**How to Parse Date in Desired format – Java Date**

[**JAVA DATE**](http://beginnersbook.com/category/technology/java-guide/java-date/)

This post is to discuss few important points about parse() method. If you are looking for String to Date and Date to String conversion then refer the following posts:

1. [**Convert String to Date in Java**](http://beginnersbook.com/2013/04/java-string-to-date-conversion/)
2. [**Convert Date to String in Java**](http://beginnersbook.com/2013/05/java-date-string-conversion/)

Converting strings to desired date format is a time consuming and tedious process in many languages including Java. However Java provides several APIs to accomplish this.

* java.text.DateFormat
* java.text.SimpleDateFormat
* java.util.Date
* java.util.Calendar
* java.util.TimeZone

The first two APIs are concerned with representing dates as strings. They both have methods:

String format (java.util.Date)  
java.util.Date parse (String)

The difference between**java.text.DateFormat** and**java.text.SimpleDateFormat** is that the[**java.text.SimpleDateFormat**](http://beginnersbook.com/2013/05/simple-date-format-java/) uses a custom format defined by special formatting characters where as the java.text.DateFormat uses the standard date formatting of the current locale

For Example

import java.text.\*;

SimpleDateFormat sdfmt1 = new SimpleDateFormat("dd/MM/yy");

SimpleDateFormat sdfmt2= new SimpleDateFormat("dd-MMM-yyyy");

java.util.Date dDate = sdfmt1.parse( strInput );

String strOutput = sdfmt2.format( dDate );

This will turn e.g. “21/4/99″ into “21-Apr-1999″.

**Points to note:**

1. The **parse()** method does not (by default) throw an Exception if the date is correctly formatted but invalid on the calendar (e.g. 29/2/2001), instead it alters the date to be a valid one (in the previous example, to 1/3/2001). If this isn’t what you want, call date.setLenient( false ).

2. The **format** represented as a 4-digit year (yyyy) then “12/6/01″ is parsed as “12/06/0001″.

If it is 2-digit (yy) then the year is set to be between 80 years before and 20 years after the date the SimpleDateFormat instance is created (e.g. today!). Thus if today is 1/10/01, then “30/9/21″ is interpreted as “30/09/2021″ and “1/10/22″ is “01/10/1922″. This boundary date can be altered by the set2DigitYearStart() method.

3. Both java.util.\* and java.sql.\* packages contain objects called Date. Hence user need to explicitly mention the Date as java.util.Date (in case if the user is working with SQL also)

More sophisticated date arithmetic and processing can be done using[**TimeZone**](http://beginnersbook.com/2013/05/java-date-timezone/), [**Calendar**](http://beginnersbook.com/2013/05/java-calendar-class/) and its child Gregorian Calendar, but these do not deal directly with dates formatted as strings.