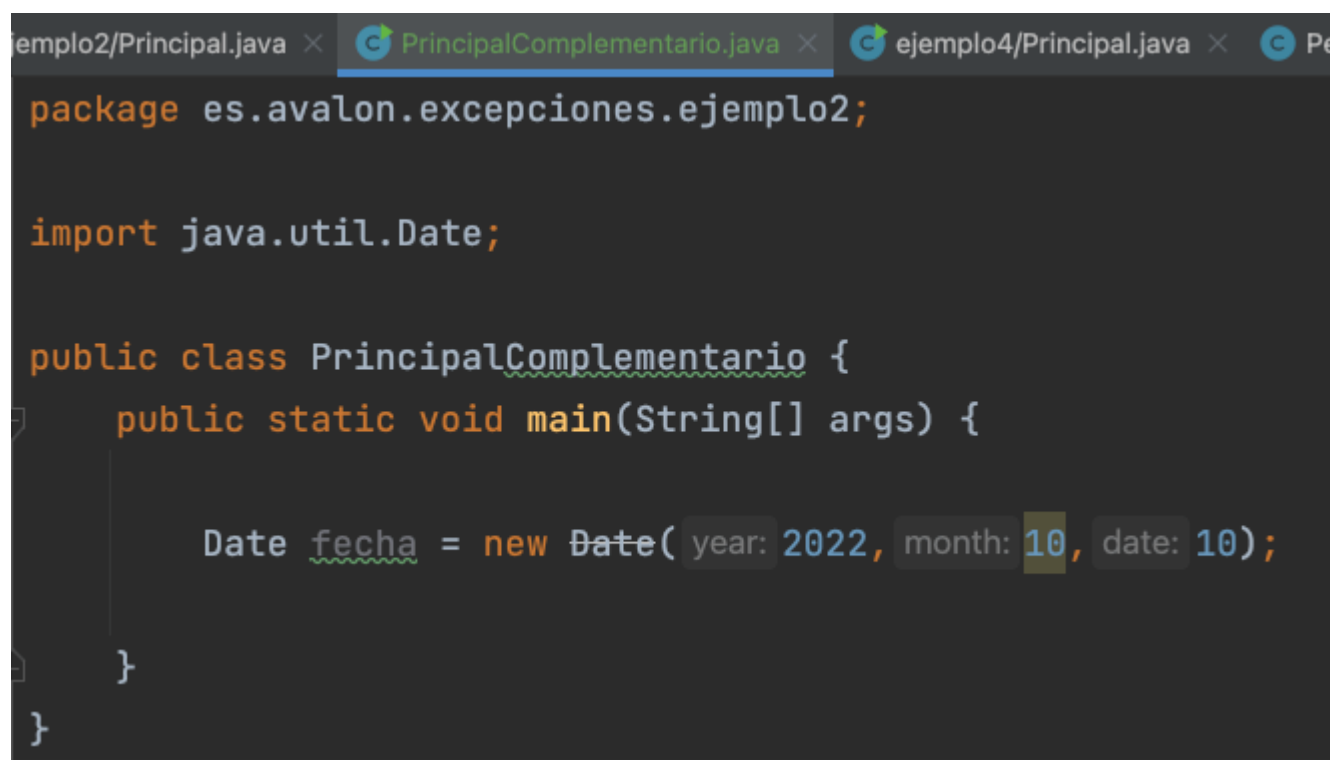


El uso de Java Calendar sigue estando muy al día . Es verdad que el API de Java 8 aporta un manejo de fechas mucho más moderno . Pero las miles de aplicaciones clásicas de Java siguen dependiendo mucho de la clase Date y de la clase Calendar . Estas clases siempre dan muchos quebraderos de cabeza cuando uno comienza a usar el lenguaje . Vamos a hablar un poco de ellas y entender mejor su manejo. Para ello lo primero que tenemos que hacer es construir una clase Date y darnos cuenta que en cuanto empezamos a construirla comienzan los problemas ya que la mayoría de sus constructores están deprecados.



```
package es.avalon.excepciones.ejemplo2;

import java.util.Date;

public class PrincipalComplementario {
    public static void main(String[] args) {

        Date fecha = new Date( year: 2022, month: 10, date: 10);

    }
}
```

Acabamos de usar un constructor que admite el año el mes y el día y directamente el API de Java no recomienda su uso y considera que esta obsoleto. Este es uno de los problemas más habituales que tenemos cuando manejamos fechas .El API de Java clásico se encarga de dividir las responsabilidades a la hora de manejar fechas . Cuando uno tiene que manejar fechas existen tres responsabilidades fundamentales

1. Almacenar el valor de la fecha (Java Date)
2. aplicar diferentes formatos a la fecha (Java DateFormat)
3. hacer cálculos de fechas (Java Calendar)

Son demasiadas responsabilidades para ubicarlas en una única clase por lo tanto Java lo divide en tres :

1. Date : Almacena la fecha
2. DateFormat y SimpleDateFormat : Aplica formatos
3. Java Calendar: Realiza cálculos sobre la fecha

Vamos a ver un ejemplo práctico :

```
package es.avalon.excepciones.ejemplo2;
```

```
import java.text.DateFormat;  
import java.text.ParseException;  
import java.text.SimpleDateFormat;  
import java.util.Calendar;  
import java.util.Date;  
import java.util.GregorianCalendar;
```

```
public class PrincipalComplementario {  
    public static void main(String[] args) {  
  
        //define el formato de la fecha  
        DateFormat formateador= new SimpleDateFormat("dd/M/yy");  
  
        try {  
            // convierte un String en formato fecha en una fecha real  
            Date fecha= formateador.parse("10/01/2022");  
  
            // creamos un calendario  
            Calendar calendario= new GregorianCalendar();  
  
            //hacemos calculos sobre el calendario
```

```

        calendario.setTime(fecha);
        //movemos el calendario
        calendario.add(Calendar.DATE,5);

        //usamos el formateador y volvemos a mostrar la fecha
        System.out.println(formateador.format(calendario.getTime()));

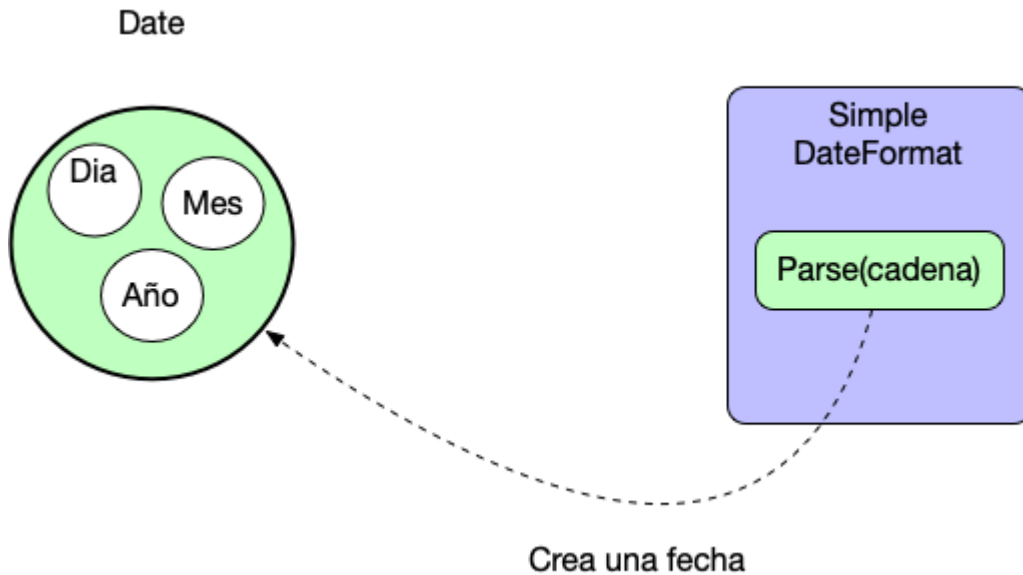
    } catch (ParseException e) {
        e.printStackTrace();
    }
}
}

```

El código es difícil de procesar en una primera instancia. Vamos a explicarlo línea a línea:

```
DateFormat formateador= new SimpleDateFormat("dd/M/yy");
```

En este caso estamos ante un formateador sencillo que se encarga de definir un formato para manipular una fecha . En este caso Día/Mes/año . Podemos ver estos formatos más a detalle a nivel de [JavaDoc en la clase SimpleDateFormat](#). Definida esta clase podemos invocar al método `parse` que se encarga de convertirla en una fecha y almacenarlo en un objeto `Date`.

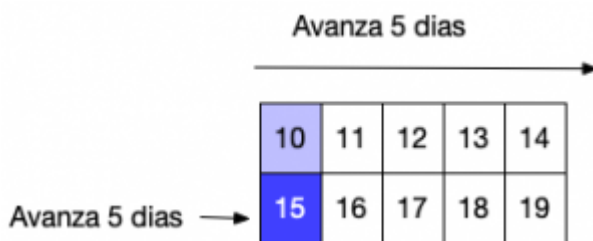


Java Calendar

Una vez creada una fecha , esta simplemente almacena la información . Si queremos realizar operaciones sobre ella lo que tenemos que construir es un Java Calendar. Una vez construido el calendario le asignamos nuestra fecha.

```
// creamos un calendario
Calendar calendario= new GregorianCalendar();
//asignamos la fecha al calendarijo
calendario.setTime(fecha);
```

Con la fecha asignada podemos realizar operaciones sobre ella como por ejemplo avanzar 5 días el calendario:



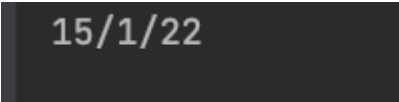
Veamoslo:

```
calendario.add(Calendar.DATE,5);
```

Una vez movido el calendario , podemos solicitar al formateador que nos vuelva a mostrar la fecha utilizando el mismo formato por la consola:

```
//usamos el formateador y volvemos a mostrar la fecha  
System.out.println(formateador.format(calendario.getTime()));
```

Si ejecutamos el código veremos la fecha impresa con día 15:



```
15/1/22
```

Otros artículos relacionados

1. [Java Comparable Interface y Ordenaciones](#)
2. [¿Que es un Java Bean?](#)
3. [Java String to Date utilizando Java 8](#)

**CURSO JPA
GRATIS
APUNTATE!!**