Tema puntual : Variables de Tipo Enum , uso y ejemplos.

**TIPOS ENUMERADOS (ENUM) EN JAVA**

Un tipo enumerado en Java es, de partida, similar al de otros lenguajes de programación. No obstante, incluye una serie de características que lo hacen muy potente. Un tipo enumerado restringe los posibles valores que puede tomar una variable. Esto ayuda a reducir los errores en el código y permite algunos usos especiales interesantes.

 Los tipos enumerados se declaran con esta sintaxis tipo:

|  |
| --- |
| Tipo Público/Privado enum nombreTipoEnumerado { ELEMENTO1, ELEMENTO2, ELEMENTO3, …, ELEMENTOn }; |

Dentro de las llaves se declaran las variables (objetos) de que consta el tipo enumerado. Por convención, sus nombres se escriben en letras mayúsculas para recordarnos que son valores fijos (que en cierto modo podemos ver como constantes). Una vez declarado el tipo enumerado, todavía no existen variables hasta que no las creemos explícitamente, de la misma manera que ocurre con cualquier tipo Java. Para crear una variable de tipo enumerado lo haremos con una declaración simple que recuerda a la creación de una variable tipo primitivo: nombreTipoEnumerado miNombreElegido;.

Esta forma de creación de variables de tipo enumerado se justifica porque los tipos enumerados en principio no tienen constructores (más adelante veremos que sí los pueden tener). Los valores de un tipo enumerado son objetos propiamente dichos, ¿de qué tipo? Del tipo enumerado cuyo nombre se haya indicado. No se pueden crear más objetos variantes del tipo enumerado que los especificados en su declaración. Tener en cuenta, porque es una confusión habitual, que los tipos enumerados no son enteros, ni cadenas (aunque a veces podamos hacer que se comporten de forma similar a como lo haría un entero o una cadena). Cada elemento de un enumerado es un objeto único disponible para su uso.

La identificación de cada objeto del tipo se hace con la sintaxis del punto, es decir, nos referimos a un elemento concreto como nombreDelTipoEnumerado.ELEMENTO1. Esta sintaxis nos quiere recordar lo que sería un campo de una clase pero no es así: en este caso es un objeto de un tipo enumerado.

**Un tipo enumerado puede ser declarado dentro o fuera de una clase,** pero no dentro de un método. Por tanto no podemos declarar un enum dentro de un método main (programa principal); si lo hacemos nos saltará el error de compilación “enum types must not be local” (los tipos enumerados no deben ser locales a un método). Por tanto el tipo enumerado lo declararemos o bien antes del public class… o bien después del public class… pero fuera de cualquier método o constructor. Veamos un primer ejemplo de uso:

|  |
| --- |
| /\*\*   \* Descripción para javadoc antes de la clase. Clase de ejemplo aprenderaprogramar.com para tipos enumerados   \* @author Alex Rodríguez   \*/    public class TestEnum {  enum TipoDeMadera { ROBLE, CAOBA, NOGAL, CEREZO, BOJ };      public static void main (String[ ] Args) {          TipoDeMadera maderaUsuario;          maderaUsuario = TipoDeMadera.ROBLE;          System.out.println ("La madera elegida por el usuario es " + maderaUsuario.toString().toLowerCase() );          System.out.println ("¿Es la madera elegida por el usuario caoba? Resultado: " + (maderaUsuario==TipoDeMadera.CAOBA) );          System.out.println ("¿Es la madera elegida por el usuario roble? Resultado: " + (maderaUsuario==TipoDeMadera.ROBLE) );      } //Cierre del main  } //Cierre de la clase |

Ejecuta el anterior código. El resultado de la ejecución será similar a este:

|  |
| --- |
| La madera elegida por el usuario es roble  ¿Es la madera elegida por el usuario caoba? Resultado: false  ¿Es la madera elegida por el usuario roble? Resultado: true |

Hemos usado el método toString() que es un método disponible para la mayoría de las clases en Java y utilizado para representar un objeto en una cadena de texto. Ten en cuenta que usamos este método porque el enumerado no es texto.

El enumerado lo hemos incluido como código dentro de la clase. En realidad, podríamos haberlo dispuesto como una clase independiente que llevara únicamente este código:

public enum TipoDeMadera { ROBLE, CAOBA, NOGAL, CEREZO, BOJ }

**EJERCICIO**

Crea una clase Vehiculo donde definas un **tipo enumerado MarcaDeVehiculo** cuyos valores posibles serán FORD, TOYOTA, SUZUKI, RENAULT, SEAT. Establece como atributos de la clase matricula (tipo String) y marca (tipo MarcaDeVehiculo) y crea los métodos getters (para obtener) y setters (para establecer el valor) de los atributos. Escribe un pequeño programa de prueba donde verifiques que los métodos funcionan correctamente. Puedes comprobar si tu código es correcto consultando en los foros aprenderaprogramar.com.

**¿Cómo empezar?**

Para declarar este tipo de dato, existe una palabra reservada, al igual que las clases e interfaces, con la que podemos definir una enumeración. Esa palabra es enum. Podemos definir los Enum Types para establecer los distintos valores posibles al igual que hacemos con las constantes estáticas en el cuerpo de una declaración:

package co.com.mm.enumeraciones;  
   
public enum Estado {  
       
    /\*\*  
     \* Indica que el proceso no ha iniciado.  
     \*/  
    POR\_INICIAR,  
    /\*\*  
     \* Indica que el proceso inicio pero no ha finalizado.  
     \*/  
    EN\_PROGRESO,  
    /\*\*  
     \* Indica que el proceso finalizo correctamente.  
     \*/  
    FINALIZADO,  
    /\*\*  
     \* Indica que el proceso finalizo con errores.  
     \*/  
    ERROR;  
}

Como puedes ver en el anterior código, estamos definiendo una enumeración la cual contiene múltiples valores para determinar el estado de un proceso. Para llevarlo a cabo, en el cuerpo donde establecemos la enumeración, definimos varios nombres que harían referencia a uno de los estados que hemos configurado anteriormente.

### ¿Cómo utilizar las enumeraciones?

Vamos a crear una clase main donde verás la utilización de los tipos enum

package co.com.mm.enumeraciones;  
  
public class Main {  
       
    public static void main(String[] args) {  
        imprimirMensaje(Estado.POR\_INICIAR);  
        imprimirMensaje(Estado.EN\_PROGRESO);  
        imprimirMensaje(Estado.FINALIZADO);  
        imprimirMensaje(Estado.ERROR);  
    }  
       
    /\*\*  
     \* Imprime un mensaje de acuerdo a la enumeraci&aocute;n.  
     \* @param estado  
     \*/  
    public static void imprimirMensaje(Estado estado) {  
        switch(estado) {  
            case POR\_INICIAR:  
                System.out.printf("El proceso aún no ha iniciado. " +  
                    "Estado: [ %s ].", estado.name());  
                break;  
            case EN\_PROGRESO:  
                System.out.printf("El proceso se encuentra en progreso. " +  
                    "Estado: [ %s ].", estado.name());  
                break;  
            case FINALIZADO:  
                System.out.printf("El proceso finalizó correctamente. " +  
                    "Estado: [ %s ].", estado.name());  
                break;  
            case ERROR:  
                System.out.printf("El proceso finalizo con errores. " +  
                    "Estado: [ %s ].", estado.name());  
                break;  
        }  
           
        System.out.println();  
    }  
}

Destacar del código anterior los siguientes aspectos: Si os habéis dado cuenta, al igual que pasa con los tipos de datos ordinarios, las clases y también las interfaces, los Enum Types definen un tipo de dato que más tarde podemos referenciar con el nombre del mismo. También podemos utilizar switch para comprobar el CASE sobre cada enumeración, esto es posible ya que este tipo de datos se comprueba de la misma manera que los tipos de datos que solemos utilizar a menudo. Por última, deciros también que las enumeraciones cuentan con un método por defecto llamado name que devuelve un string con el nombre del elemento enumerado.

### Enumeración con constructor

A continuación os vamos a explicar como manejar la enumeración con constructor, y así ampliarlo con más información que el nombre de los elementos a enumerar. Para ello, podemos creamos un constructor que agregue estos valores adicionales de los que te hemos hablado antes.

package co.com.mm.enumeraciones;  
   
/\*\*  
 \* Enumeracion que define los diferentes estados de un proceso X.  
 \*/  
public enum Estado {  
       
    /\*\*  
     \* Indica que el proceso no ha iniciado.  
     \*/  
    POR\_INICIAR("El proceso aún no ha iniciado. Estado: [ %s ]."),  
    /\*\*  
     \* Indica que el proceso inicio pero no ha finalizado.  
     \*/  
    EN\_PROGRESO("El proceso se encuentra en progreso. Estado: [ %s ]."),  
    /\*\*  
     \* Indica que el proceso finalizo correctamente.  
     \*/  
    FINALIZADO("El proceso finalizó correctamente. Estado: [ %s ]."),  
    /\*\*  
     \* Indica que el proceso finalizo con errores.  
     \*/  
    ERROR("El proceso finalizo con errores. Estado: [ %s ].");  
       
    /\*\*  
     \* Mensaje del estado.  
     \*/  
    private String mensaje;  
       
    private Estado(String mensaje) {  
        this.mensaje = mensaje;  
    }  
       
    /\*\*  
     \* Retorna el mensaje de la enumeración.  
     \* @return  
     \*/  
    public String getMensaje() {  
        return this.mensaje;  
    }  
}

En este ejemplo, hemos definido:

* Un constructor el cual os obligatorio poner como privado para poder esperar la lista de valores de las enumeraciones y no otras instancias.
* Un método llamado getMensaje que puede ser llamado por clientes de la enumeración para obtener el mensaje correspondiente
* Parentesis en los elementos de la enumeración, para que cada elemento represente una instancia de la enumeración.

A continuación, vamos a modificar la clase main para adaptarla a las nuevas características del Enum Type

package co.com.mm.enumeraciones;  
  
public class Main {  
   
    public static void main(String[] args) {  
        imprimirMensaje(Estado.POR\_INICIAR);  
        imprimirMensaje(Estado.EN\_PROGRESO);  
        imprimirMensaje(Estado.FINALIZADO);  
        imprimirMensaje(Estado.ERROR);  
    }  
   
    /\*\*  
     \* Imprime un mensaje de acuerdo a la enumeraci&aocute;n.  
     \*  
     \* @param estado  
     \*/  
    public static void imprimirMensaje(Estado estado) {  
        System.out.printf(estado.getMensaje(), estado.name());  
        System.out.println();  
    }  
}

Como puedes ver hemos simplificado el ejemplo para usar la enumeración de una manera más óptima