Tipo enumerado: enum

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:

```
public enum nombreTipoEnumerado { ELEMENTO1, ELEMENTO2,  …, 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. Los valores de un tipo enumerado son objetos propiamente dichos.

No se pueden crear más objetos variantes del tipo enumerado que los especificados en su declaración.

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.

No podemos declarar un enum dentro de un método main (programa principal); si lo hacemos nos saltará el error de compilación. Veamos un primer ejemplo de uso:

```
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 es: " + maderaUsuario.toString());
     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) );
   }
}
```

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

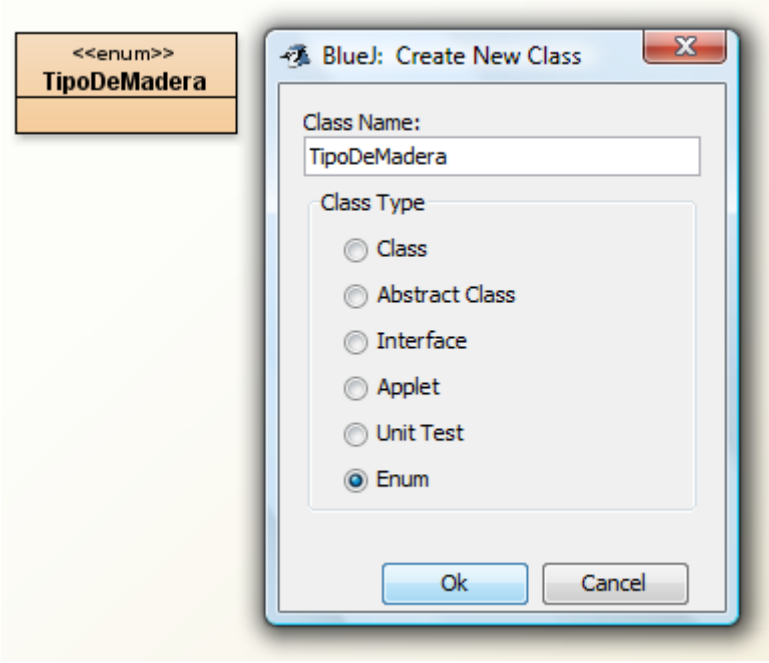
```
La madera elegida es ROBLE

¿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 }
```

En BlueJ, cuando pulsamos sobre New Class… una opción es Enum. Si elegimos esta opción (o si el código que escribimos se corresponde con una clase Enum), la clase se verá en el diagrama de clases con la palabra <<Enum>> en su icono.



Se dispone de métodos especiales como el método **values()**, que devuelve un array con todos los valores del enum.