Enum (Enumerados) en Java

Un enumerado (o Enum) es una clase "especial" (tanto en Java como en otros lenguajes) que limitan la creación de objetos a los especificados explícitamente en la implementación de la clase. La única limitación que tienen los enumerados respecto a una clase normal es que si tiene constructor, este debe de ser privado para que no se puedan crear nuevos objetos.

Vamos a empezar con un sencillo ejemplo sobre una clase Enum. Volviendo a los ejemplo relacionados con el fútbol, tenemos que lo futbolistas están caracterizados por una demarcación a la hora de jugar un partido de fútbol, por tanto las demarcaciones en las que puede jugar un futbolista son finitas y por tanto se pueden enumerar en: Portero, Defensa, Centrocampista y Delantero. Con esta especificación podemos crearnos la siguiente clase "Enum" llamada "Demarcación":

```
public enum Demarcacion
{
    PORTERO, DEFENSA, CENTROCAMPISTA, DELANTERO
}
```

Por convenio los nombres de los enumerados se escriben en mayúsculas.

Es muy importante entender que un "`Enum`" en java es realmente una clase (cuyos objetos solo pueden ser los definidos en esta clase: PORTERO, …, DELANTERO) que hereda de la clase "`Enum(java.lang.Enum)`" y por tanto los enumerados tienen una serie de métodos heredados de esa clase padre. A continuación vamos a mostrar algunos de los métodos más utilizados de los enumerados:

```
public enum Demarcacion{PORTERO, DEFENSA, CENTROCAMPISTA, DELANTERO}

Demarcacion delantero = Demarcacion.DELANTERO;// Instancia de un enum de la clase Demarcación
delantero.name();  // Devuelve un String con el nombre de la constante (DELANTERO)
delantero.toString(); // Devuelve un String con el nombre de la constante (DELANTERO)
delantero.ordinal();  // Devuelve un entero con la posición del enum según está declarada (3).
delantero.compareTo(Enum otro);// Compara el enum con el parámetro según el orden
Demarcacion.values();    // Devuelve un array que contiene todos los enum
```

Visto cuales son los métodos más utilizados dentro de los enumerados, vamos a poner un ejemplo para ver los resultados que nos devuelven estos métodos. Dado el siguiente fragmento de código:

```
Demarcacion delantero = Demarcacion.DELANTERO;
Demarcacion defensa = Demarcacion.DEFENSA;

// Devuelve un String con el nombre de la constante
System.out.println("delantero.name()= "+delantero.name());
System.out.println("defensa.toString()= "+defensa.toString());

//  Devuelve un entero con la posición de la constante según está
declarada.
System.out.println("delantero.ordinal()= "+delantero.ordinal());

// Compara el enum con el parámetro según el orden en el que están
// declaradas las constantes.
System.out.println("delantero.compareTo(portero)=
                        "+delantero.compareTo(defensa));
```

```
System.out.println("delantero.compareTo(delantero)=
                              "+delantero.compareTo(delantero));

// Recorre todas las constantes de la enumeración
for(Demarcacion d: Demarcacion.values()){
     System.out.println(d.toString()+" - ");
}
```

Tenemos como salida los siguientes resultados:

```
delantero.name()= DELANTERO
defensa.toString()= DEFENSA
delantero.ordinal()= 3
delantero.compareTo(defensa)= 2
delantero.compareTo(delantero)= 0
PORTERO - DEFENSA - CENTROCAMPISTA – DELANTERO
```

A continuación mostramos la implementación de la clase "Futbolista":

```
public class Futbolista {

       private int dorsal;
       private String Nombre;
       private Demarcacion demarcacion;
       private String equipo;

       public Futbolista() {
       }

       public Futbolista(String nombre, int dorsal, Demarcacion demarcacion, String
equipo) {
              this.dorsal = dorsal;
              Nombre = nombre;
              this.demarcacion = demarcacion;
              this.equipo = equipo;
       }

       // Metodos getter y setter

       ..............................


       @Override
       public String toString() {
              return this.dorsal + " - " + this.Nombre + " - "
                           + this.demarcacion.name() + " - " + this.equipo;
       }
}

Futbolista casillas = new Futbolista("Casillas", 1, Demarcacion.PORTERO, "Real Madrid");
Futbolista   capdevila   =   new   Futbolista("Capdevila",   11,   Demarcacion.DEFENSA,
"Villarreal");
Futbolista iniesta = new Futbolista("Iniesta", 6, Demarcacion.CENTROCAMPISTA, "Barsa");
Futbolista navas = new Futbolista("Navas", 22, Demarcacion.DELANTERO, "Sevilla");
```

Como vemos la demarcación sólo puede ser uno de los valores declarados en la clase enumerado. Si llamamos al método "toString()" declarado en la clase futbolista, podemos imprimir por pantalla los datos de los futbolistas. Dado el siguiente código:

```
System.out.println(casillas.toString());
System.out.println(capdevila.toString());
System.out.println(iniesta.toString());
System.out.println(navas.toString());
```

Y dado el siguiente método "toString()"

```
@Override
public String toString() {
     return this.dorsal + " - " + this.Nombre + " - "
               + this.demarcacion.name() + " - " + this.equipo;
}
```

Tenemos como salida lo siguiente:

```
1 - Casillas - PORTERO - Real Madrid
11 - Capdevila - DEFENSA - Villareal
6 - Iniesta - CENTROCAMPISTA - Barsa
22 - Navas - DELANTERO - Sevilla
```