Tipos de datos Enum

¿Qué es un ENUM? En su forma más simple, **una enumeración es una lista de constantes con nombre que definen un nuevo tipo de datos**. Un objeto de un tipo de enumeración solo puede contener los valores definidos por la lista. Por lo tanto, una enumeración le brinda una manera de definir con precisión un nuevo tipo de datos que tiene **un número fijo de valores válidos.**

Por ejemplo, los 4 palos en un mazo de cartas pueden ser 4 enumeradores llamados Picas, Diamantes, Corazones y Tréboles, que pertenecen a un tipo enumerado llamado Cartas. Otros ejemplos incluyen tipos de enumerados naturales (como los planetas, días de la semana, meses del año, colores, direcciones, etc.).

Desde una perspectiva de programación, las enumeraciones son útiles siempre que necesite definir un **conjunto de valores que represente una colección de elementos.** Por ejemplo, puede usar una enumeración para representar un conjunto de códigos de estado, como *éxito*, *espera*, *error* y *reintentos*, que indican el progreso de alguna acción. En el pasado, dichos valores se definían como **variables finales**, pero las enumeraciones ofrecen un enfoque más estructurado.

Declaración de enum en Java

La declaración de Enum puede hacerse fuera de **una clase**, o dentro de una clase (*class*), pero NO dentro de un método.

```
enum Estaciones {
   Invierno, Primavera, Verano, Otoño
}

class Principal {

   public static void main(String... args) {
      //TipoDeDato que acabamos de crear     nombre        incialización
      VariableCompuestaDatosPersona miVariable = new VariableCompuestaDatosPersona();

      for (Estaciones estacion : Estaciones.values()) {
         System.out.println(estacion);
      }
   }}

class Principal {

enum Estaciones {
   Invierno, Primavera, Verano, Otoño
}

   public static void main(String... args) {
      //TipoDeDato que acabamos de crear     nombre        incialización
      VariableCompuestaDatosPersona miVariable = new VariableCompuestaDatosPersona();

      for (Estaciones estacion : Estaciones.values()) {
         System.out.println(estacion);
      }
   }
}
```

Ejemplo con enum en Java

Por ejemplo, aquí hay una enumeración simple que enumera varias formas de transporte:

```
//Una enumeración de transporte

enum Transporte{

    COCHE, CAMIÓN, AVION, TREN, BARCO;}
```

Una vez que haya definido una enumeración, **puedo crear una variable de ese tipo**. Se declara y utiliza una variable de enumeración de la misma manera que con los tipos primitivos. Por ejemplo, esto declara *tp* como una variable del tipo de enumeración *Transporte*:

```
Transporte tp;
```

Como *tp* es de tipo *Transporte*, los únicos valores que se le pueden asignar son los definidos por la enumeración. Por ejemplo, esto asigna *tp* el valor *AVION*:

```
tp = Transporte.AVION;
```

Se pueden comparar dos constantes de enumeración utilizando el operador relacional **==**. Por ejemplo, esta declaración compara el valor en *tp* con la constante *TREN*:

```
if(tp == Transporte.TREN) // ...
```

Un valor de enumeración también se puede usar para controlar una sentencia switch. Por supuesto, todas las declaraciones de **case** deben usar constantes de la misma enumeración que la utilizada por la expresión de switch. Por ejemplo, este *switch* es perfectamente válido:

```
//Uso de enum para controlar una sentencia switch

switch(tp){

            case COCHE:

                //

            case CAMION:

                //

        }
```

Observe que en las sentencias *case*, los nombres de las constantes de enumeración se usan sin estar calificados por el nombre de tipo de enumeración. Es decir, se utiliza *CAMION*, no *Transporte.CAMION*. Esto se debe a que el tipo de enumeración en la expresión de *switch* ya ha especificado implícitamente el tipo de enumeración de las constantes de *case*.

4.1. Código de Ejemplo

```
enum Transporte {
   COCHE, CAMION, AVION, TREN, BARCO;
}

class Principal {

   public static void main(String... args) {
      //Crear variable tp
      Transporte tp;

      //Asignarle un valor de los que tiene dentro el conjunto
      //Siempre se accede con el nombre del enum y el ., luego nos mostrara el listado de constantes
      tp = Transporte.AVION;

      //Mostramos el valor de la variable tp
      System.out.println("Valor de tp: " + tp);
      System.out.println();

      //asignamos un valor nuevo
      tp = Transporte.TREN;

      //Comparación de 2 valores enum, podemos comparar con el ==
      if (tp == Transporte.TREN) {
         System.out.println("tp tiene el valor de TREN\n");
      }

      //enum para controlar sentencia switch
      switch (tp) {
         //En este caso no hace falta utiilzar el nombre de la variable antes del nombre de la constante porque ya esta
puesto en los parentesis del switch
         case COCHE:
            System.out.println("Un auto lleva personas.");
            break;
         case CAMION:
            System.out.println("Un camión lleva carga.");
            break;
         case AVION:
            System.out.println("Un avión vuela.");
            break;
         case TREN:
            System.out.println("Un tren corre sobre rieles.");
            break;
         case BARCO:
            System.out.println("Un barco navega en el agua.");
            break;
      }
   }
}
```

Antes de continuar, es necesario hacer un punto sobre estilo. Las constantes en *Transporte* usan mayúsculas. (Por lo tanto, se usa *COCHE*, *no coche*.) Sin embargo, **no se requiere el uso de mayúsculas**. En otras palabras, no existe una regla que requiera que las constantes de enumeración estén en mayúsculas.

Debido a que las enumeraciones a menudo reemplazan las variables finales, que tradicionalmente se han usado en mayúsculas, algunos programadores creen que las **constantes de enumeración en mayúsculas también son apropiadas**. Hay otros puntos de vista y estilos.