Tipos Enumerados en Java

José Manuel Pérez Lobato

http://docs.oracle.com/javase/tutorial/java/javaOO/enum.html

Sintaxis

[ModificadorAcceso] **enum** NombreTipo{

VALOR1, VALOR2, VALOR3, VALOR4

}[;]

- El punto y coma (;) al final de la declaración del tipo enumerado es opcional.
- No es necesario que las constantes de los valores estén en mayúsculas pero por convenio se debe hacer.
- El nombre del tipo sigue el mismo convenio que los nombres de las clases.
- Un tipo enumerado puede ser declarado dentro o fuera de una clase, pero no dentro de un método.

Características

- Sirven para que una variable tome valor sólo dentro de un conjunto de valores predefinidos.
- Si tenemos una aplicación para los tipos de colores restringidos a ROJO, VERDE y AZUL podemos crear un tipo enumerado para delimitar dicha selección:

Declaración:

public enum Color {ROJO, VERDE, AZUL};

Utilización (indicando el tipo (Color) al que pertenece)

```
static void prueba1() {
   Color c=Color.ROJO;
   System.out.println(c);
  }
```

Las constantes de un tipo enumerado son, implícitamente, static y final y no pueden ser cambiadas después de creadas.

Color.ROJO= Color.VERDE; sería una instrucción errónea.

Uso de enum en switch

- Los tipos enumerados se pueden utilizar en los switchs:

```
static void prueba3(Color c){
  switch (c) {
    case ROJO: System.out.println("Colorado"); break;
    case VERDE: System.out.println("Color de la hierba");break;
    case AZUL: System.out.println("Color del mar");break;
    default : System.out.println("Error");break;
  }
}
public static void main (String arg[]){
    prueba3(Color.AZUL);
    Color c=null;   //Correcto porque c no es simplemente un valor, se trata como un objeto
    prueba3(c);  //Error de ejecución, NullPointerException en la instrucción switch (c ) {..
  }
```

- No hace falta poner Color.Rojo en el switch porque la variable c ya se ha definido como de tipo Color
- No se puede poner en el switch un case null ya que ese no es un valor válido.
- El default, en este ejemplo, es inalcanzable porque se han referenciado anteriormente todos los posibles valores pero puede ser útil si no se referencian todos ellos. Por ejemplo si hubiéramos omitido el AZUL.

Valores asociados y métodos adicionales

- Los tipos enumerados son como una clase de tipo especial en Java y **pueden contener constructores** (que deben ser **private o package** pero no public) , **métodos y atributos**.
- Las **constantes** se deben definir **primero**, antes que cualquier otro atributo o método
- Cuando un enum tiene atributos o métodos la **lista de constantes** debe **acabar con punto y coma**.
- Se puede **asociar un valor numérico, o de otro tipo** (char, String, etc.) **a cada valor enumerado** (por ejemplo 5 para PEQUEÑO, 8 para MEDIANO, etc.) En ese caso hay que **crear un atributo del tipo del valor asociado y un constructor** que realice la asignación del valor.

Constructores, métodos y variables dentro de un tipo enumerado

```
public enum TamanoB {
   PEQUENO(5),MEDIANO(8),GRANDE(10);
   int valor;
   private TamanoB (int valor){    this.valor=valor;  }
   int getValor(){return valor;  }
}


//método en otra clase:
 static void prueba2(){
   Bebida b;
   b= new Bebida(TamanoB.GRANDE, "Refresco de
naranja");
   System.out.println(b);
   System.out.println(b.getTamano());   //GRANDE
   System.out.println(b.getTamano().getValor()); //10
   b.setTamano(TamanoB.MEDIANO);
   }
```

```
public class Bebida {
  TamanoB tamano;
  String nombre;
  Bebida (TamanoB t,  String nom){
     tamano=t;
     nombre=nom;
  }
  public String toString() {
    return "Bebida [tamano=" + tamano + ", nombre=" +
nombre + "]";
  }
   public TamanoB getTamano() {    return tamano;  }
   public void setTamano(TamanoB tamano) {
      this.tamano = tamano;}
   }
```

Restricciones

- No se puede invocar al constructor directamente.
  - Sería erróneo poner **TamanoB t=new TamanoB(10);** aunque el constructor no fuera private.
- Se puede definir más de un argumento en un constructor de un tipo enumerado, asimismo, se puede definir más de un constructor para un tipo enumerado siempre y cuando este tenga argumentos diferentes(sobrecargar el constructor)

```
public enum TamanoMesa {
  PEQUENO(5,10),MEDIANO(8,15),GRANDE(10,25), ENORMECUADRADA(50);
  int ancho;
  int largo;
  TamanoMesa (int a, int l){
    ancho=a;
    largo=l;
  }
 TamanoMesa (int a){
      ancho=a;
      largo=10;
 }
}
```

Métodos heredados de Enum

- ***public final String name()*** Devuelve un String con el nombre de la constante que contiene tal y como aparece en la declaración.
- ***public final int ordinal()*** Devuelve un entero con la posición de la constante según está declarada. A la primera constante le corresponde la posición **cero**.
- ***public String toString()*** Devuelve un String con el nombre de la constante que contiene tal y como aparece en la declaración. Sobrescribe el método toString de la clase Object.
- ***public final boolean equals(Object other)*** Devuelve true si el valor de la variable enum es igual al objeto que recibe. Sobrescribe el método equals de la clase Object.
- ***public final int compareTo(Enum other)*** Compara el enum con el que recibe según el orden en el que están declaradas las constantes. Devuelve un número negativo, cero o un número positivo según el objeto sea menor, igual o mayor que el que recibe como parámetro. Solo se pueden comparar enumeraciones del mismo tipo.
- ***public static enumConstant valueOf(String s)*** Devuelve la constante que coincide exactamente con el String que recibe como parámetro.

Métodos adicionales

- El compilador añade automáticamente métodos adicionales cuando creamos un tipo enumerado con enum
- Entre otros añade un método :
- ***public static enumConstant* T[] values()**

  que devuelve un array que contiene todas las constantes de la enumeración en el orden en que se han declarado. Se suele usar en bucles for each para recorrer el enum:

  for (Color c : Color.values())

  System.out.printf("Color %s ", c);

Uso métodos heredados en Enum

```
public class UsoMetodosHeredadosEnum {
  public enum FinDeSemana {SABADO, DOMINGO};

    public static void main(String[] args) {
        FinDeSemana op1 = FinDeSemana.SABADO;
        FinDeSemana op2 = FinDeSemana.DOMINGO;
        if(op1.name().equals("SABADO"))
                System.out.println("Cadena SABADO");
        System.out.println(op1.ordinal());
        if(op1.compareTo(op2)>0)
                System.out.println(op1 + " > " + op2);
        String cadena = "SABADO";
        if(FinDeSemana.valueOf(cadena) == FinDeSemana.SABADO)
                System.out.println("Cadena SABADO");
        System.out.println("Listado");
        for(FinDeSemana x : FinDeSemana.values())
                System.out.println(x);
    }
}
```