Anexo I.- Fechas en Java

Para obtener la fecha actual y, en general, para trabajar con fechas en Java tenemos varias posibilidades y dependen de qué, cómo y dónde queramos representar, deberemos hacer uso de unas clases u otras. Pero antes de continuar, tiene que quedar claro que el manejo de estructuras de datos que representan fechas ha sido históricamente un problema en Java, sobre todo, la gestión de las zonas horarias. Si nuestra aplicación requiere representar las fechas en diferentes zonas horarias con exactitud deberíamos considerar utilizar la librería **Joda Time** (https://www.joda.org/joda-time/). Al no ser parte del lenguaje no vamos a utilizarla, pero conviene que le echéis un vistazo.

Date

Es la clase más simple para representar Fechas en Java. Tenemos tres constructores:

- Date(). Permite crear un objeto de tipo Date con la fecha y hora actuales del sistema.
- Date(int year, int month, int day). Permite crear un objeto de tipo Date a partir del año, el mes y día indicados como parámetro. Este método está obsoleto (Deprecated) y no se debería utilizar.
- Date(long date). Permite crear un objeto de tipo Date a partir del número de milisegundos indicado como parámetro, que han pasado desde el 1 de Enero de 1970 a las 00:00:00.

Como a métodos más significativos tenemos:

- int getHours(). Deprecated.
- int getMinutes(). Deprecated.
- int getSeconds(). Deprecated.
- void setHours(int i). Deprecated.
- void setMinutes(int i). Deprecated.
- void setSeconds(int i). Deprecated.
- void setTime(long date). Permite establecer la fecha actual indicando un número de milisegundos.
- String toString(). Convierte la fecha a un String en formato yyyy-mm-dd.

GregorianCalendar

Si deseamos más opciones, como por ejemplo sumar y/o restar fechas, la clase GregorianCalendar es una buena opción. Se trata de una subclase de la clase Calendar, que nos proporciona un calendario gregoriano estándar.

```
GregorianCalendar fecha = new GregorianCalendar();
```

Si queremos utilizar GregorianCalendar debemos hacer el siguiente import:

```
import java.util.GregorianCalendar;
```

Algunos de sus constructores son:

- GregorianCalendar(). Construye un objeto GregorianCalendar a partir de la fecha y hora del sistema con la zona horaria y localización por defecto.
- GregorianCalendar(int year, int month, int dayOfMonth). Construye un objeto GregorianCalendar a

partir del año, mes y día indicados como parámetro con la zona horaria y localización por defecto.

- GregorianCalendar(int year, int month, int dayOfMonth, int hourOfDay, int minute, int second). Construye un objeto GregorianCalendar a partir del año, mes, día, hora, minutos y segundos indicados como parámetro con la zona horaria y localización por defecto.
- GregorianCalendar(Locale aLocale). Construye un objeto GregorianCalendar con la representación correspondiente a la localización indicada como parámetro a partir de la fecha y hora del sistema con la zona horaria por defecto.
- GregorianCalendar(TimeZone zone). Construye un objeto GregorianCalendar con la zona horaria indicada como parámetro a partir de la fecha y hora del sistema con la localización por defecto.

Una vez creado el objeto que contiene la fecha del sistema, para obtener un campo concreto de la fecha se utiliza el método get(campo) donde campo indica la parte de la fecha que se quiere obtener. Cada una de estas partes está definida como una constante de la clase Calendar.

Por ejemplo, para obtener el año de la fecha actual podemos ejecutar las siguientes instrucciones:

```
GregorianCalendar fecha = new GregorianCalendar();
int año = fecha.get(Calendar.YEAR);
```

Ejemplo: programa Java que obtiene la fecha y hora actual del sistema y la muestra por pantalla. El programa mostrará la hora, el mes, el día, la hora, los minutos y los segundos.

```
import java.util.*;
public class Fechas {
   public static void main(String[] args) {
      //Instanciamos el objeto Calendar
      //en fecha obtenemos la fecha y hora del sistema
      GregorianCalendar fecha = new GregorianCalendar();
      //Obtenemos el valor del año, el mes y el día
      //hora, minuto y segundo del sistema
      //empleando el método get y el parámetro correspondiente
      int anyo = fecha.get(GregorianCalendar.YEAR);
      int mes = fecha.get(GregorianCalendar.MONTH);
      int dia = fecha.get(GregorianCalendar.DAY_OF_MONTH);
      int hora = fecha.get(GregorianCalendar.HOUR_OF_DAY);
      int minuto = fecha.get(GregorianCalendar.MINUTE);
      int segundo = fecha.get(GregorianCalendar.SECOND);
      System.out.println("Fecha actual: " + dia + "/" + (mes+1) + "/" + anyo);
      System.out.println("Hora actual: " + hora + ":" + minuto + ":" +segundo);
   }
}
```

El programa muestra la fecha y hora actual de la siguiente forma:

```
Fecha actual: 10/1/2020
Hora actual: 10:20:58
```

Tenemos que tener en cuenta que el valor devuelve por Calendar.MONTH correspondiendo al mes empieza por 0, es decir, el valor 0 corresponde a Enero y el valor 11 corresponde a Diciembre, por lo tanto deberemos sumarle 1 a este valor antes de operar con él.
Además de obtener la fecha, también podemos asignar valores a un campo determinado de la fecha. Para asignar valores a campos concretos o a fechas completas se utiliza el método set:
Ejemplo:

```
fecha.set(2020,5,1); //asigna año, mes, día
fecha.set(2020,5,1,3,45,12); //asigna valor a todos los campos
fecha.set(GregorianCalendar.YEAR,2020); //asigna el valor 2020 al año
```

También podemos sumar o restar valores a la fecha y hora.
Ejemplo: sumar 10 días a la fecha actual:

```
fecha.add(GregorianCalendar.DAY_OF_MONTH, 10);
```

Ejemplo: sumar 10 minutos a la hora actual:

```
fecha.add(GregorianCalendar.MINUTE, 10);
```

Ejemplo: restar 2 años a la fecha actual:

```
fecha.add(GregorianCalendar.YEAR, -2);
```

Otro método que nos puede interesar es:

- Date getTime(). Permite obtener un objeto de tipo Date a partir de un GregorianCalendar.

SimpleDateFormat

Para darle formato a la salida de una fecha tenemos varias opciones, pero una de las más utilizadas es **SimpleDateFormat**. Esta clase permite especificar con qué formato queremos mostrar la fecha. Debe quedar claro que la fecha no es modificada para nada, simplemente su visualización.
Constructores:

- SimpleDateFormat(String pattern). Dónde pattern es un String (patrón de representación) que indica como se deberá mostrar la fecha. Vemos algunos de los patrones (cuidado con las mayúsculas y minúsculas):
  - y: Año
  - M: Mes
  - w: Semana del año
  - W: Semana del mes
  - d: día del mes
  - D: día del año
  - m: minutos
  - s: segundos
  - S: milisegundos

  Un ejemplo podría ser:

```
SimpleDateFormat fecha = new SimpleDateFormat("dd/MM/yyyy hh:mm:ss");
```

Como método más destacado tenemos:

- `String format(Date fecha)`. Que recibe un objeto de tipo Date y obtiene su representación según el pattern indicado en el constructor.

Veamos un ejemplo completo:

```java
import java.util.*;
import java.texto.SimpleDateFormat;
public class Calendario {

   public static void main(String[] args) {

      //Qué día de la semana fue NAVIDAD en el año 2005
      Calendar navidad2005 = new GregorianCalendar(2005, Calendar.DECEMBER, 25);
      int diaSetmana = navidad2005.get(Calendar.DAY_OF_WEEK);
      System.out.println("Día (1=Domingo): " + diaSetmana );

      //Cuántos días tuvo febrero de 2005
      Calendar febrero2005 = new GregorianCalendar(2005, Calendar.FEBRUARY, 1);
      int diesFebrer = febrero2005.getActualMaximum(Calendar.DAY_OF_MONTH);
      System.out.println("dias en Feb 2005: " + diesFebrer );

      //Cuántos días tuvo febrero de 2000
      Calendar febrero2000 = new GregorianCalendar(2000, Calendar.FEBRUARY, 1);
      diesFebrer = febrero2000.getActualMaximum(Calendar.DAY_OF_MONTH);
      System.out.println("Días en Feb 2000: " + diesFebrer );

      //Obtener la diferencia en milisegundos
      long diffMillis = navidad2005.getTimeInMillis() - febrero2005.getTimeInMillis();

      //Obtener la diferencia en segundos
      long diffSecs = diffMillis/1000;

      //Obtener la diferencia en minutos
      long diffMins = diffSecs/60;

      //Obtener la diferencia en horas
      long diffHours = diffMins/60;

      //Obtener la diferencia en dias
      long diffDays = diffHours/24;
      //Obtener la diferencia en meses
      long diffMonths = diffDays/30;

      System.out.println("diffMillis: " + diffMillis );
      System.out.println("diffSecs: " + diffSecs );
      System.out.println("diffMins: " + diffMins );
      System.out.println("diffHours: " + diffHours );
      System.out.println("diffDays: " + diffDays );
      System.out.println("diffMonths: " + diffMonths );

      Calendar fecha = new GregorianCalendar();
      // Obtenemos el valor del año, mes y día .
      int anyo = fecha.get(Calendar.YEAR);
      int mes = fecha.get(Calendar.MONTH)+1;
      int dia = fecha.get(Calendar.DATE);
```

```
        int hora = fecha.get(Calendar.HOUR);
        int minuto = fecha.get(Calendar.MINUTE);
        int segundo = fecha.get(Calendar.SECOND);

        System.out.println("AÑO ACTUAL: " + anyo);
        System.out.println("MES ACTUAL: " + mes);
        System.out.println("DÍA ACTUAL: " + dia);


        fecha.set(Calendar.YEAR, 2020);
        fecha.set(Calendar.MONTH, 02); //los meses empiezan por 0.
        fecha.set(Calendar.DAY_OF_MONTH, 30);
        // Asignamos año, mes y dia.
        fecha.set(2020, 02, 30);
        // Asginamos año, mes, dia, hora y minutos
        fecha.set(2020, 02, 30, 19, 00);

        // Añadimos 30 minutos a la fecha actual.
        fecha.add(Calendar.MINUTE, 30);
        // Añadimos 100 días a la fecha actual.
        fecha.add(Calendar.DATE, 100);
        // Quitamos 10 años a la fecha actual.
        fecha.add(Calendar.YEAR, -10);
        System.out.println("AÑO ACTUAL: " + anyo);

        System.out.println("FECHA ACTUAL: " + día + "/" + (mes) + "/" + anyo);
        System.out.printf("Hora Actual: %02d:%02d:%02d %n", hora, minuto, segundo);

        //Imprimir la fecha y hora actuales FORMA 2
        Date ahoraMismo = new Date();
        SimpleDateFormat dataImprimir = new SimpleDateFormat("dd/MMMMM/yyyy hh:mm:ss");
        String longFechaActual = dataImprimir.format(ahoraMismo);
        System.out.println(longFechaActual);
    }
}
```

LocalDate, LocalTime, LocalDateTime, DateTimeFormatter y Period

Desde la versión 8 de Java podemos encontrar en el paquete **java.time** la definición de estas 3 clases.

LocalDate

Es un clase inmutable que representa una fecha sin tiempo. Habitualmente en formato yyyy-MM-dd.

Para utilizarla primero debemos importar la clase:

```
import java.time.LocalDate;
```

Para obtener la fecha actual podemos utilizar el método estático ***now()***

```
LocalDate hoy = LocalDate.now();
```

Para crear una fecha específica podemos utilizar el método estático ***of()***

```
LocalDate hoy = LocalDate.of(2024, 02, 01);
```

Podemos comparar dos LocalDate con los métodos ***isBefore()*** y ***isAfter()***

```
boolean antes = hoy.isBefore(fecha);
boolean despues = hoy.isAfter(fecha);
```

Podemos darle formato a un LocalDate mediante la clase **DateTimeFormatter**

```
DateTimeFormatter dateTimeFormatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
String fechaFormateada = hoy.format(dateTimeFormatter);
```

Podemos crear un LocalDate a partir de un String utilizando un **DateTimeFormatter**

```
LocalDate otraFecha = LocalDate.parse("05/02/2024", dateTimeFormatter);
```

Period

Esta clase permite crear periodos de tiempo para operar con fechas (sumar y restar periodos de tiempo) y calcular diferencias de tiempo.
El método **of(years, months, days)** permite obtener un periodo expresado en años, meses y días.

```
Period periodo = Period.of(0, 2, 3);
LocalDate fecha = LocalDate.now();
LocalDate nuevaFecha = fecha.plus(periodo);
LocalDate otraFecha = fecha.minus(periodo);
```