

|  |  |
| --- | --- |
| **Author(s)** | Arjun Singh |
| **Authorized by** | Head, ETA |
| **Creation/Revision Date** | May, 2016 |
| **Version** | 1.0 |

**Usage Guidelines**

Do not forward this document to any non-Infosys mail ID. Forwarding this document to a non-Infosys mail ID may lead to disciplinary action against you, including termination of employment.

Contents of this material cannot be used in any other internal or external document without explicit permission from ETA@infosys.com.

# **COPYRIGHT NOTICE**

*© 2016 Infosys Limited, Bangalore, India. All Rights Reserved.*

*Infosys believes the information in this document is accurate as of its publication date; such information is subject to change without notice. Infosys acknowledges the proprietary rights of other companies to the trademarks, product names and such other intellectual property rights mentioned in this document. Except as expressly permitted, neither this documentation nor any part of it may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, printing, photocopying, recording or otherwise, without the prior permission of Infosys Limited and/ or any named intellectual property rights holders under this document.*

Education, Training and Assessment Department

Infosys Limited

Electronics City

Hosur Road

Bangalore - 561 229, India.

Tel: 91 80 852 0261-270

Fax: 91 80 852 0362

[www.infosys.com](http://www.infosys.com)

<mailto:ETA@infosys.com>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **Author(s)** | **Reviewer(s)** | **Description** |
| 1.0 | May, 2016 | Arjun\_Singh07 | Nishant\_Raj01 | Baseline version |

Document Revision History

Table of Contents

[**COPYRIGHT NOTICE** 5](#_Toc451853497)

[Calendar class in JAVA 8](#_Toc451853498)

[What is Calendar class? 8](#_Toc451853499)

[Mutability issues in Calendar class 8](#_Toc451853500)

[Creating an instance of Calendar class 8](#_Toc451853501)

[get(): obtain Calendar data. 8](#_Toc451853502)

[add(): to supplement the chosen fields with given values. 9](#_Toc451853503)

[after(): comparing Calendar objects. 9](#_Toc451853504)

[before(): comparing Calendar objects. 9](#_Toc451853505)

[compareTo(): returning *int* value after comparing 2 Calendar objects. 10](#_Toc451853506)

[getTime(): to obtain time information for Calendar instance. 10](#_Toc451853507)

[set(): setting the values. 10](#_Toc451853508)

[setTime(): setting the Date object in Calendar instance. 11](#_Toc451853509)

[clear(): sets field to undefined values. 11](#_Toc451853510)

[Condition check 1 : To check for a particular day of week 12](#_Toc451853511)

[Condition check 2: To check if given day is 3rd of month or not 12](#_Toc451853512)

# Calendar class in JAVA

## What is Calendar class?

* The Calendar class is an abstract class that provides methods for converting between a specific instant in time and a set of calendar fields such as YEAR, MONTH, DAY\_OF\_MONTH, HOUR, and so on, and for manipulating the calendar fields, such as getting the date of the next week.
* An instant in time can be represented by a millisecond value that is an offset from the Epoch, January 1, 1970 00:00:00.000 GMT (Gregorian).

## Mutability issues in Calendar class

Till Java 1.7, Java SE has two separate date and time APIs - java.util.Date and java.util.Calendar. Both APIs are consistently described as difficult to use by Java developers on weblogs and forums. Notably, both use a zero-index for months, which is a cause of many bugs. The current Java SE API also suffers in multi-threaded environments. Immutable classes are known to be inherently thread-safe as their state cannot change. However, both Date and Calendar are mutable, which requires programmers to consider cloning and threading explicitly.

## Creating an instance of Calendar class

Like other locale-sensitive classes, Calendar provides a class method, getInstance, for getting a generally useful object of this type. Calendar's getInstance method returns a Calendar object whose calendar fields have been initialized with the current date and time:

* 1. Calendar rightNow = Calendar.getInstance();
* The calendar field values can be set by calling the set methods.

## get(): obtain Calendar data.

get() method returns the required field of a Calendar instance:

Calendar cal = Calendar.*getInstance*();

cal.set(2015, Calendar.***MAY***, 18, 10, 12);

**int** year = cal.get(Calendar.***YEAR***);

**int** month = cal.get(Calendar.***MONTH***);

**int** day = cal.get(Calendar.***DAY\_OF\_MONTH***);

**int** hour = cal.get(Calendar.***HOUR\_OF\_DAY***);

**int** minute = cal.get(Calendar.***MINUTE***);

**int** second = cal.get(Calendar.***SECOND***);

System.***out***.println("Year is:"+year+", Month is:"+month+", Day is:"+day+", Hour is:"+hour+", Minute is:"+minute+", Second is:"+second);

OUTPUT:

Year is:2015, Month is:4, Day is:18, Hour is:10, Minute is:12, Second is:48

## add(): to supplement the chosen fields with given values.

add() method adds the required value to the specified field:

Calendar cal = Calendar.*getInstance*();

cal.set(2015, Calendar.***MAY***, 18, 10, 12);

cal.add(Calendar.***YEAR***, 5);

System.***out***.println(cal.get(Calendar.***YEAR***));

OUTPUT:

2020

## after(): comparing Calendar objects.

after() method checks if the calling instance is later than the argument instance.

Calendar cal1 = Calendar.*getInstance*();

Calendar cal2 = Calendar.*getInstance*();

cal1.set(2015, Calendar.***MAY***, 18, 10, 12);

cal2.set(2015, Calendar.***JUNE***, 18, 10, 12);

**boolean** result = cal2.after(cal1);

System.***out***.println(result);

OUTPUT:

true

## before(): comparing Calendar objects.

before() method checks if the calling instance is earlier than the argument instance.

Calendar cal1 = Calendar.*getInstance*();

Calendar cal2 = Calendar.*getInstance*();

cal1.set(2015, Calendar.***MAY***, 18, 10, 12);

cal2.set(2015, Calendar.***JUNE***, 18, 10, 12);

**boolean** result = cal2.before(cal1);

System.***out***.println(result);

OUTPUT:

false

## compareTo(): returning *int* value after comparing 2 Calendar objects.

CompareTo() method gives the value 0 if the time represented by the argument is equal to the time represented by this Calendar; a value less than 0 if the time of this Calendar is before the time represented by the argument; and a value greater than 0 if the time of this Calendar is after the time represented by the argument.

Calendar cal1 = Calendar.*getInstance*();

Calendar cal2 = Calendar.*getInstance*();

cal1.set(2015, Calendar.***MAY***, 18, 10, 12);

cal2.set(2015, Calendar.***JUNE***, 18, 10, 12);

System.***out***.println(cal1.compareTo(cal2));

OUTPUT:

-1

## getTime(): to obtain time information for Calendar instance.

getTime() returns a Date object representing the time value.

Calendar cal1 = Calendar.*getInstance*();

cal1.set(2015, Calendar.***MAY***, 18, 10, 12);

System.***out***.println(cal1.getTime());

OUTPUT:

Mon May 18 10:12:35 IST 2015

## set(): setting the values.

set() method sets the values for the calendar fields.

Calendar cal1 = Calendar.*getInstance*();

cal1.set(2020, Calendar.***DECEMBER***, 18, 10, 12);

System.***out***.println(cal1.getTime());

OUTPUT:

Fri Dec 18 10:12:12 IST 2020

## setTime(): setting the Date object in Calendar instance.

setTime() method sets this Calendar's time with the given Date.

Calendar cal1 = Calendar.*getInstance*();

cal1.set(2020, Calendar.***DECEMBER***, 18, 10, 12);

Date date = **new** Date(787842556);

cal1.setTime(date);

System.***out***.println("After setting Time: "+cal1.getTime());

OUTPUT:

After setting Time: Sat Jan 10 08:20:42 IST 1970

## clear(): sets field to undefined values.

The clear() method sets the given calendar field value and the time value of this Calendar undefined.

Calendar cal1 = Calendar.*getInstance*();

cal1.set(2020, Calendar.***DECEMBER***, 18, 10, 12);

System.***out***.println("Current calendar: "+ cal1.getTime());

cal1.clear(Calendar.***YEAR***);

System.***out***.println("The calendar shows: "+cal1.getTime());

cal1.clear(Calendar.***MONTH***);

System.***out***.println("The calendar shows: " +cal1.getTime());

OUTPUT:

Current calendar: Fri Dec 18 10:12:58 IST 2020

The calendar shows: Fri Dec 18 10:12:58 IST 1970

The calendar shows: Sun Jan 18 10:12:58 IST 1970

# Condition check 1 : To check for a particular day of week

To check if the given day is a Saturday or not:

Calendar cal1 = Calendar.*getInstance*();

cal1.set(2020, Calendar.***DECEMBER***, 18, 10, 12);

**int** day = cal1.get(Calendar.***DAY\_OF\_WEEK***);

**if**(day==7){

System.***out***.println(**true**);

}

**else**

System.***out***.println(**false**);

OUTPUT:

false

# Condition check 2: To check if given day is 3rd of month or not

To check if the day is 3rd Monday of the month or not:

Calendar cal1 = Calendar.*getInstance*();

cal1.set(2015, Calendar.***MAY***, 18, 10, 12);

**int** dayOfTheWeek = cal1.get(Calendar.***DAY\_OF\_WEEK***);

**int** dayOfTheWeekInMonth = cal1.get(Calendar.***DAY\_OF\_WEEK\_IN\_MONTH***);

**if**(dayOfTheWeek == 2 && dayOfTheWeekInMonth == 3){

System.***out***.println(**true**);

}

**else**{

System.***out***.println(**false**);

}

OUTPUT:

true