IES CHAN DO MONTE
C.S. de Desarrollo de Aplicaciones Multiplataforma
Modulo Acceso a datos

ANEXO:

Java: Manejo y formateo de Fechas: DateFormat y SimpleDateFormat

Java cuenta el número de milisegundos desde el **primero de enero de 1970**. Esto significa, que por ejemplo, el 2 de Enero de 1970 empezó 86400000 milisegundos después. Asimismo, el 31 de Diciembre de 1969 empezó 86400000 milisegundos antes que el 1 de enero de 1970.

La **clase Date de Java** toma nota de **esos milisegundos como un valor Long** y como este un número con signo, las fechas pueden ser expresadas antes y después del comienzo del 1 de enero de 1970.

La clase Date

La clase Date, encontrada en el paquete java.util, encapsula un valor Long representando un momento específico en el tiempo.

Constructores:

- **Date(),** el cual crea un objeto Date teniendo por valor el día, mes, años y hora del momento en el cual fue creado.
- **Date(long milseg)** crea un objeto Date teniendo por valor el día, mes, años y hora tomando el momento *milseg* trancurrido desde el 1 de Enero de 1970.

El método:

- **getTime()** devuelve un valor long de un objeto Date, y son los miliseguntos trancurrido desde el 1 de Enero de 1970

En el ejemplo siguiente, usamos el constructor Date() para crear una fecha basada en su ejecución y usamos el método getTime() para ver el número de milisegundos que ese objeto Date representa.

Ejemplo:

```
import java.util.*;
public class Now {   public static void main(String[] args)
{   Date now = new Date();
long nowLong = now.getTime();
System.out.println("El valor es: " + nowLong);   }   }
```

Obtendremos:

El valor es: 1349129892668

Ahora ¿cómo podemos formatear ese valor para que sea un poco mas entendible a simple vista y sin tener que tener una calculadora a mano?.

Clase DateFormat

Un propósito de la clase DateFormat es crear Strings de un objeto Date de tal forma que la fecha se pueda manejar y entender más fácilmente. Se encuntra en el paquete java.text.

Dependiendo del país, la fecha tiene diferentes formatos. En España preferimos verla como "25 de diciembre 2012", mientras que en Estados Unidos prefieren "Diciembre 25, 2012".

DateFormat es una clase abstracta no podemos hacer crear objetos (no podemos hacer new DateFormat().)

Para instanciarla se utiza el método **getDateInstance**();

```
DateFormat df = DateFormat.getDateInstance();
```

Cuando **una instancia de DateFormat es creada sin ningún parámetro** se toma el lenguaje y el formato

predeterminado.

DateFormat provee **algunas constantes** de solo lectura para que podamos usar de argumentos en el método getInstance() como por ejemplo: **SHORT, MEDIUM, LONG y FUL**L.

- **Podemos *convertir un objeto Date a String con el método format().***

Ejemplo:

```
Date now = new Date();
DateFormat df =  DateFormat.getDateInstance();
DateFormat df1 = DateFormat.getDateInstance(DateFormat.SHORT);
DateFormat df2 = DateFormat.getDateInstance(DateFormat.MEDIUM);
DateFormat df3 = DateFormat.getDateInstance(DateFormat.LONG);
DateFormat df4 = DateFormat.getDateInstance(DateFormat.FULL);
String s =  df.format(now);
String s1 = df1.format(now);
String s2 = df2.format(now);
String s3 = df3.format(now);
String s4 = df4.format(now);
System.out.println("(Default) Hoy es:" + s);
System.out.println("(SHORT)   Hoy es:" + s1);
System.out.println("(MEDIUM)  Hoy es:" + s2);
System.out.println("(LONG)    Hoy es:" + s3);
System.out.println("(FULL)    Hoy es:" + s4);
```

```
(Default) Hoy es:02-oct-2012
(SHORT)   Hoy es:2/10/12
(MEDIUM)  Hoy es:02-oct-2012
(LONG)    Hoy es:2 de octubre de 2012
(FULL)    Hoy es:martes 2 de octubre de 2012
```

Podemos utilizar otros formatos de fecha utilizando otro idioma y país. Se crea una instancia de la clase **Locale**. En el constructor se le pasa 2 parámetros que especificacan el lenguaje y pais respectivamente.

Por ejemplo:

```
Locale locIT = new Locale("it", "IT");  // Italy
Locale locPT = new Locale("pt");        // Portugal
Locale locBR = new Locale("pt", "BR");  // Brazil
Locale locIN = new Locale("hi", "IN");  // India
Locale locMEX = new Locale("es", "MX"); // México
```

Ejemplo:

```
Locale localBrasil = new Locale("pt","BR");
DateFormat df =
            DateFormat.getDateInstance(DateFormat.LONG, localBrasil);
Date fecha = new Date();
System.out.println(df.format(fecha));
```

- **Parsear un String**

Podemos usar la clase DateFormat para **crear objetos Date desde un String**, utilizando el **método parse()**.

Este método en particular puede lanzar una excepción ParseException, por lo tanto tenemos que capturarla.

Un ejemplo:

```
import java.util.*;
 import java.text.*;
```

```
public class ParseExample
{ public static void main(String[] args) {
String ds = "November 1, 2012";
DateFormat df = DateFormat.getDateInstance();
 try {
       Date d = df.parse(ds); }
catch(ParseException e)
       { System.out.println("Error al pasar de String a Fecha );
} } }
```

SimpleDateFormat

Las instancias default devueltas por DateFormat muchas veces puede ser suficiente para algunos propósitos, pero no cubre todo el aspectro de posibilidades

Por ello **simpleDateFormat**, **nos permite construir nuestros propios formatos**.

Las fechas van a ser construidas tomando como **referencia un String que especificará un patrón** para su construcción.

Veamos un ejemplo sencillo con la ayuda del objeto SimpleDateFormat y usando los patrones de dd, MM y yyyy para representar el día, el mes y el año.

```
Date fecha = new Date();
 SimpleDateFormat formato = new SimpleDateFormat("dd/MM/yyyy");
String resultado = formato.format(fecha);
 System.out.println("formato de la fecha:" + resultado);
```

El patrón usado dd/MM/yyyy retorna una fecha de la manera: 02/10/2012

Letras que se pueden utilizar en los patrones:

Letter	Date or Time Component	Presentation	Examples
G	Era designator	Text	AD
y	Year	Year	1996; 96
Y	Week year	Year	2009; 09
M	Month in year	Month	July; Jul; 07
w	Week in year	Number	27
W	Week in month	Number	2
D	Day in year	Number	189
d	Day in month	Number	10
F	Day of week in month	Number	2
E	Day name in week	Text	Tuesday; Tue
u	Day number of week (1 = Monday, ..., 7 = Sunday)	Number	1
a	Am/pm marker	Text	PM
H	Hour in day (0-23)	Number	0
k	Hour in day (1-24)	Number	24
K	Hour in am/pm (0-11)	Number	0
h	Hour in am/pm (1-12)	Number	12
m	Minute in hour	Number	30
s	Second in minute	Number	55
S	Millisecond	Number	978

Para comenzar, simplemente comentar que esta clase es muy útil para presentar números por pantalla, teniendo el control en todo momento del formato que tendrán. Más adelante dedicaremos otro post a DecimalFormat, que hereda de esta primera.

NumberFormat es la clase más sencilla de las dos. Principalmente presenta los siguientes métodos:

getInstance() - Simplemente obtiene el formato del idioma actual.

getCurrencyInstance() - Igual que el anterior, pero con formato de moneda.

y sus respectivas variantes con un parámetro de tipo Locale, que sustituirá al idioma actual.

Para realizar el formato será necesario llamar al método format,

Java: Manejo y formateo de Números : NumberFormat y DecimalFormat

NumberFormat es la clase más sencilla para dar formatos a los números teniendo en cuenta el pais y el idioma. Principalmente presenta los siguientes métodos:

Al igual que DateFormat, es una clase abstracta, por lo que para instanciarla tenemos que utilizar, uno de estos dos métodos:

- **getInstance()** - Simplemente obtiene el formato del idioma actual.
- **getCurrencyInstance()** - Igual que el anterior, pero con formato de moneda.

Los anteriores métodos se le puede pasar un **parámetro de tipo Locale**, que sustituirá al idioma y pais actual.

Para realizar el formato será necesario llamar al método **format**().

Ejemplo:

```
NumberFormat nf = NumberFormat.getInstance();
System.out.println(nf.format(76543210.1234));
// Resultado: 76.543.210

// Ahora con un locale distinto
nf = NumberFormat.getInstance(Locale.ENGLISH);
System.out.println(nf.format(76543210.1234));
// Resultado: 76,543,210
   /*
 * Método getIntegerInstance, sirve para redondear números decimales.
 * Ojo, redondear, ¡no truncar!
 */
nf = NumberFormat.getIntegerInstance();
System.out.println(nf.format(123456.789));
// Resultado: 123.457

// Ahora, en Francés
nf = NumberFormat.getIntegerInstance(Locale.FRENCH);
System.out.println(nf.format(123456.789));
// Resultado: 123 457

/* Formato moneda */
nf = NumberFormat.getCurrencyInstance();
System.out.println(nf.format(12345678));
// Resultado: 12.345.678,00 €

// Ahora en dólares:
nf = NumberFormat.getCurrencyInstance(Locale.US);
System.out.println(nf.format(12345678 * 1.5023));
// Resultado: $18,546,912.06
```

DecimalFormat

La clase DecimalFormat se encuentra en el paquete java.text y nos permite mostrar los números con el formato que queramos, es decir, con cuántos decimales, si queremos punto o coma para los decimales, etc.

Un uso simple de DecimalFormat puede ser este

```
import java.text.DecimalFormat;
DecimalFormat formateador = new DecimalFormat("####.####");

// Esto sale en pantalla con cuatro decimales, es decir, 3,4324
System.out.println (formateador.format (3.43242383));
```

Símbolos para utilizar en los patrones

Symbol	Location	Localized?	Meaning
0	Number	Yes	Digit
#	Number	Yes	Digit, zero shows as absent
.	Number	Yes	Decimal separator or monetary decimal separator
-	Number	Yes	Minus sign
,	Number	Yes	Grouping separator
E	Number	Yes	Separates mantissa and exponent in scientific notation. *Need not be quoted in prefix or suffix.*
;	Subpattern boundary	Yes	Separates positive and negative subpatterns
%	Prefix or suffix	Yes	Multiply by 100 and show as percentage

Si usamos ceros en vez de #, los huecos se rellenarán con ceros.

```
import java.text.DecimalFormat;
DecimalFormat formateador = new DecimalFormat("0000.0000");
// Esto sale en pantalla con cuatro cifras enteras y cuatro decimales, es decir,
0003,4300
System.out.println (formateador.format (3.43));
```

Porcentajes

Si usamos en el patrón el signo de porcentaje %, el número se multiplicará automáticamente por 100 al presentarlo en pantalla.

```
DecimalFormat formateador = new DecimalFormat("###.##%");
 // Esto saca en pantalla 34,44%
System.out.println (formateador.format(0.3444));
```