Uso de la Clase DecimalFormat

Muchas veces necesitamos imprimir una variable doble que almacenamos o que obtuvimos por medio de algún cálculo matemático, pero nos vemos con el problema de que nos salen unos 20 decimales, así que para solucionar este problema utilizaremos la Clase DecimalFormat, primero veamos algunos conceptos acerca de esta clase:

DecimalFormat: Es una clase de Java que nos permite mostrar los números en la pantalla (sea en consola o en un JtextFiled) con un formato deseado, es decir, limitar el número de decimales, si usaremos punto o coma, etc, incluso podemos tomar los valores desde un JTextField y reconstruir el número, ahora el ejemplo:

```
//import requerido para el Formateador

import java.text.DecimalFormat;
DecimalFormat formateador = new DecimalFormat("####.####");
// Imprime esto con cuatro decimales, es decir: 7,1234
System.out.println (formateador.format (7.12342383));
```

Si utilizamos ceros en lugar de los #, los dígitos no existentes se rellenarán con ceros, un ejemplo:

```
DecimalFormat formateador = new DecimalFormat("0000.0000");
// Imprime con 4 cifras enteras y 4 decimales: 0001,8200
System.out.println (formateador.format (1.82));
```

También podemos utilizar el signo de porcentaje (%) en la máscara y así el número se multiplicará automáticamente por 100 al momento de Imprimir.

```
DecimalFormat formateador = new DecimalFormat("###.##%");
// Imprime: 68,44%
System.out.println (formateador.format(0.6844));
```

La clase DecimalFormat usa por defecto el formato para el lenguaje que tengamos instalado en el ordenador. Es decir, si nuestro sistema operativo está en español, se usará la coma para los decimales y el punto para los separadores de miles. Si estamos en inglés, se usará el punto decimal.
Una opción para cambiar esto, es utilizar la clase DecimalFormatSymbols, que vendrá rellena con lo del idioma por defecto, y cambiar en ella el símbolo que nos interese. Por ejemplo, si estamos en español y queremos usar el punto decimal en vez de la coma, podemos hacer lo siguiente:

```
import java.text.DecimalFormat;
import java.text.DecimalFormatSymbols;
DecimalFormatSymbols simbolos = new DecimalFormatSymbols();
simbolos.setDecimalSeparator('.');
DecimalFormat formateador = new DecimalFormat("####.####",simbolos);
// Imprime: 3.4324
System.out.println (formateador.format (3.43242383));

Reconstruyendo un Número:
Supongamos que leemos algún número desde consola o un JTextField y deseamos reconstruirlo
con DecimalFormat, se hace de la siguente forma:

JTextField textField = new JTextField();
DecimalFormat formateador = new DecimalFormat("####.####");
String texto = textField.getText();
try
{
// parse() lanza una ParseException en caso de fallo que hay que capturar.
   Number numero = formateador.parse(texto);
   double valor = numero.doubleValue();
   // Optimizando las Lineas anteriores:
   // double valor = formateador.parse(texto).doubleValue();
}
catch (ParseException e)
{
// Error. El usuario ha escrito algo que no se puede convertir a número.
}
```