncentivo();
    }
// Clase EmpleadoTiempoParcial
@Override
    public String toString() {
        return "El empleado a tiempo parcial: " +getNombre()+
        " tiene un salario de: $"+calcularSalario()+"\n"
                + "con un incentivo de $"+calcularIncentivo();
    }
```

Clase MainApp

```
import Model.EmpleadoTiempoCompleto;
import Model.EmpleadoTiempoParcial;
```

Importamos las clases hijas del paquete "Model"

```
EmpleadoTiempoCompleto empleadoTiempoCompleto =
        new EmpleadoTiempoCompleto( nombre: "Kevin", ID: 22017021, salario: 267, horasTrabajadas: 40);

EmpleadoTiempoParcial empleadoTiempoParcial =
        new EmpleadoTiempoParcial( nombre: "Miguel", ID: 9234950, salario: 93, horasPorSemana: 26);
```

Dentro del método main, instanciamos las clases del empleado a tiempo completo y a tiempo parcial. Finalmente le pasamos los parámetros al constructor.

```
System.out.printf("""
        (1) Empleado a tiempo completo:
        %s
        (2) Empleado a tiempo parcial:
        %s
        """, empleadoTiempoCompleto,empleadoTiempoParcial);
```

Finalmente, mandamos a imprimir los valores del objeto creado.

Salida

```
(1) Empleado a tiempo completo:
El empleado a tiempo completo: Kevin tiene un salario de: $10680.0
con un incentivo de $534.0
(2) Empleado a tiempo parcial:
El empleado a tiempo parcial: Miguel tiene un salario de: $2418.0
con un incentivo de $73.0
```