Useful Classes

`Date, SimpleDateFormat ,Calendar, Math, Random, StringTokenizer`

Objectives

- Evaluate different types of ultility classes in Java to know when to use which class

Table of Content

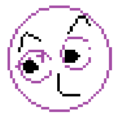
Date and Time

- To work with date and time, there are 3 important classes
  - **`Date`**
  - **`Calendar`**
  - **`GregorianCalendar`**
- All of these are in **`java.util`**
- **`GregorianCalendar`** is a subclass of **`Calendar`**
- It is recommended to use **`Calender`** class whenever possible because, most of the methods under **`Data`** class are deprecated.

java.util.Date

- A **`Date`** object is represented internally, as a single long number; which represents, the number in milliseconds since January 1, 1970, 00:00:00 GMT.

- Most of its methods are deprecated because, many of them are not amenable to internationalization. On the other hand, **`Calender`** class has replacement methods for Data class; and thus, this is the recommended.



Have a quick look at the Date methods.

Example: Date

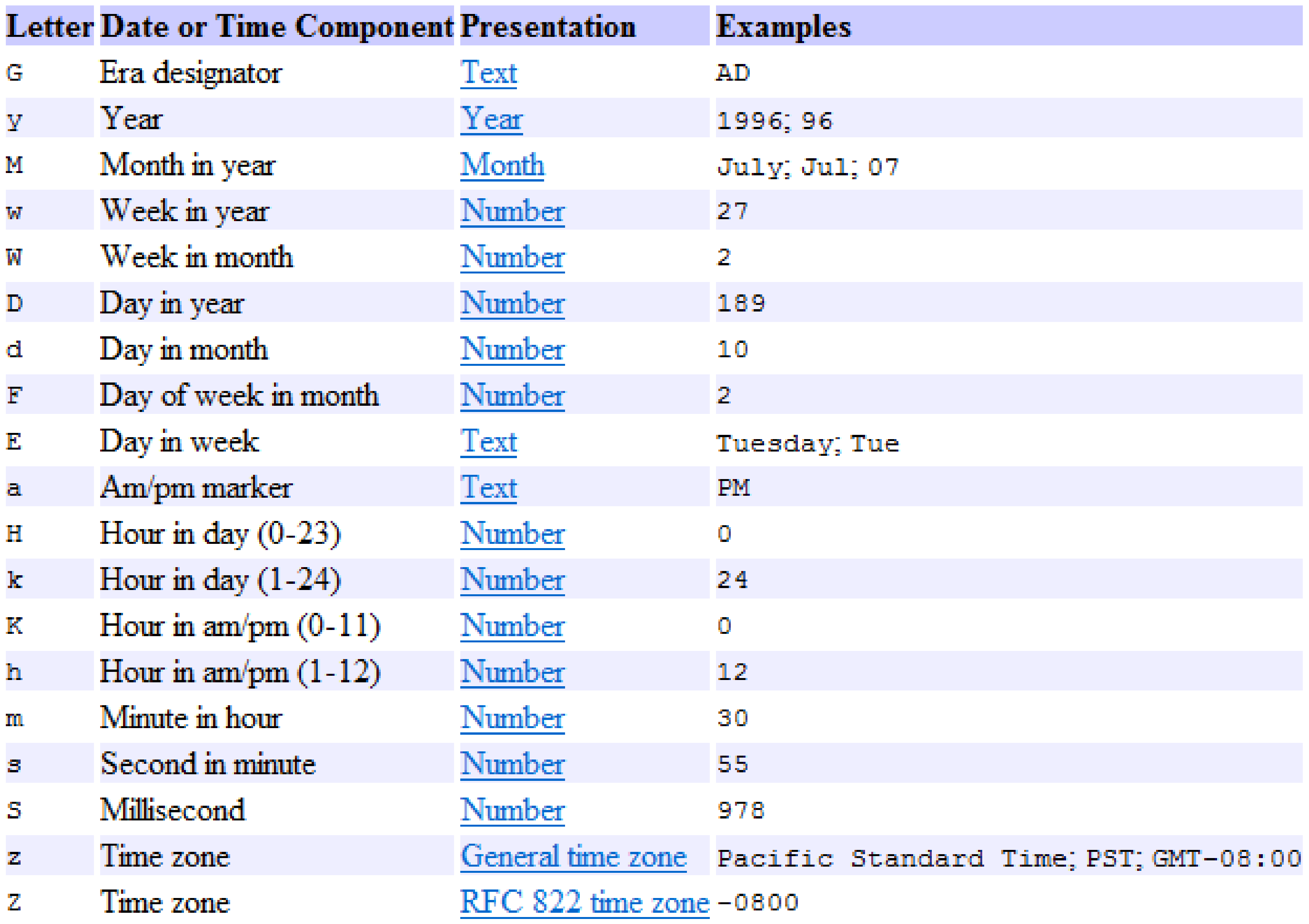
```
import java.util.Date;
public class DateEx{
public static void main(String[] args) {
   Date d= new Date();
   System.out.println(d);
   Date d1=new Date(d.getTime()+1000);
   System.out.println(d.after(d1));
   System.out.println(d.before(d1));
   System.out.println(d1.compareTo(d));
   System.out.println(d.compareTo(d1));}}
```

```
Fri Nov 18 14:14:36 IST 2011
false
true
1
-1
```

Formatting Dates

- **`java.text.SimpleDateFormat`** class is used , for formatting and parsing dates in a locale-sensitive manner.
- In the constructor, the date format can be specified using, predefined letters that correspond to some meaning.
- Constructors **:**
  - **`SimpleDateFormat()`**
    - uses the default pattern and date format symbols for the default locale.
  - **`SimpleDateFormat(String pattern)`**
    - uses the given pattern and the default date format symbols for the default locale.

Pattern letters (JSE documentation)



Letter	Date or Time Component	Presentation	Examples
`G`	Era designator	Text	`AD`
`y`	Year	Year	`1996; 96`
`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 in week	Text	`Tuesday; Tue`
`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`
`z`	Time zone	General time zone	`Pacific Standard Time; PST; GMT-08:00`
`Z`	Time zone	RFC 822 time zone	`-0800`

SimpleDateFormat Methods

- **`final String format(Date date)`**
  - Formats a Date into a date/time string, as per specification in the constructor.
- **`Date parse(String source) throws ParseException`**
  - Parses text from the beginning of the given string, to produce a date, based on the format specification in the constructor.
  - **`java.text.ParseException`** is a checked exception, which is thrown if the expected string does not match the specified format.
  - **`Calendar getCalendar()`**
  - Gets the calendar associated with this date/time formatter.

Example: SimpleDateFormat

```
import java.text.*;
import java.util.Date;
public class Text{
public static void main(String[] args) throws
ParseException {
      Date now = new Date( );
       SimpleDateFormat ft =
       new SimpleDateFormat ("E dd MMM yyyy 'at'
hh:mm:ss a zzz");
       System.out.println(ft.format(now));
       SimpleDateFormat ft1 =
       new SimpleDateFormat ("dd.mm.yyyy");
      Date d= ft1.parse("10.7.1967");
       System.out.println(t.format(d));
}}
```

```
Fri 18 Nov 2011 at 02:53:09 PM IST
Tue 10 Jan 1967 at 12:07:00 AM IST
```

Formatting character for Date in `printf`

?*We have seen how to format numbers and string using* ***`System.out.printf`*** *statement. Do you recall them?*

- **`System.out.printf`** statement can be used to display date in the desired format.
- The format characters for time and date are in the next slides.
- These characters must have prefix of 't' and 'T' conversions
- Example:

```
Date now = new Date( );
System.out.printf(" %1$tA %1$tB %1$td, %1$tY
 %1$tH:%1$tM:%1$tS %1$tp %1$tZ ",now);
```

  This prints: **`Friday November 18, 2011 15:50:02 pm IST`**

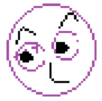
  `Calendar` and its subclass can also be used in place of **`Date`**

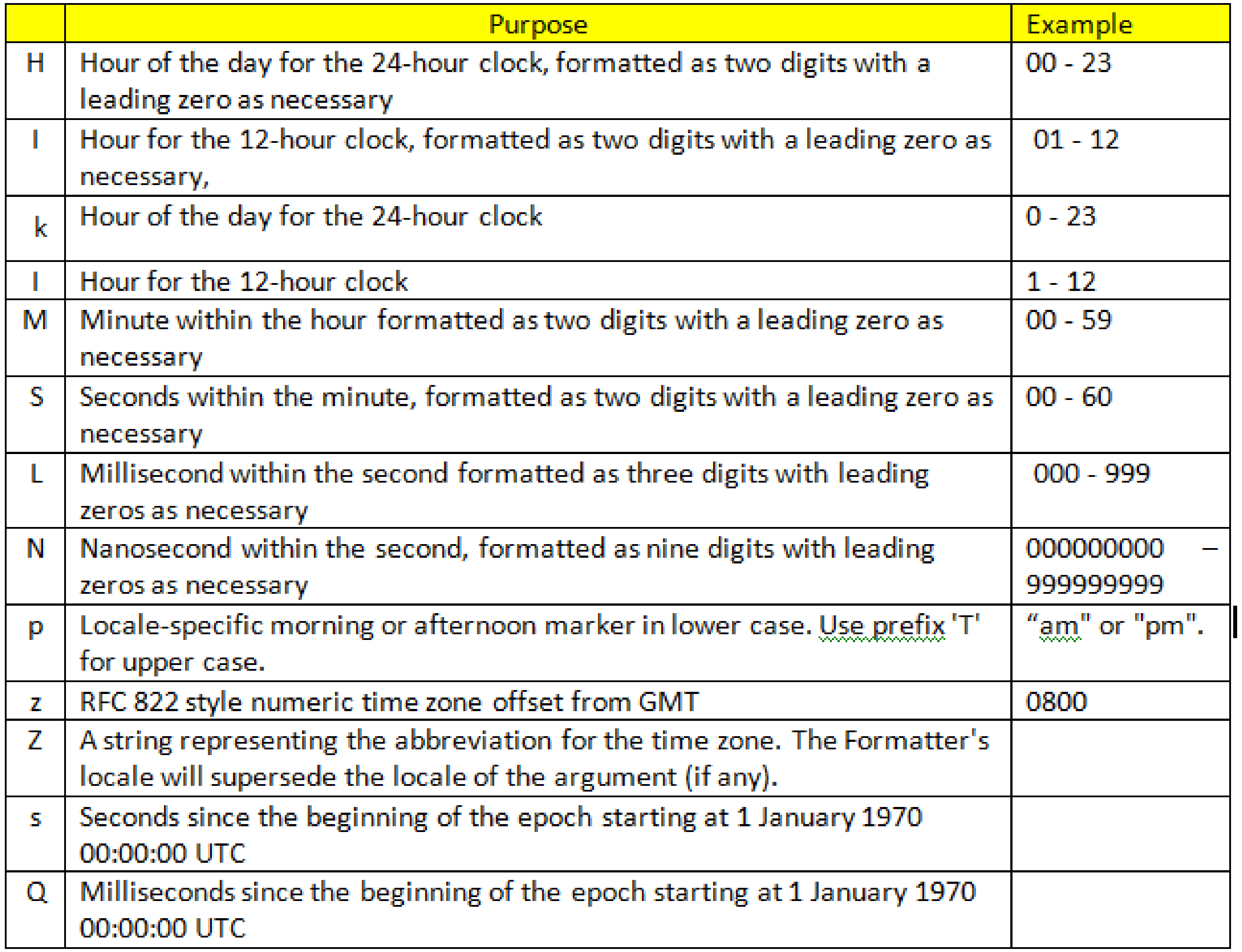
- Be careful when you use **`printf`**. Small error would cause **`UnknownFormatConversionException`** to be thrown at runtime

Date format characters (JSE)

	Purpose	Example
B	Locale-specific full month name,	"January","February"
b, h	Locale-specific abbreviated month name	"Jan","Feb"
A	Locale-specific full name of the day of the week,	"Sunday","Monday"
a	Locale-specific short name of the day of the week,	"Sun","Mon"
C	Four-digit year divided by 100, formatted as two digits with leading zero as necessary	00-99
Y	Year, formatted as at least four digits with leading zeros as necessary	, e.g. 0092 equals 92 CE for the Gregorian calendar
y	Last two digits of the year, formatted with leading zeros as necessary	00 - 99.
j	Day of year, formatted as three digits with leading zeros as necessary	001 - 366 for the Gregorian calendar
m	Month, formatted as two digits with leading zeros as necessary	01 - 12
d	Day of month, formatted as two digits with leading zeros as necessary	01 - 31
e	Day of month, formatted as two digits	1 - 31

Time format characters (JSE)





	Purpose	Example
H	Hour of the day for the 24-hour clock, formatted as two digits with a leading zero as necessary	00 - 23
I	Hour for the 12-hour clock, formatted as two digits with a leading zero as necessary,	01 - 12
k	Hour of the day for the 24-hour clock	0 - 23
l	Hour for the 12-hour clock	1 - 12
M	Minute within the hour formatted as two digits with a leading zero as necessary	00 - 59
S	Seconds within the minute, formatted as two digits with a leading zero as necessary	00 - 60
L	Millisecond within the second formatted as three digits with leading zeros as necessary	000 - 999
N	Nanosecond within the second, formatted as nine digits with leading zeros as necessary	000000000 – 999999999
p	Locale-specific morning or afternoon marker in lower case. Use prefix 'T' for upper case.	“am" or "pm".
z	RFC 822 style numeric time zone offset from GMT	0800
Z	A string representing the abbreviation for the time zone. The Formatter's locale will supersede the locale of the argument (if any).	
s	Seconds since the beginning of the epoch starting at 1 January 1970 00:00:00 UTC	
Q	Milliseconds since the beginning of the epoch starting at 1 January 1970 00:00:00 UTC	

Calendar and GregorianCalendar

- **`Calendar`** is an abstract class.
- **`GregorianCalendar`** is a concrete subclass of **`Calendar`**. This class, provides the standard calender system, used by the most in the world.
- To create an instance of **`Calendar`** class **`getInstance()`** static method is used. This a **`Calendar`** object with the system's date and time.
- In Gregorian Calender, the value is stored as time, in milliseconds represented by a value that is an offset from the *Epoch*, January 1, 1970 00:00:00.000 GMT.
- Constructor:
  - **`GregorianCalendar()`**
  - **`GregorianCalendar(int year, int month, int dayOfMonth,[int hourOfDay, int minute, int second])`**

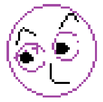
using Calendar

```
/*cal value depends on the current system date and
time.*/
Calendar cal =Calendar.getInstance();

System.out.println(cal instanceof GregorianCalendar);
// true
System.out.println(cal.getTime());
//Fri Nov 18 16:13:12 IST 2011

cal.clear();
System.out.println(cal.getTime())
//Thu Jan 01 00:00:00 IST 1970
```

`Calendar/ GregorianCalendar` members



Go through the members of `Calendar/GregorianCalendar` members for 10 minutes

Exercise

- *A library has books and each book has a number . Members can borrow only one book from the library and they must return the book within a week's time. If exceeded, Rs.10 is charged as fine for each day from then on. If not returned even after a month, Rs.50 is charged for every subsequent day. For every subsequent month, 50 * number of months is collected as fine*. .
- *At the beginning of every month, a bill is produced; for all the members, with a token amount of Rs. 100, as a mark of continuance of the membership*. *In cases where the fine is applicable, the fine amount is also added.*
- *Write a java program to implement this application. The code must be able to take inputs to create books, members, dates (issue and return) and also print the bill at the end of every month.*   (1 hour)

Exercise

- *User enters the date, in the form 7 July 2012. Display the following based on the week it falls on*

  print white for Mon

  print red for Tue

  print green for Wed

  print yellow for Thru

  print pink for Fri

  Sat and Sun are not acceptable values.

  30 mins

Exercise

Write a java program to display the silver jubilee, golden jubilee and diamond jubilee celebration dates of a movie, whose release date will be entered by the user. Assume that the movie will run successfully.(Silver Jubilee 25, Golden Jubilee 50, Diamond Jubilee 60, Platinum Jubilee 75).Note that, if these dates fall on a Sunday or any public holiday, then the date must be moved to next day.

(30 mins)

`java.lang.Math`

- **`Math`** is a final class contains, methods to perform mathematical operations. . All the methods are **`static`**.

Go through Math class methods for 5 minutes

Exercise

- Write a java program to do the following.

a) *Generate a random number between 1 and 100 and display the floor value and ceil value of the same.*

b) *Display the cube roots of the list of prime numbers till 100.*

(30 mins)

Test your understanding

- Can you guess what the below code will display?

```
double a = 3.5;
System.out.println( Math.round(a));
System.out.println( Math.ceil(a));
System.out.println( Math.floor(a))
```

- That should have been easy. What will it display, if the number is negative?

```
double a = -3.5;
```

java.util.StringTokenizer

- This class, allows a string, to be split into tokens based on delimiter. The default delimiter is space.

```
StringTokenizer st = new StringTokenizer("europe
asia america");
while (st.hasMoreTokens()) {
System.out.println(st.nextToken());}
```

Prints

```
    europe
    asia
    america
```

Go through StringTokenizer class methods for 5 minutes

Performance issue

- We saw two ways to split a string:
  - Using **`StringTokenizer`**
  - Using **`split()`** method of **`String`**

- Between the two, **`split()`** method of **`String`** is believed to give better performance.
- Java documentation states that

  "**`StringTokenizer`** is a legacy class, that is retained for compatibility reasons; although, its use is not encouraged in the new code. It is recommended to use the **`split`** method of **`String`** or the **`java.util.regex`** package instead."

```
String x = "sd,fg,ll";
String[] y = x.split(",");
System.out.println(y[0]+" "+y[1]+" "+y[2]);
```

Activity

The program gets the input in the form of numbers, separated by a comma. For example "23,44,345.8“.

Write a code to sum up the numbers and display its result.

Summary

- Date, Calendar and GregorianCalendar are 3 important classes to work with date and time.
- Calendar is an abstract class. GregorianCalendar is a concrete subclass of Calendar.
- A Date object is represented internally, as a single long number.
- java.text.SimpleDateFormat class is used for formatting and parsing dates in a locale-sensitive manner.
- In Gregorian Calender, the value is stored as time, in milliseconds.
- Math is a final class contains, methods to perform mathematical operations. . All the methods are static.
- StringTokenizer is a class that allows a string, to be split into tokens based on delimiter. The default delimiter is space.