Java Enum

Un enum, también llamado enumeración o tipo enumerado es un tipo de dato definido por el usuario que solo puede tomar como valores los definidos en una lista.

Un enum se declara de forma general:

```
[modificadores] enum nombreEnum {VALOR1, VALOR2, VALOR3, ...}
```

modificadores (opcional) puede ser public, private, protected además de static.

enum es la palabra reservada para definir enumeraciones en Java.

nombreEnum es el nombre del nuevo tipo creado.

VALOR1, VALOR2, ... son los valores que pueden tomar las variables que se declaren de este tipo.

Se pueden declarar enumeraciones dentro o fuera de una clase pero **nunca dentro de un método**.

Ejemplo:

```
public enum Dia {LUNES, MARTES, MIÉRCOLES, JUEVES, VIERNES, SABADO, DOMINGO}
```

Por convenio se suelen escribir los valores en mayúsculas ya que se trata de constantes.

Una vez creado el tipo enum ya podemos declarar variables.

Para declarar una variable de tipo Dia:

```
Dia d;
```

La variable d solo podrá tomar uno de los valores definidos en la lista de valores.

Para darle un valor a las variables:

nombreDelEnum.VALOR;

Por ejemplo:

```
d = Dia.JUEVES;
```

Ejemplo de uso:

```
public class Enum1 {

   enum Dia {LUNES, MARTES, MIERCOLES, JUEVES, VIERNES, SABADO, DOMINGO}

    public static void main(String[] args) {

        Dia d; // declaramos una variable del tipo Dia

        d = Dia.DOMINGO; //asignamos un valor

        if (d == Dia.DOMINGO || d == Dia.SABADO)  //comparamos valores

            System.out.println("Estamos en fin de semana");

        else

            System.out.println("Aún no ha llegado el fin de semana");
```

```
        switch (d) {  //ejemplo de switch
            case LUNES:
            case MARTES:
            case MIERCOLES:
            case JUEVES:
            case VIERNES:
               System.out.println("Aún no ha llegado el fin de semana");
                break;
            default:
                System.out.println("Estamos en fin de semana");
        }
    }
}
```

Los tipos enumerados en Java son mucho más potentes que sus equivalentes en lenguajes como C++:

Un enum en Java es una Clase.

En general para utilizarlos los escribiremos en una clase con el mismo nombre del enum que añadiremos a nuestro proyecto

```
//Archivo Dia.java
public enum Dia{
          LUNES, MARTES, MIÉRCOLES, JUEVES, VIERNES, SABADO, DOMINGO
}
```

Aunque ya hemos visto antes que esto no es necesario y podemos escribirlos en el mismo archivo de la clase donde los vamos a utilizar.

Si declaramos el enum en el mismo archivo que la clase que lo usa y fuera de esta clase debemos tener en cuenta que en un archivo .java solo puede haber una clase pública y aquí también se incluyen las enumeraciones.

Por ser una clase, un enum en Java **puede contener atributos y métodos**.

Cada constante del enum es un objeto de la clase.

Todas las clases enum declaradas heredan de forma implícita de la clase Enum de Java (java.lang.Enum)

Aunque un enum es una clase, no se pueden crear instancias de un enum.

Algunos métodos que se heredan de Enum:

name()

public final String name()

Devuelve un String con el nombre de la constante que contiene tal y como aparece en la declaración.

ordinal()

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.

toString()

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.

equals()

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.

compareTo()

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.

valueOf()

public static enumConstant valueOf(String s)

Devuelve la constante que coincide exactamente con el String que recibe como parámetro.

values()

public static enumConstant [] values()

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.

Ejemplo de uso de los métodos heredados de la clase Enum:

```
public class Enum2 {


    public enum Opcion {UNO, DOS, TRES, CUATRO}


    public static void main(String[] args) {
```

```
        Opcion op = Opcion.DOS;
        Opcion op2 = Opcion.CUATRO;
        if(op.name().equals("DOS"))
           System.out.println("Cadena DOS");
        System.out.println(op.ordinal());
        if(op2.compareTo(op)>0)
            System.out.println(op2 + " > " + op);
        String cadena = "UNO";
        if(Opcion.valueOf(cadena) == Opcion.UNO)
            System.out.println("Cadena UNO");
        for(Opcion x : Opcion.values())
            System.out.println(x);
    }
}
```

En Java podemos declarar constantes de un enum relacionadas con un valor.

Para ello la clase enum debe tener un atributo para cada valor relacionado con la constante y un constructor para asignar los valores.

Esos valores se pasan al constructor cuando se crea la constante.

El constructor debe tener acceso private o package. No puede ser público.

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.

Ejemplo de constantes enum con valores asociados. Los valores 100 y 80 que acompañan a las constantes se pasan al constructor.

```
//Archivo Precios.java
public enum Precios {PRECIONORMAL(100), PRECIOVIP(80);
     double precio;
     Precios(double p){
         precio = p;
     }
    public double getPrecio() {
        return precio;
    }
}
```

```
//Clase Principal
public class Enum3 {
    public static void main(String[] args) {
        Precios p = Precios.PRECIOVIP;     //precio = 80
        System.out.println(p.getPrecio()); //muestra 80
    }
}
```