

Programación orientada a objetos

Introducción

En muchas ocasiones de la vida cotidiana, nos encontramos con situaciones en las que necesitamos trabajar con conjuntos de datos que son constantes, es decir, valores que no cambian y que ya son conocidos previamente. Estos datos constantes pueden representar opciones, categorías, estados, tipos o cualquier otro conjunto predefinido de valores.

Por ejemplo, considera una aplicación de gestión de pedidos en línea. En este caso, tendrás diferentes estados para un pedido, como "pendiente", "en proceso", "enviado" o "entregado". Estos estados son constantes y se utilizan en todo el sistema para representar el estado actual de cada pedido.

Es crucial disponer de una forma eficiente y segura de representar y manejar estos conjuntos constantes de datos. Aquí es donde entran en juego los enums en Java.

Enums

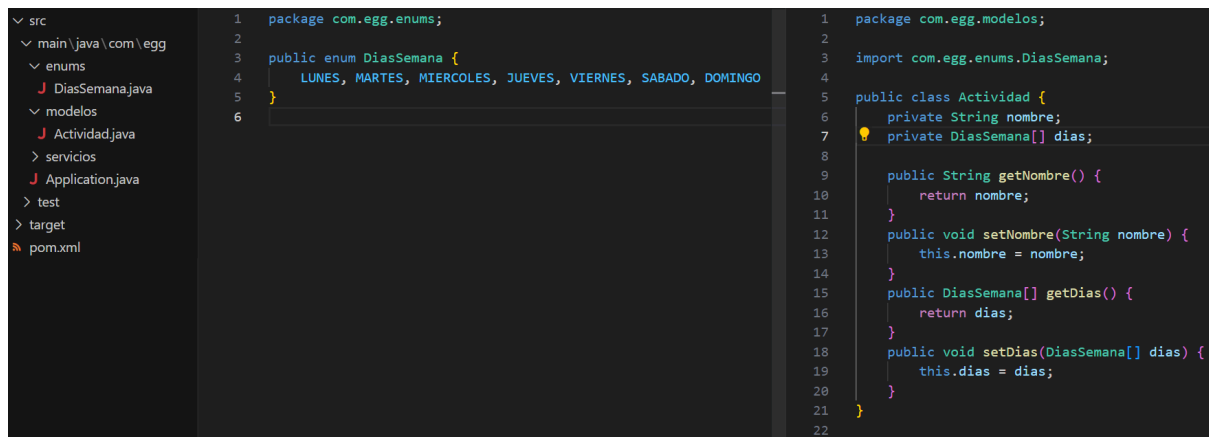
Los enums (enumeraciones) en Java son un tipo de datos especial que permite definir una colección de constantes nombradas. Son una forma más segura y legible de representar un conjunto fijo de valores en comparación con el uso de constantes enteras o cadenas de texto.

Para declarar un enum en Java, utilizamos la palabra clave `enum` seguida del nombre del enum y los valores constantes entre llaves.

Aquí tienes un ejemplo básico de cómo se declara un enum llamado `DiaSemana` que representa los días de la semana:

```
public enum DiaSemana {  
    LUNES, MARTES, MIERCOLES, JUEVES, VIERNES, SABADO, DOMINGO  
}
```

Por convención, los nombres de las constantes son en mayúsculas.



```
1 package com.egg.enums;
2
3 public enum DiasSemana {
4     LUNES, MARTES, MIERCOLES, JUEVES, VIERNES, SABADO, DOMINGO
5 }
6
```

```
1 package com.egg.modelos;
2
3 import com.egg.enums.DiasSemana;
4
5 public class Actividad {
6     private String nombre;
7     private DiasSemana[] dias;
8
9     public String getNombre() {
10         return nombre;
11     }
12     public void setNombre(String nombre) {
13         this.nombre = nombre;
14     }
15     public DiasSemana[] getDias() {
16         return dias;
17     }
18     public void setDias(DiasSemana[] dias) {
19         this.dias = dias;
20     }
21 }
22
```

Es buena práctica que los Enum estén alojados en una carpeta independiente.

Enums con propiedades

En Java, los enums son tratados como clases especiales y pueden tener métodos, campos y constructores. Por ejemplo, podemos agregar un campo descripción al enum `DiaSemana` y un constructor para asignar un valor a este campo para cada constante del enum. También podemos definir un método para obtener la descripción de cada día de la semana.

```
public enum DiaSemana {
    LUNES("Primer día de la semana"),
    MARTES("Segundo día de la semana"),
    MIERCOLES("Tercer día de la semana"),
    JUEVES("Cuarto día de la semana"),
    VIERNES("Quinto día de la semana"),
    SABADO("Sexto día de la semana"),
    DOMINGO("Séptimo día de la semana");

    private String descripcion;

    private DiaSemana(String descripcion) {
        this.descripcion = descripcion;
    }

    public String getDescripcion() {
        return descripcion;
    }
}
```

En este ejemplo, se agrega un campo descripción al enum `DiaSemana` y un constructor que permite asignar un valor a este campo para cada constante del enum. También se define un método `getDescripcion()` para obtener la descripción de cada día de la semana, pero no se crea el método `setDescription()`, no solo porque al ser una constante no debería cambiar sus propiedades, sino también porque no se puede.

Las constantes de un enum y sus propiedades se cargan de manera estática. En Java, los enums se consideran tipos de datos finales y se definen como conjuntos fijos de valores constantes en tiempo de compilación. Esto significa que una vez que se ha definido un enum y se ha asignado un conjunto de constantes, no se pueden agregar, eliminar o modificar esas constantes durante la ejecución del programa.

Manipulación de Enums

Aunque no se pueden modificar las constantes de un Enum en tiempo de ejecución, podemos realizar varias operaciones y manipulaciones con ellos. Por ejemplo, podemos acceder a las constantes del enum, obtener todos los valores, comparar valores de enum y utilizar enums en estructuras de control como `switch`.

Enum en Java es una clase final extendida implícitamente por el compilador, y por lo tanto, hereda métodos de la clase `java.lang.Enum`. Algunos de los métodos más comunes proporcionados por la clase `Enum` son:

1. **values():** Este método devuelve un arreglo que contiene todas las constantes enum en el orden en que fueron declaradas.
2. **valueOf(String name):** Este método devuelve la constante enum cuyo nombre coincide con la cadena especificada.
3. **ordinal():** Este método devuelve la posición (índice) de la constante enum en su declaración, comenzando desde cero.
4. **name():** Este método devuelve el nombre de la constante enum tal como fue declarado. Es útil para obtener el nombre de la constante enum como una cadena.

```
public class Application {
    public static void main(String[] args) {
        DiaSemana dia = DiaSemana.LUNES;
        // Acceder a las constantes del enum
        System.out.println(dia);

        // Obtener todos los valores del enum
        DiaSemana[] dias = DiaSemana.values();
        for (DiaSemana d : dias) {
            System.out.println(d);
        }
    }
}
```

```
// Comparar valores de enum
if (dia == DiaSemana.LUNES) {
    System.out.println("Es LUNES");
}

// Utilizar enums en estructuras de control
switch (dia) {
    case LUNES, MARTES, MIERCOLES, JUEVES, VIERNES ->
System.out.println("Es un día laboral");
    case SABADO, DOMINGO -> System.out.println("Es fin de semana");
}
}
```

Con estas operaciones, podemos aprovechar al máximo la potencia y la flexibilidad de los enums en Java para representar conjuntos de datos constantes de manera segura y eficiente.