Uso de la clase **GregorianCalendar**

Se quedó para el final el método que inicia una transmisión, porque la interfaz dice que este método va a responder con una cadena que va a incluir la hora y fecha en que se otorga la transmisión, de haber transmisiones disponibles. Para saber la hora y día en que sucede un evento tienes que recurrir a la hora y fecha que tiene registrado el sistema operativo (o la computadora misma). Esta información (este servicio) te lo proporciona una clase de Java llamada GregorianCalendar, que también se encuentra en el paquete java.util. Para poder usarla tienes que avisar que la vas a importar, junto a la importación de java.util.Scanner, como sigue:

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

GregorianCalendar es una clase que extiende a (hereda de) Calendar, por lo que al dar el resumen de lo que pudieras usar de esta clase, no va a aparecer en la documentación de Java8 bajo esta clase, sino que la mayoría de los campos van a aparecer en la clase Calendar. Vale la pena mencionar que si bien se trata de enteros estáticos (de clase), mientras que algunos de ellos simplemente indican la posición del atributo que contiene ese valor o cómo obtenerlo, hay otros que son constantes de la clase. Se están manejando estos enteros como constantes simbólicas. Los valores para los distintos campos de un calendario se establecen cuando se construye un ejemplar o cuando se invoca a alguno de los métodos que cambian el valor de uno o varios de los campos. Se habla de "campos" y no de "atributos" porque tienes acceso a los valores a través de las constantes simbólicas que los eligen. No se listan todos los campos, sino únicamente aquellos que podrías utilizar. Cuando se trata de la indicación de un campo tienes que usar el método get de GregorianCalendar, mientras que si se trata de una constante de clase, lo puedes usar directamente.

Tabla 1 Algunas constantes simbólicas de la clase GregorianCalendar (1/2)

Modificador y tipo	**Atributo y descripción**
static int	AM, PM **Constantes** de clase. Indican el periodo del día desde la medianoche hasta inmediatamente antes que el mediodía y desde el mediodía hasta inmediatamente antes de media noche, respectivamente.
static int	AM_PM **Campo** para get y set. Contiene el periodo del día (AM o bien PM).
static int	January. . . Decemeber **Constantes** de clase. Valor del mes correspondiente a la fecha (de 1 a 12 respectivamente).
static int	DATE **Campo** numérico para set y get indicando el día del mes.
static int	DAY_OF_MONTH **Campo** para set y get indicando el día del mes.
static int	DAY_OF_WEEK **Campo** numérico para indicar el día de la semana, usando get y set.

Tabla 1 Algunas constantes simbólicas de la clase GregorianCalendar (2/2)

Modificador y tipo	Atributo y descripción
static int	DAY_OF_WEEK_IN_MONTH **Campo** numérico para indicar el ordinal del día de la semana dentro del mes.
static int	DAY_OF_YEAR **Campo** numérico para get y set indicando el número de día dentro del año.
static int	HOUR **Campo** numérico para get y set. Indica la hora de la mañana o de la tarde
static int	HOUR_OF_DAY **Campo** numérico para get y set que indica la hora del día en 24 horas.
protected boolean	isTimeSet **Atributo** booleano: True si la hora es válida.
static int	MILLISECOND **Campo** numérico para get y set que indica el milisegundo dentro del segundo.
static int	MONTH **Campo** numérico para get y set que indica el número ordinal del mes dentro del año (JANUARY... DECEMBER) dentro del año (0... 11).
static int	SUNDAY... SATURDAY **Constantes** de clase con los valores de 1 a 7 respectivamente.
static int	SECOND **Campo** numérico para indicar el segundo dentro del minuto (0... 59).
protected long	time **Atributo**. El tiempo establecido en este momento para este calendario, expresado en milisegundos transcurridos desde el 1º de enero de 1970, 0:00:00 GMT. Pero tomando en cuenta que no es público, no lo podrán usar directamente en sus proyectos.
static int	YEAR Campo numérico para get y set indicando el año.

Hay dos atributos de objeto entre los que se listaron y son **protected long** time, que corresponde al atributo donde se almacena, precisamente, la fecha "actual" en el momento de construir el objeto; y el atributo isTimeSet que indica si el objeto usado tiene o no el tiempo establecido.

Todos estos atributos son, en realidad, de la clase Calendar, que es abstracta (no está completamente definida) por lo que no podemos, directamente, construir objetos de ella. Por eso estamos usando GregorianCalendar que sí es una clase completamente definida. GregorianCalendar tiene la posibilidad de representar fechas y horas en el calendario Juliano, pero esto no es de nuestro interés en este momento. Asimismo, cuenta con dos atributos adicionales (AD y BC) pero tampoco los necesitamos, así que no los revisaremos.

Tenemos un valor por omisión para algunos de los campos indicados por por los atributos estáticos de la clase Calendar (que se copian a GregorianCalendar). Listamos a continuación los que vamos a usar y que ya mencionamos.

Tabla 2 Valores por omisión en la clase GregorianCalendar

Campo	Valor por omisión
YEAR	1970
MONTH	JANUARY
DAY_OF_MONTH	1
DAY_OF_WEEK	primer día de la semana de este año
DAY_OF_WEEK_IN_MONTH	1
AM_PM	AM
HOUR, HOUR_OF_DAY, MINUTE, SECOND, MILLISECOND	0

Los atributos que se marcaron como **campo** se tienen que usar con el método get para consultar y set para actualizar, usando como argumento el campo. Los atributos marcados como constantes son constantes de clase, por lo que se pueden usar precediéndolos del identificador de la clase o de un objeto de la clase.

Por ejemplo, el que se encuentra en la posición marcada por YEAR es 1970. Si preguntas por GregorianCalendar.YEAR directamente, el valor que te da es 1, mientras que si preguntas por GregorianCalendar.AM_PM te dice 9. Esto quiere decir que estos atributos indican no un valor sino un lugar, que es el mismo para todos los objetos de esta clase.

Para los valores que no están listados no aplican valores por omisión. Recuerda que los valores en mayúsculas son constantes simbólicas, por lo que para usarlas deberán ir precedidas del nombre de la clase o de un objeto de la clase; si no se hace así dará error de sintaxis, pues no de trata de valores de la clase **Disco**. Para aquellas constantes que identifican nombres simbólicos de, por ejemplo, meses o días, contienen un valor entero. El de January es 0, mientras que el valor de SUNDAY es 1. También hay forma de obtener los nombres de los meses en inglés, como cadenas, pero no los utilizarás en este proyecto.

Constructores de la clase GregorianCalendar

Veremos únicamente dos constructores, aunque la clase tiene siete. El primero de ellos es el constructor sin parámetros:

```
GregorianCalendar()
        Construye un calendario gregoriano con el valor de tiempo en el momento
         en que el objeto se construye.
```

El segundo de ellos que se lista es aquel al que le puedes dar todos los datos desmenuzados. En realidad, todos los constructores que tienen los primeros tres parámetros y agregan, en orden, uno o más de los parámetros que siguen, son constructores válidos.

```
GregorianCalendar(int year, int month, int dayOfMonth,
                  int hourOfDay, int minute, int second)
        Construye el objeto tomando como valores de inicialización los datos dados.
```

Como GregorianCalendar extiende a Calendar, a continuación se describen aquellos métodos de este último que te pueden ser útiles para tu proyecto. Todos estos métodos son de objeto, por lo que tendrán que ir precedidos del **objeto** de la clase GregorianCalendar al que se refiere, mientras que los atributos son, en general, **constantes de la clase** GregorianCalendar, por lo que tendrán que ir precedidos del nombre de la clase o de un objeto de esta clase.

Tabla 3 Algunos métodos útiles de Calendar

Modificador y tipo	**Firma y descripción**
int	get(int campo) Regresa el valor numérico contenido en el campo especificado.
int	getActualMaximum(int campo) Regresa el valor numérico máximo que puede tomar el contenido del campo en este calendario.
int	getActualMinimum(int campo) Regresa el valor numérico mínimo que puede tomar el contenido del campo en este calendario.
long	getTimeInMillis() Regresa el valor del calendario **this** en milisegundos.
void	set(int campo, int valor) Actualiza el valor del contenido del campo a valor.
void	set(int year, int month, int date,int hourOfDay, int minute, int second) De manera parecida al constructor, toma como mínimo los tres primeros parámetros para actualizar su valor y puede agregar, en orden, uno, dos o tres parámetros adicionales. El nombre del campo indica cuál es el lugar que ocupa el atributo que va a modificar.
String	toString() Da el calendario en forma de cadena, dando el nombre de cada uno de los campos y su valor.

Ahora agregaremos aquellos métodos adicionales, definidos en GregorianCalendar, que te interesarán:

Tabla 4 Algunos métodos de GregorianCalendar adicionales a los de Calendar

Modificador y tipo	**Firma y descripción**
void	**add**(int campo, int cantidad) Suma al contenido del campo la cantidad.
boolean	**isLeapYear**(int year) Responde verdadero o falso a si el argumento (year) dado es bisiesto o no. Puede ser una constante o bien el contenido del campo YEAR del objeto.

Veamos algunos ejemplos del uso de esta clase, sin olvidar que tuviste que importarla.

```
1   GregorianCalendar gc = new GregorianCalendar();
2       /* Fecha en este momento */
3   System.out.println("Primera: \t"+gc.get(gc.MILLISECOND)
4                      + "\tmes="+gc.get(gc.MONTH));
5       /* Crear nuevo objeto con nueva fecha */
6   gc = new GregorianCalendar();
7   System.out.println("Segunda: \t"
8                      + gc.get(GregorianCalendar.MILLISECOND));
9       /* Cambiar algunos atributos */
10  gc.set(1976,gc.MAY,23, 10,30,27);
11  System.out.println("Tercera: \t"+gc.get(gc.MILLISECOND)
12                     +"\tmes="+gc.get(gc.MONTH));
```

El resultado se encuentra a continuación. Nota que en el renglón **??** se cambió el atributo del campo en la posición MONTH, pero no el del campo MILLISECOND. Como todo sucede muy rápido en la computadora, los dos últimos renglones tienen el mismo valor para este campo. Si volvieras a ejecutar estas líneas, te podrían salir valores distintos.

```
Primera:        666     mes=9
Segunda:        668
Tercera:        668     mes=4
```

Como puedes observar, el número de milisegundos que transcurren desde que imprime la computadora la primera línea hasta que imprime la segunda es de 2 milisegundos y en algunas ocasiones ni siquiera cambia. Como no se le cambió el valor de milisegundos ni se construyó otro objeto, este valor permanece igual para la tercera impresión. Entre la primera y segunda impresión, como se está construyendo un objeto nuevo en cada ocasión, si estas construcciones estuviesen más separadas con código podría haber un cambio mayor. Cada vez que lo ejecutemos terminaremos con otra respuesta. Al ejecutar este código (separando con código las dos construcciones de objetos pero sin mostrarlo) obtenemos el siguiente resultado:

```
Primera:        416     mes=9
Segunda:        422
Tercera:        422     mes=4
```

Nota que el campos para el mes –gc.get(gc.MONTH)– varía de 0 para enero hasta 11 para diciembre.