3-8

Enumeraciones

Enumerated Types

- Java permite tipos enumerados para declarar variables.
- Un tipo enumerado establece los valores posibles para una variable de ese tipo.
- Los valores son identificadores escogidos por el programados
- Ejemplo: Una enumeración de las estaciones del año:

```
enum Estacion {invierno, primavera, verano,
                                otono};
```

- Se pueden listar cualquier cantidad de valores.

Enumerated Types

- Una vez que la enumeración es definida, se pueden decalarar variables de ese tipo.
  - `Estacion temporada;`

  y se le puede asignar un valor.

```
temporada = Estacion.primavera;
```

- Los valores que puede recibir, son los especificados en la enumeración.
- No se puede asignar otro valor distinto a los enumerados.

Enumerated Types

- Las enumeraciones son un tipo especial de clases, y cada variable enumerada es un objeto.

- EL método `ordinal()` regresa el valor ordinal del objeto.

- El método `name()` regresa el nombre del identificador correspondiente al valor del objeto.

```
public class EjemploEnum {
    enum Sabor {vainilla, chocolate, fresa, coco, cafe,
                napolitano, menta, chocochip}

    public static void main (String[] args)    {
      Sabor cono1, cono2, cono3;
      cono1 = Sabor.menta;
      cono2 = Sabor.chocolate;
      System.out.println ("Valor cono1: " + cono1);
      System.out.println ("Ordinal cono1: " + cono1.ordinal());
      System.out.println ("Nombre cono1: " + cono1.name());
      System.out.println ("\nValor cono2: " + cono2);
      System.out.println ("Ordinal cono2: " + cono2.ordinal());
      System.out.println ("Nombre cono2: " + cono2.name());
      cono3 = cono1;
      System.out.println ("\nValor cono2: " + cono3);
      System.out.println ("Ordinal cono2: " + cono3.ordinal());
      System.out.println ("Nombre cono3: " + cono3.name());
   }
}
```

13

Clases envolventes

Wrapper Classes

- El paquete `java.lang` provee clases envolventes para los tipos primitivos.
- Son útiles para usar tipos promitivos como objetos.

Tipo primitivo	Clase envolvente (wrapper)
`byte`	`Byte`
`short`	`Short`
`int`	`Integer`
`long`	`Long`
`float`	`Float`
`double`	`Double`
`char`	`Character`
`boolean`	`Boolean`
`void`	`Void`

Wrapper Classes

- La declaración crea un objeto `Integer` que representa un entero con valor 40.

```
Integer age = new Integer(40);
```

- Cómo es objeto, tiene métodos que pueden ser utilizados.
- Por ejemplo, `Integer` contiene un método para convertir una cadena `String` a un valor `int`:

```
num = Integer.parseInt(str);
```

- Las clases envolventes también contienen constantes útiles.
- Por ejemplo, la clase `Integer` contiene `MIN_VALUE` y `MAX_VALUE` que indican los valores extremos para tipos `int`

String => value

- El método parse funciona para obtener el valor a partir de una cadena.

```
Integer.parseInt(“42”)          => 42
Boolean.parseBoolean(“true”)    => true
Double.parseDouble(“2.71”)      => 2.71
```

Valores MAX_VALUE :


```
byteObj = new Byte(Byte.MAX_VALUE);
shortObj = new Short(Short.MAX_VALUE);
intObj = new Integer(Integer.MAX_VALUE);
longObj = new Long(Long.MAX_VALUE);
floatObj = new Float(Float.MAX_VALUE);
doubleObj = new Double(Double.MAX_VALUE);

printNumValues("MAXIMUM NUMBER VALUES:");
```

Valores MAX_VALUE :

Byte:127

Short:32767

Integer:2147483647

Long:9223372036854775807

Float:3.4028235E38

Double:1.79769313486231570E308

Autoboxing – from Primitive to Wrapper

- *Convertir un tipo primitivo en un objeto de su clase envolvente se llama: Autoboxing*
- Ejemplo:

```
Integer obj;
int num = 42;
obj = num;
```

- La asignación crea un objeto Integer.
- También funciona a la inversa (Unboxing)

num = obj;

```
public class EjemploWrapper {

    public static void main(String[] args) {
        Double dblwrap1 = new Double("3.14159");
        Double dblwrap2 = new Double(Math.PI);

        System.out.println("Entero de dblwrap1 = "
                + dblwrap1.intValue());
        System.out.println("Si comparamos dblwrap1 y dblwrap2??  "
                + dblwrap1.compareTo(dblwrap2));
        System.out.println("dblwrap1 es igual a dblwrap2??  "
                + dblwrap1.equals(dblwrap2));
    }
}
```