Fechas en Java

Java tiene 2 clases principales para manejar fechas:

- java.util.Date
- java.util.Calendar

Date es una clase que representa una fecha. Un Date no está pensado para cambiar su estado (y de hecho, casi todos los métodos para cambiar modificar su estado se encuentran obsoletos), ni tiene métodos para realizarle consultas. Simplemente representa una fecha exacta en el calendario (con horas, minutos, segundos, etc).

Los Date se utilizan para contener justamente datos de una fecha. En definitiva, un Date es el tipo de datos que utilizamos para almacenar e intercambiar fechas entre los objetos.

Calendar es una clase que también representa una fecha, pero es mutable. Es decir, un Calendar permite cambiar su estado (la fecha) a través de métodos. Tiene varios métodos para operar (sumar, restar, setear campos) y consultar información de la fecha.

Los Calendar son usados, entonces, para manipular una fecha.

Obteniendo un Calendar

Lo primero necesario para utilizar un Calendar es, obviamente, obtener una instancia del mismo. Pero Calendar es una interfaz... ¿quién la implementa? Java SE trae una implementación de Calendar: *java.util.GregorianCalendar*, que es una implementación del calendario Gregoriano. El calendario Gregoriano es el que utilizamos a diario en nuestra vida. Existen otras implementaciones que se pueden bajar, como ser del calendario Juliano.

La forma más simple de obtener un Calendar es:

```
Calendar ahora = Calendar.getInstance();
```

El método *getInstance()* devuelve una instancia de *GregorianCalendar* con la fecha (y hora) del momento.

Obteniendo un Calendar a partir de un Date

Muchas veces, sin embargo, vamos a tener un Date que queremos transformar. Deberemos entonces crear un Calendar y setearle los datos del Date que tenemos, para poder manipularlo. Para esto, tendremos que hacer:

```
Calendar calendar = Calendar.getInstance();
    calendar.setTime(unDate);
```

El calendar tendrá entonces los mismos datos que el Date unDate, y podremos ahora operar y modificar la fecha. Recuerden, la modificación ocurre sobre el Calendar, y no sobre el Date (el cual es inmutable).

Obteniendo valores del Calendar

Puede ocurrir que necesitemos averiguar un dato particular de nuestro Date. Por ejemplo, qué año o mes tiene nuestro Date.

Para consultar el Calendar se utiliza el método get(field). Get recibe un parámetro que indica qué campo queremos obtener. Por ejemplo:

```
int dia = calendar.get(Calendar.DAY_OF_MONTH);   //dia del mes
    int mes = calendar.get(Calendar.MONTH);  //mes, de 0 a 11
    int anio = calendar.get(Calendar.YEAR);  //año

    int hora24 = calendar.get(Calendar.HOUR_OF_DAY); //hora en formato
24hs
    int minutos = calendar.get(Calendar.MINUTE);
    int segundos = calendar.get(Calendar.SECOND);
    int milisegundos = calendar.get(Calendar.MILLISECOND);
```

La clase Calendar tiene muchas más constantes para poder consultar varios otros datos de una fecha.

¡Recuerden que los meses en un Calendar son tratados de 0 a 11!

Cambiando valores de un Calendar

Así como existe el método get(field) para obtener datos de un Calendar, contamos con el método set(field, value) para setearle valores y modificar nuestro Calendar.

El método set(field, value) recibe 2 parámetros: el primero indica qué campo quieren modificar. El segundo, indica el valor que deseamos setearle.

Por ejemplo, para setear una fecha a "última hora del día" podrían hacer:

```
Calendar calendar = Calendar.getInstance();

    calendar.set(Calendar.HOUR_OF_DAY, 23);
    calendar.set(Calendar.MINUTE, 59);
    calendar.set(Calendar.SECOND, 59);
    calendar.set(Calendar.MILLISECOND, 999);
```

O, alternativamente, de una manera quizás un poco más prolija (sin necesidad de saber cual es el valor "máximo" para cada campo):

```
Calendar calendar = Calendar.getInstance();
```

```
    calendar.set(Calendar.HOUR_OF_DAY,
calendar.getActualMaximum(Calendar.HOUR_OF_DAY));
    calendar.set(Calendar.MINUTE,
calendar.getActualMaximum(Calendar.MINUTE));
    calendar.set(Calendar.SECOND,
calendar.getActualMaximum(Calendar.SECOND));
    calendar.set(Calendar.MILLISECOND,
calendar.getActualMaximum(Calendar.MILLISECOND));
```

Recuerden, un Calendar cambia de estado, pero no el Date con el cual lo construyeron (o que luego pueden obtener).

Operando con el Calendar

Eventualmente, es posible que querramos operar con un Calendar. Es decir, sumar o restar algún campo en particular.

Para esto existe el método add(field, value). Este método recibe 2 parámetros: el primero indica la unidad que queremos sumar, y el segundo el valor.

Por ejemplo, si quisieramos obtener la fecha del día siguiente, podríamos hacer:

```
Calendar calendar = Calendar.getInstance();
    calendar.add(Calendar.DAY_OF_MONTH, 1);
```

Si quisieramos la fecha de la semana siguiente:

```
Calendar calendar = Calendar.getInstance();
    calendar.add(Calendar.WEEK_OF_MONTH, 1);
```

Obviamente, si quisieramos restar, debemos pasar valores negativos. Por ejemplo, la fecha de hace dos horas sería:

```
Calendar calendar = Calendar.getInstance();
    calendar.add(Calendar.HOUR_OF_DAY, -2);
```

El Calendar se encarga de manejar automáticamente la magia de cambio de año, año bisiesto y demás temas.

Modo permisivo - Modo no-permisivo

Por default, el Calendar opera en modo permisivo (lenient). En este modo, el calendario permite que le seteen valores "inválidos", y los ajusta a la fecha real. Por ejemplo, setear el dia "32" al mes de enero nos ubicará en realidad en el primero de febrero.

En el modo no-permisivo (non-lenient) el Calendar tirará una java.lang.IllegalArgumentException cuando se intente setear un valor "fuera de rango".

Para deshabilitar el modo permisivo, deben hacer:

```
calendar.setLenient(false);    //deshabilita el modo permisivo
```

Depende lo que esten haciendo, es posible que les resulte útil deshabilitar el modo permisivo antes de empezar a utilizar el Calendar.

Convirtiendo un Calendar a Date

Una vez que operamos con el Calendar, lo habituarl es obtener el Date que representa la nueva fecha. Para esto, utilizan el método getTime():

```
Date fecha = calendar.getTime();
```

Recuerden que pueden seguir utilizando el Calendar, pero que las nuevas operaciones no afectarán al Date que crearon.

Validar una fecha

Se puede usar la clase *java.text.SimpleDateFormat* para validar si un String contiene una fecha válida. En este caso es importante infocar al método *simpleDateFormat.setLenient(false)* antes de llamar al método *simpleDateFormat.parse()*, como se muestra en el siguiente ejemplo:

```
/** Valida si el parámetro es una fecha con el formato "dd/MM/yyyy".
  * @return true si cumple el formato, false en caso contrario.
  */
private static boolean isFechaValida(String fechax) {
  try {
      SimpleDateFormat formatoFecha = new SimpleDateFormat("dd/MM/yyyy",
Locale.getDefault());
      formatoFecha.setLenient(false);
      formatoFecha.parse(fechax);
  } catch (ParseException e) {
      return false;
  }
  return true;
}
```

//////////

Comparacion entre fechas

```
GregorianCalendar calDesde = new GregorianCalendar();
GregorianCalendar calHasta = new GregorianCalendar();

if (calDesde.after(calHasta)) {
     //La fecha de calDesde es > que la fecha de calHasta
}

if (calDesde.before(calHasta)) {
```

```
    //La fecha de calDesde es < que la fecha de calHasta
}
```

Ejemplo para comprobar resultados:

```
package javaapplication1;

import javax.swing.*;

import java.util.*;

public class Main {

   public String ingreso;

   public float numing;

   public int decision;

   public static void main(String[] args) {

   Calendar ahora = Calendar.getInstance();

   Date fecha = new Date();

   fecha.setHours(5);

   fecha.setMinutes(35);

   fecha.setSeconds(23);

   System.out.println("instance: "+ahora);

   Calendar calendar = Calendar.getInstance();

   calendar.setTime(fecha);

   System.out.println("calendar:"+calendar.getTime()+"el mes es 3, los meses se numeran
desde 0");

   int dia = calendar.get(Calendar.DAY_OF_MONTH);   //dia del mes

   int mes = calendar.get(Calendar.MONTH);  //mes, de 0 a 11

   int anio = calendar.get(Calendar.YEAR);  //año
```

```
    int hora24 = calendar.get(Calendar.HOUR_OF_DAY); //hora en formato 24hs
   int minutos = calendar.get(Calendar.MINUTE);
   int segundos = calendar.get(Calendar.SECOND);
   System.out.println(dia+"/"+mes+"/"+anio);
   System.out.println(hora24+":"+minutos+":"+segundos);
        }
}
```