| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/GregorianCalendar.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/util/FormatterClosedException.html)   [**NEXT CLASS**](http://docs.google.com/java/util/HashMap.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/GregorianCalendar.html)    [**NO FRAMES**](http://docs.google.com/GregorianCalendar.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | [CONSTR](#tyjcwt) | [METHOD](#3dy6vkm) | DETAIL: [FIELD](#2s8eyo1) | [CONSTR](#26in1rg) | [METHOD](#1y810tw) |

## **java.util**

Class GregorianCalendar

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 [java.util.Calendar](http://docs.google.com/java/util/Calendar.html)  
 **java.util.GregorianCalendar**

**All Implemented Interfaces:** [Serializable](http://docs.google.com/java/io/Serializable.html), [Cloneable](http://docs.google.com/java/lang/Cloneable.html), [Comparable](http://docs.google.com/java/lang/Comparable.html)<[Calendar](http://docs.google.com/java/util/Calendar.html)>

public class **GregorianCalendar**extends [Calendar](http://docs.google.com/java/util/Calendar.html)

GregorianCalendar is a concrete subclass of Calendar and provides the standard calendar system used by most of the world.

GregorianCalendar is a hybrid calendar that supports both the Julian and Gregorian calendar systems with the support of a single discontinuity, which corresponds by default to the Gregorian date when the Gregorian calendar was instituted (October 15, 1582 in some countries, later in others). The cutover date may be changed by the caller by calling [setGregorianChange()](http://docs.google.com/java/util/GregorianCalendar.html#setGregorianChange(java.util.Date)).

Historically, in those countries which adopted the Gregorian calendar first, October 4, 1582 (Julian) was thus followed by October 15, 1582 (Gregorian). This calendar models this correctly. Before the Gregorian cutover, GregorianCalendar implements the Julian calendar. The only difference between the Gregorian and the Julian calendar is the leap year rule. The Julian calendar specifies leap years every four years, whereas the Gregorian calendar omits century years which are not divisible by 400.

GregorianCalendar implements *proleptic* Gregorian and Julian calendars. That is, dates are computed by extrapolating the current rules indefinitely far backward and forward in time. As a result, GregorianCalendar may be used for all years to generate meaningful and consistent results. However, dates obtained using GregorianCalendar are historically accurate only from March 1, 4 AD onward, when modern Julian calendar rules were adopted. Before this date, leap year rules were applied irregularly, and before 45 BC the Julian calendar did not even exist.

Prior to the institution of the Gregorian calendar, New Year's Day was March 25. To avoid confusion, this calendar always uses January 1. A manual adjustment may be made if desired for dates that are prior to the Gregorian changeover and which fall between January 1 and March 24.

Values calculated for the WEEK\_OF\_YEAR field range from 1 to 53. Week 1 for a year is the earliest seven day period starting on getFirstDayOfWeek() that contains at least getMinimalDaysInFirstWeek() days from that year. It thus depends on the values of getMinimalDaysInFirstWeek(), getFirstDayOfWeek(), and the day of the week of January 1. Weeks between week 1 of one year and week 1 of the following year are numbered sequentially from 2 to 52 or 53 (as needed).

For example, January 1, 1998 was a Thursday. If getFirstDayOfWeek() is MONDAY and getMinimalDaysInFirstWeek() is 4 (these are the values reflecting ISO 8601 and many national standards), then week 1 of 1998 starts on December 29, 1997, and ends on January 4, 1998. If, however, getFirstDayOfWeek() is SUNDAY, then week 1 of 1998 starts on January 4, 1998, and ends on January 10, 1998; the first three days of 1998 then are part of week 53 of 1997.

Values calculated for the WEEK\_OF\_MONTH field range from 0 to 6. Week 1 of a month (the days with WEEK\_OF\_MONTH = 1) is the earliest set of at least getMinimalDaysInFirstWeek() contiguous days in that month, ending on the day before getFirstDayOfWeek(). Unlike week 1 of a year, week 1 of a month may be shorter than 7 days, need not start on getFirstDayOfWeek(), and will not include days of the previous month. Days of a month before week 1 have a WEEK\_OF\_MONTH of 0.

For example, if getFirstDayOfWeek() is SUNDAY and getMinimalDaysInFirstWeek() is 4, then the first week of January 1998 is Sunday, January 4 through Saturday, January 10. These days have a WEEK\_OF\_MONTH of 1. Thursday, January 1 through Saturday, January 3 have a WEEK\_OF\_MONTH of 0. If getMinimalDaysInFirstWeek() is changed to 3, then January 1 through January 3 have a WEEK\_OF\_MONTH of 1.

The clear methods set calendar field(s) undefined. GregorianCalendar uses the following default value for each calendar field if its value is undefined.

| Field | Default Value |
| --- | --- |
| ERA | AD |
| YEAR | 1970 |
| MONTH | JANUARY |
| DAY\_OF\_MONTH | 1 |
| DAY\_OF\_WEEK | the first day of week |
| WEEK\_OF\_MONTH | 0 |
| DAY\_OF\_WEEK\_IN\_MONTH | 1 |
| AM\_PM | AM |
| HOUR, HOUR\_OF\_DAY, MINUTE, SECOND, MILLISECOND | 0 |

Default values are not applicable for the fields not listed above.

**Example:**

// get the supported ids for GMT-08:00 (Pacific Standard Time)  
 String[] ids = TimeZone.getAvailableIDs(-8 \* 60 \* 60 \* 1000);  
 // if no ids were returned, something is wrong. get out.  
 if (ids.length == 0)  
 System.exit(0);  
  
 // begin output  
 System.out.println("Current Time");  
  
 // create a Pacific Standard Time time zone  
 SimpleTimeZone pdt = new SimpleTimeZone(-8 \* 60 \* 60 \* 1000, ids[0]);  
  
 // set up rules for daylight savings time  
 pdt.setStartRule(Calendar.APRIL, 1, Calendar.SUNDAY, 2 \* 60 \* 60 \* 1000);  
 pdt.setEndRule(Calendar.OCTOBER, -1, Calendar.SUNDAY, 2 \* 60 \* 60 \* 1000);  
  
 // create a GregorianCalendar with the Pacific Daylight time zone  
 // and the current date and time  
 Calendar calendar = new GregorianCalendar(pdt);  
 Date trialTime = new Date();  
 calendar.setTime(trialTime);  
  
 // print out a bunch of interesting things  
 System.out.println("ERA: " + calendar.get(Calendar.ERA));  
 System.out.println("YEAR: " + calendar.get(Calendar.YEAR));  
 System.out.println("MONTH: " + calendar.get(Calendar.MONTH));  
 System.out.println("WEEK\_OF\_YEAR: " + calendar.get(Calendar.WEEK\_OF\_YEAR));  
 System.out.println("WEEK\_OF\_MONTH: " + calendar.get(Calendar.WEEK\_OF\_MONTH));  
 System.out.println("DATE: " + calendar.get(Calendar.DATE));  
 System.out.println("DAY\_OF\_MONTH: " + calendar.get(Calendar.DAY\_OF\_MONTH));  
 System.out.println("DAY\_OF\_YEAR: " + calendar.get(Calendar.DAY\_OF\_YEAR));  
 System.out.println("DAY\_OF\_WEEK: " + calendar.get(Calendar.DAY\_OF\_WEEK));  
 System.out.println("DAY\_OF\_WEEK\_IN\_MONTH: "  
 + calendar.get(Calendar.DAY\_OF\_WEEK\_IN\_MONTH));  
 System.out.println("AM\_PM: " + calendar.get(Calendar.AM\_PM));  
 System.out.println("HOUR: " + calendar.get(Calendar.HOUR));  
 System.out.println("HOUR\_OF\_DAY: " + calendar.get(Calendar.HOUR\_OF\_DAY));  
 System.out.println("MINUTE: " + calendar.get(Calendar.MINUTE));  
 System.out.println("SECOND: " + calendar.get(Calendar.SECOND));  
 System.out.println("MILLISECOND: " + calendar.get(Calendar.MILLISECOND));  
 System.out.println("ZONE\_OFFSET: "  
 + (calendar.get(Calendar.ZONE\_OFFSET)/(60\*60\*1000)));  
 System.out.println("DST\_OFFSET: "  
 + (calendar.get(Calendar.DST\_OFFSET)/(60\*60\*1000)));  
  
 System.out.println("Current Time, with hour reset to 3");  
 calendar.clear(Calendar.HOUR\_OF\_DAY); // so doesn't override  
 calendar.set(Calendar.HOUR, 3);  
 System.out.println("ERA: " + calendar.get(Calendar.ERA));  
 System.out.println("YEAR: " + calendar.get(Calendar.YEAR));  
 System.out.println("MONTH: " + calendar.get(Calendar.MONTH));  
 System.out.println("WEEK\_OF\_YEAR: " + calendar.get(Calendar.WEEK\_OF\_YEAR));  
 System.out.println("WEEK\_OF\_MONTH: " + calendar.get(Calendar.WEEK\_OF\_MONTH));  
 System.out.println("DATE: " + calendar.get(Calendar.DATE));  
 System.out.println("DAY\_OF\_MONTH: " + calendar.get(Calendar.DAY\_OF\_MONTH));  
 System.out.println("DAY\_OF\_YEAR: " + calendar.get(Calendar.DAY\_OF\_YEAR));  
 System.out.println("DAY\_OF\_WEEK: " + calendar.get(Calendar.DAY\_OF\_WEEK));  
 System.out.println("DAY\_OF\_WEEK\_IN\_MONTH: "  
 + calendar.get(Calendar.DAY\_OF\_WEEK\_IN\_MONTH));  
 System.out.println("AM\_PM: " + calendar.get(Calendar.AM\_PM));  
 System.out.println("HOUR: " + calendar.get(Calendar.HOUR));  
 System.out.println("HOUR\_OF\_DAY: " + calendar.get(Calendar.HOUR\_OF\_DAY));  
 System.out.println("MINUTE: " + calendar.get(Calendar.MINUTE));  
 System.out.println("SECOND: " + calendar.get(Calendar.SECOND));  
 System.out.println("MILLISECOND: " + calendar.get(Calendar.MILLISECOND));  
 System.out.println("ZONE\_OFFSET: "  
 + (calendar.get(Calendar.ZONE\_OFFSET)/(60\*60\*1000))); // in hours  
 System.out.println("DST\_OFFSET: "  
 + (calendar.get(Calendar.DST\_OFFSET)/(60\*60\*1000))); // in hours

**Since:** JDK1.1 **See Also:**[TimeZone](http://docs.google.com/java/util/TimeZone.html), [Serialized Form](http://docs.google.com/serialized-form.html#java.util.GregorianCalendar)

| **Field Summary** | |
| --- | --- |
| static int | [**AD**](http://docs.google.com/java/util/GregorianCalendar.html#AD)            Value of the ERA field indicating the common era (Anno Domini), also known as CE. |
| static int | [**BC**](http://docs.google.com/java/util/GregorianCalendar.html#BC)            Value of the ERA field indicating the period before the common era (before Christ), also known as BCE. |

| **Fields inherited from class java.util.**[**Calendar**](http://docs.google.com/java/util/Calendar.html) |
| --- |
| [ALL\_STYLES](http://docs.google.com/java/util/Calendar.html#ALL_STYLES), [AM](http://docs.google.com/java/util/Calendar.html#AM), [AM\_PM](http://docs.google.com/java/util/Calendar.html#AM_PM), [APRIL](http://docs.google.com/java/util/Calendar.html#APRIL), [areFieldsSet](http://docs.google.com/java/util/Calendar.html#areFieldsSet), [AUGUST](http://docs.google.com/java/util/Calendar.html#AUGUST), [DATE](http://docs.google.com/java/util/Calendar.html#DATE), [DAY\_OF\_MONTH](http://docs.google.com/java/util/Calendar.html#DAY_OF_MONTH), [DAY\_OF\_WEEK](http://docs.google.com/java/util/Calendar.html#DAY_OF_WEEK), [DAY\_OF\_WEEK\_IN\_MONTH](http://docs.google.com/java/util/Calendar.html#DAY_OF_WEEK_IN_MONTH), [DAY\_OF\_YEAR](http://docs.google.com/java/util/Calendar.html#DAY_OF_YEAR), [DECEMBER](http://docs.google.com/java/util/Calendar.html#DECEMBER), [DST\_OFFSET](http://docs.google.com/java/util/Calendar.html#DST_OFFSET), [ERA](http://docs.google.com/java/util/Calendar.html#ERA), [FEBRUARY](http://docs.google.com/java/util/Calendar.html#FEBRUARY), [FIELD\_COUNT](http://docs.google.com/java/util/Calendar.html#FIELD_COUNT), [fields](http://docs.google.com/java/util/Calendar.html#fields), [FRIDAY](http://docs.google.com/java/util/Calendar.html#FRIDAY), [HOUR](http://docs.google.com/java/util/Calendar.html#HOUR), [HOUR\_OF\_DAY](http://docs.google.com/java/util/Calendar.html#HOUR_OF_DAY), [isSet](http://docs.google.com/java/util/Calendar.html#isSet), [isTimeSet](http://docs.google.com/java/util/Calendar.html#isTimeSet), [JANUARY](http://docs.google.com/java/util/Calendar.html#JANUARY), [JULY](http://docs.google.com/java/util/Calendar.html#JULY), [JUNE](http://docs.google.com/java/util/Calendar.html#JUNE), [LONG](http://docs.google.com/java/util/Calendar.html#LONG), [MARCH](http://docs.google.com/java/util/Calendar.html#MARCH), [MAY](http://docs.google.com/java/util/Calendar.html#MAY), [MILLISECOND](http://docs.google.com/java/util/Calendar.html#MILLISECOND), [MINUTE](http://docs.google.com/java/util/Calendar.html#MINUTE), [MONDAY](http://docs.google.com/java/util/Calendar.html#MONDAY), [MONTH](http://docs.google.com/java/util/Calendar.html#MONTH), [NOVEMBER](http://docs.google.com/java/util/Calendar.html#NOVEMBER), [OCTOBER](http://docs.google.com/java/util/Calendar.html#OCTOBER), [PM](http://docs.google.com/java/util/Calendar.html#PM), [SATURDAY](http://docs.google.com/java/util/Calendar.html#SATURDAY), [SECOND](http://docs.google.com/java/util/Calendar.html#SECOND), [SEPTEMBER](http://docs.google.com/java/util/Calendar.html#SEPTEMBER), [SHORT](http://docs.google.com/java/util/Calendar.html#SHORT), [SUNDAY](http://docs.google.com/java/util/Calendar.html#SUNDAY), [THURSDAY](http://docs.google.com/java/util/Calendar.html#THURSDAY), [time](http://docs.google.com/java/util/Calendar.html#time), [TUESDAY](http://docs.google.com/java/util/Calendar.html#TUESDAY), [UNDECIMBER](http://docs.google.com/java/util/Calendar.html#UNDECIMBER), [WEDNESDAY](http://docs.google.com/java/util/Calendar.html#WEDNESDAY), [WEEK\_OF\_MONTH](http://docs.google.com/java/util/Calendar.html#WEEK_OF_MONTH), [WEEK\_OF\_YEAR](http://docs.google.com/java/util/Calendar.html#WEEK_OF_YEAR), [YEAR](http://docs.google.com/java/util/Calendar.html#YEAR), [ZONE\_OFFSET](http://docs.google.com/java/util/Calendar.html#ZONE_OFFSET) |

| **Constructor Summary** | |
| --- | --- |
| [**GregorianCalendar**](http://docs.google.com/java/util/GregorianCalendar.html#GregorianCalendar())()            Constructs a default GregorianCalendar using the current time in the default time zone with the default locale. |
| [**GregorianCalendar**](http://docs.google.com/java/util/GregorianCalendar.html#GregorianCalendar(int,%20int,%20int))(int year, int month, int dayOfMonth)            Constructs a GregorianCalendar with the given date set in the default time zone with the default locale. |
| [**GregorianCalendar**](http://docs.google.com/java/util/GregorianCalendar.html#GregorianCalendar(int,%20int,%20int,%20int,%20int))(int year, int month, int dayOfMonth, int hourOfDay, int minute)            Constructs a GregorianCalendar with the given date and time set for the default time zone with the default locale. |
| [**GregorianCalendar**](http://docs.google.com/java/util/GregorianCalendar.html#GregorianCalendar(int,%20int,%20int,%20int,%20int,%20int))(int year, int month, int dayOfMonth, int hourOfDay, int minute, int second)            Constructs a GregorianCalendar with the given date and time set for the default time zone with the default locale. |
| [**GregorianCalendar**](http://docs.google.com/java/util/GregorianCalendar.html#GregorianCalendar(java.util.Locale))([Locale](http://docs.google.com/java/util/Locale.html) aLocale)            Constructs a GregorianCalendar based on the current time in the default time zone with the given locale. |
| [**GregorianCalendar**](http://docs.google.com/java/util/GregorianCalendar.html#GregorianCalendar(java.util.TimeZone))([TimeZone](http://docs.google.com/java/util/TimeZone.html) zone)            Constructs a GregorianCalendar based on the current time in the given time zone with the default locale. |
| [**GregorianCalendar**](http://docs.google.com/java/util/GregorianCalendar.html#GregorianCalendar(java.util.TimeZone,%20java.util.Locale))([TimeZone](http://docs.google.com/java/util/TimeZone.html) zone, [Locale](http://docs.google.com/java/util/Locale.html) aLocale)            Constructs a GregorianCalendar based on the current time in the given time zone with the given locale. |

| **Method Summary** | |
| --- | --- |
| void | [**add**](http://docs.google.com/java/util/GregorianCalendar.html#add(int,%20int))(int field, int amount)            Adds the specified (signed) amount of time to the given calendar field, based on the calendar's rules. |
| [Object](http://docs.google.com/java/lang/Object.html) | [**clone**](http://docs.google.com/java/util/GregorianCalendar.html#clone())()            Creates and returns a copy of this object. |
| protected  void | [**computeFields**](http://docs.google.com/java/util/GregorianCalendar.html#computeFields())()            Converts the time value (millisecond offset from the [Epoch](http://docs.google.com/Calendar.html#Epoch)) to calendar field values. |
| protected  void | [**computeTime**](http://docs.google.com/java/util/GregorianCalendar.html#computeTime())()            Converts calendar field values to the time value (millisecond offset from the [Epoch](http://docs.google.com/Calendar.html#Epoch)). |
| boolean | [**equals**](http://docs.google.com/java/util/GregorianCalendar.html#equals(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) obj)            Compares this GregorianCalendar to the specified Object. |
| int | [**getActualMaximum**](http://docs.google.com/java/util/GregorianCalendar.html#getActualMaximum(int))(int field)            Returns the maximum value that this calendar field could have, taking into consideration the given time value and the current values of the [getFirstDayOfWeek](http://docs.google.com/java/util/Calendar.html#getFirstDayOfWeek()), [getMinimalDaysInFirstWeek](http://docs.google.com/java/util/Calendar.html#getMinimalDaysInFirstWeek()), [getGregorianChange](http://docs.google.com/java/util/GregorianCalendar.html#getGregorianChange()) and [getTimeZone](http://docs.google.com/java/util/Calendar.html#getTimeZone()) methods. |
| int | [**getActualMinimum**](http://docs.google.com/java/util/GregorianCalendar.html#getActualMinimum(int))(int field)            Returns the minimum value that this calendar field could have, taking into consideration the given time value and the current values of the [getFirstDayOfWeek](http://docs.google.com/java/util/Calendar.html#getFirstDayOfWeek()), [getMinimalDaysInFirstWeek](http://docs.google.com/java/util/Calendar.html#getMinimalDaysInFirstWeek()), [getGregorianChange](http://docs.google.com/java/util/GregorianCalendar.html#getGregorianChange()) and [getTimeZone](http://docs.google.com/java/util/Calendar.html#getTimeZone()) methods. |
| int | [**getGreatestMinimum**](http://docs.google.com/java/util/GregorianCalendar.html#getGreatestMinimum(int))(int field)            Returns the highest minimum value for the given calendar field of this GregorianCalendar instance. |
| [Date](http://docs.google.com/java/util/Date.html) | [**getGregorianChange**](http://docs.google.com/java/util/GregorianCalendar.html#getGregorianChange())()            Gets the Gregorian Calendar change date. |
| int | [**getLeastMaximum**](http://docs.google.com/java/util/GregorianCalendar.html#getLeastMaximum(int))(int field)            Returns the lowest maximum value for the given calendar field of this GregorianCalendar instance. |
| int | [**getMaximum**](http://docs.google.com/java/util/GregorianCalendar.html#getMaximum(int))(int field)            Returns the maximum value for the given calendar field of this GregorianCalendar instance. |
| int | [**getMinimum**](http://docs.google.com/java/util/GregorianCalendar.html#getMinimum(int))(int field)            Returns the minimum value for the given calendar field of this GregorianCalendar instance. |
| [TimeZone](http://docs.google.com/java/util/TimeZone.html) | [**getTimeZone**](http://docs.google.com/java/util/GregorianCalendar.html#getTimeZone())()            Gets the time zone. |
| int | [**hashCode**](http://docs.google.com/java/util/GregorianCalendar.html#hashCode())()            Generates the hash code for this GregorianCalendar object. |
| boolean | [**isLeapYear**](http://docs.google.com/java/util/GregorianCalendar.html#isLeapYear(int))(int year)            Determines if the given year is a leap year. |
| void | [**roll**](http://docs.google.com/java/util/GregorianCalendar.html#roll(int,%20boolean))(int field, boolean up)            Adds or subtracts (up/down) a single unit of time on the given time field without changing larger fields. |
| void | [**roll**](http://docs.google.com/java/util/GregorianCalendar.html#roll(int,%20int))(int field, int amount)            Adds a signed amount to the specified calendar field without changing larger fields. |
| void | [**setGregorianChange**](http://docs.google.com/java/util/GregorianCalendar.html#setGregorianChange(java.util.Date))([Date](http://docs.google.com/java/util/Date.html) date)            Sets the GregorianCalendar change date. |
| void | [**setTimeZone**](http://docs.google.com/java/util/GregorianCalendar.html#setTimeZone(java.util.TimeZone))([TimeZone](http://docs.google.com/java/util/TimeZone.html) zone)            Sets the time zone with the given time zone value. |

| **Methods inherited from class java.util.**[**Calendar**](http://docs.google.com/java/util/Calendar.html) |
| --- |
| [after](http://docs.google.com/java/util/Calendar.html#after(java.lang.Object)), [before](http://docs.google.com/java/util/Calendar.html#before(java.lang.Object)), [clear](http://docs.google.com/java/util/Calendar.html#clear()), [clear](http://docs.google.com/java/util/Calendar.html#clear(int)), [compareTo](http://docs.google.com/java/util/Calendar.html#compareTo(java.util.Calendar)), [complete](http://docs.google.com/java/util/Calendar.html#complete()), [get](http://docs.google.com/java/util/Calendar.html#get(int)), [getAvailableLocales](http://docs.google.com/java/util/Calendar.html#getAvailableLocales()), [getDisplayName](http://docs.google.com/java/util/Calendar.html#getDisplayName(int,%20int,%20java.util.Locale)), [getDisplayNames](http://docs.google.com/java/util/Calendar.html#getDisplayNames(int,%20int,%20java.util.Locale)), [getFirstDayOfWeek](http://docs.google.com/java/util/Calendar.html#getFirstDayOfWeek()), [getInstance](http://docs.google.com/java/util/Calendar.html#getInstance()), [getInstance](http://docs.google.com/java/util/Calendar.html#getInstance(java.util.Locale)), [getInstance](http://docs.google.com/java/util/Calendar.html#getInstance(java.util.TimeZone)), [getInstance](http://docs.google.com/java/util/Calendar.html#getInstance(java.util.TimeZone,%20java.util.Locale)), [getMinimalDaysInFirstWeek](http://docs.google.com/java/util/Calendar.html#getMinimalDaysInFirstWeek()), [getTime](http://docs.google.com/java/util/Calendar.html#getTime()), [getTimeInMillis](http://docs.google.com/java/util/Calendar.html#getTimeInMillis()), [internalGet](http://docs.google.com/java/util/Calendar.html#internalGet(int)), [isLenient](http://docs.google.com/java/util/Calendar.html#isLenient()), [isSet](http://docs.google.com/java/util/Calendar.html#isSet(int)), [set](http://docs.google.com/java/util/Calendar.html#set(int,%20int)), [set](http://docs.google.com/java/util/Calendar.html#set(int,%20int,%20int)), [set](http://docs.google.com/java/util/Calendar.html#set(int,%20int,%20int,%20int,%20int)), [set](http://docs.google.com/java/util/Calendar.html#set(int,%20int,%20int,%20int,%20int,%20int)), [setFirstDayOfWeek](http://docs.google.com/java/util/Calendar.html#setFirstDayOfWeek(int)), [setLenient](http://docs.google.com/java/util/Calendar.html#setLenient(boolean)), [setMinimalDaysInFirstWeek](http://docs.google.com/java/util/Calendar.html#setMinimalDaysInFirstWeek(int)), [setTime](http://docs.google.com/java/util/Calendar.html#setTime(java.util.Date)), [setTimeInMillis](http://docs.google.com/java/util/Calendar.html#setTimeInMillis(long)), [toString](http://docs.google.com/java/util/Calendar.html#toString()) |

| **Methods inherited from class java.lang.**[**Object**](http://docs.google.com/java/lang/Object.html) |
| --- |
| [finalize](http://docs.google.com/java/lang/Object.html#finalize()), [getClass](http://docs.google.com/java/lang/Object.html#getClass()), [notify](http://docs.google.com/java/lang/Object.html#notify()), [notifyAll](http://docs.google.com/java/lang/Object.html#notifyAll()), [wait](http://docs.google.com/java/lang/Object.html#wait()), [wait](http://docs.google.com/java/lang/Object.html#wait(long)), [wait](http://docs.google.com/java/lang/Object.html#wait(long,%20int)) |

| **Field Detail** |
| --- |

### BC

public static final int **BC**

Value of the ERA field indicating the period before the common era (before Christ), also known as BCE. The sequence of years at the transition from BC to AD is ..., 2 BC, 1 BC, 1 AD, 2 AD,...

**See Also:**[Calendar.ERA](http://docs.google.com/java/util/Calendar.html#ERA), [Constant Field Values](http://docs.google.com/constant-values.html#java.util.GregorianCalendar.BC)

### AD

public static final int **AD**

Value of the ERA field indicating the common era (Anno Domini), also known as CE. The sequence of years at the transition from BC to AD is ..., 2 BC, 1 BC, 1 AD, 2 AD,...

**See Also:**[Calendar.ERA](http://docs.google.com/java/util/Calendar.html#ERA), [Constant Field Values](http://docs.google.com/constant-values.html#java.util.GregorianCalendar.AD)

| **Constructor Detail** |
| --- |

### GregorianCalendar

public **GregorianCalendar**()

Constructs a default GregorianCalendar using the current time in the default time zone with the default locale.

### GregorianCalendar

public **GregorianCalendar**([TimeZone](http://docs.google.com/java/util/TimeZone.html) zone)

Constructs a GregorianCalendar based on the current time in the given time zone with the default locale.

**Parameters:**zone - the given time zone.

### GregorianCalendar

public **GregorianCalendar**([Locale](http://docs.google.com/java/util/Locale.html) aLocale)

Constructs a GregorianCalendar based on the current time in the default time zone with the given locale.

**Parameters:**aLocale - the given locale.

### GregorianCalendar

public **GregorianCalendar**([TimeZone](http://docs.google.com/java/util/TimeZone.html) zone,  
 [Locale](http://docs.google.com/java/util/Locale.html) aLocale)

Constructs a GregorianCalendar based on the current time in the given time zone with the given locale.

**Parameters:**zone - the given time zone.aLocale - the given locale.

### GregorianCalendar

public **GregorianCalendar**(int year,  
 int month,  
 int dayOfMonth)

Constructs a GregorianCalendar with the given date set in the default time zone with the default locale.

**Parameters:**year - the value used to set the YEAR calendar field in the calendar.month - the value used to set the MONTH calendar field in the calendar. Month value is 0-based. e.g., 0 for January.dayOfMonth - the value used to set the DAY\_OF\_MONTH calendar field in the calendar.

### GregorianCalendar

public **GregorianCalendar**(int year,  
 int month,  
 int dayOfMonth,  
 int hourOfDay,  
 int minute)

Constructs a GregorianCalendar with the given date and time set for the default time zone with the default locale.

**Parameters:**year - the value used to set the YEAR calendar field in the calendar.month - the value used to set the MONTH calendar field in the calendar. Month value is 0-based. e.g., 0 for January.dayOfMonth - the value used to set the DAY\_OF\_MONTH calendar field in the calendar.hourOfDay - the value used to set the HOUR\_OF\_DAY calendar field in the calendar.minute - the value used to set the MINUTE calendar field in the calendar.

### GregorianCalendar

public **GregorianCalendar**(int year,  
 int month,  
 int dayOfMonth,  
 int hourOfDay,  
 int minute,  
 int second)

Constructs a GregorianCalendar with the given date and time set for the default time zone with the default locale.

**Parameters:**year - the value used to set the YEAR calendar field in the calendar.month - the value used to set the MONTH calendar field in the calendar. Month value is 0-based. e.g., 0 for January.dayOfMonth - the value used to set the DAY\_OF\_MONTH calendar field in the calendar.hourOfDay - the value used to set the HOUR\_OF\_DAY calendar field in the calendar.minute - the value used to set the MINUTE calendar field in the calendar.second - the value used to set the SECOND calendar field in the calendar.

| **Method Detail** |
| --- |

### setGregorianChange

public void **setGregorianChange**([Date](http://docs.google.com/java/util/Date.html) date)

Sets the GregorianCalendar change date. This is the point when the switch from Julian dates to Gregorian dates occurred. Default is October 15, 1582 (Gregorian). Previous to this, dates will be in the Julian calendar.

To obtain a pure Julian calendar, set the change date to Date(Long.MAX\_VALUE). To obtain a pure Gregorian calendar, set the change date to Date(Long.MIN\_VALUE).

**Parameters:**date - the given Gregorian cutover date.

### getGregorianChange

public final [Date](http://docs.google.com/java/util/Date.html) **getGregorianChange**()

Gets the Gregorian Calendar change date. This is the point when the switch from Julian dates to Gregorian dates occurred. Default is October 15, 1582 (Gregorian). Previous to this, dates will be in the Julian calendar.

**Returns:**the Gregorian cutover date for this GregorianCalendar object.

### isLeapYear

public boolean **isLeapYear**(int year)

Determines if the given year is a leap year. Returns true if the given year is a leap year. To specify BC year numbers, 1 - year number must be given. For example, year BC 4 is specified as -3.

**Parameters:**year - the given year. **Returns:**true if the given year is a leap year; false otherwise.

### equals

public boolean **equals**([Object](http://docs.google.com/java/lang/Object.html) obj)

Compares this GregorianCalendar to the specified Object. The result is true if and only if the argument is a GregorianCalendar object that represents the same time value (millisecond offset from the [Epoch](http://docs.google.com/Calendar.html#Epoch)) under the same Calendar parameters and Gregorian change date as this object.

**Overrides:**[equals](http://docs.google.com/java/util/Calendar.html#equals(java.lang.Object)) in class [Calendar](http://docs.google.com/java/util/Calendar.html) **Parameters:**obj - the object to compare with. **Returns:**true if this object is equal to obj; false otherwise.**See Also:**[Calendar.compareTo(Calendar)](http://docs.google.com/java/util/Calendar.html#compareTo(java.util.Calendar))

### hashCode

public int **hashCode**()

Generates the hash code for this GregorianCalendar object.

**Overrides:**[hashCode](http://docs.google.com/java/util/Calendar.html#hashCode()) in class [Calendar](http://docs.google.com/java/util/Calendar.html) **Returns:**a hash code value for this object.**See Also:**[Object.equals(java.lang.Object)](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)), [Hashtable](http://docs.google.com/java/util/Hashtable.html)

### add

public void **add**(int field,  
 int amount)

Adds the specified (signed) amount of time to the given calendar field, based on the calendar's rules.

*Add rule 1*. The value of field after the call minus the value of field before the call is amount, modulo any overflow that has occurred in field. Overflow occurs when a field value exceeds its range and, as a result, the next larger field is incremented or decremented and the field value is adjusted back into its range.

*Add rule 2*. If a smaller field is expected to be invariant, but it is impossible for it to be equal to its prior value because of changes in its minimum or maximum after field is changed, then its value is adjusted to be as close as possible to its expected value. A smaller field represents a smaller unit of time. HOUR is a smaller field than DAY\_OF\_MONTH. No adjustment is made to smaller fields that are not expected to be invariant. The calendar system determines what fields are expected to be invariant.

**Specified by:**[add](http://docs.google.com/java/util/Calendar.html#add(int,%20int)) in class [Calendar](http://docs.google.com/java/util/Calendar.html) **Parameters:**field - the calendar field.amount - the amount of date or time to be added to the field. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if field is ZONE\_OFFSET, DST\_OFFSET, or unknown, or if any calendar fields have out-of-range values in non-lenient mode.**See Also:**[Calendar.roll(int,int)](http://docs.google.com/java/util/Calendar.html#roll(int,%20int)), [Calendar.set(int,int)](http://docs.google.com/java/util/Calendar.html#set(int,%20int))

### roll

public void **roll**(int field,  
 boolean up)

Adds or subtracts (up/down) a single unit of time on the given time field without changing larger fields.

*Example*: Consider a GregorianCalendar originally set to December 31, 1999. Calling [roll(Calendar.MONTH, true)](http://docs.google.com/java/util/GregorianCalendar.html#roll(int,%20boolean)) sets the calendar to January 31, 1999. The YEAR field is unchanged because it is a larger field than MONTH.

**Specified by:**[roll](http://docs.google.com/java/util/Calendar.html#roll(int,%20boolean)) in class [Calendar](http://docs.google.com/java/util/Calendar.html) **Parameters:**up - indicates if the value of the specified calendar field is to be rolled up or rolled down. Use true if rolling up, false otherwise.field - the time field. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if field is ZONE\_OFFSET, DST\_OFFSET, or unknown, or if any calendar fields have out-of-range values in non-lenient mode.**See Also:**[add(int,int)](http://docs.google.com/java/util/GregorianCalendar.html#add(int,%20int)), [Calendar.set(int,int)](http://docs.google.com/java/util/Calendar.html#set(int,%20int))

### roll

public void **roll**(int field,  
 int amount)

Adds a signed amount to the specified calendar field without changing larger fields. A negative roll amount means to subtract from field without changing larger fields. If the specified amount is 0, this method performs nothing.

This method calls [Calendar.complete()](http://docs.google.com/java/util/Calendar.html#complete()) before adding the amount so that all the calendar fields are normalized. If there is any calendar field having an out-of-range value in non-lenient mode, then an IllegalArgumentException is thrown.

*Example*: Consider a GregorianCalendar originally set to August 31, 1999. Calling roll(Calendar.MONTH, 8) sets the calendar to April 30, **1999**. Using a GregorianCalendar, the DAY\_OF\_MONTH field cannot be 31 in the month April. DAY\_OF\_MONTH is set to the closest possible value, 30. The YEAR field maintains the value of 1999 because it is a larger field than MONTH.

*Example*: Consider a GregorianCalendar originally set to Sunday June 6, 1999. Calling roll(Calendar.WEEK\_OF\_MONTH, -1) sets the calendar to Tuesday June 1, 1999, whereas calling add(Calendar.WEEK\_OF\_MONTH, -1) sets the calendar to Sunday May 30, 1999. This is because the roll rule imposes an additional constraint: The MONTH must not change when the WEEK\_OF\_MONTH is rolled. Taken together with add rule 1, the resultant date must be between Tuesday June 1 and Saturday June 5. According to add rule 2, the DAY\_OF\_WEEK, an invariant when changing the WEEK\_OF\_MONTH, is set to Tuesday, the closest possible value to Sunday (where Sunday is the first day of the week).

**Overrides:**[roll](http://docs.google.com/java/util/Calendar.html#roll(int,%20int)) in class [Calendar](http://docs.google.com/java/util/Calendar.html) **Parameters:**field - the calendar field.amount - the signed amount to add to field. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if field is ZONE\_OFFSET, DST\_OFFSET, or unknown, or if any calendar fields have out-of-range values in non-lenient mode.**Since:** 1.2 **See Also:**[roll(int,boolean)](http://docs.google.com/java/util/GregorianCalendar.html#roll(int,%20boolean)), [add(int,int)](http://docs.google.com/java/util/GregorianCalendar.html#add(int,%20int)), [Calendar.set(int,int)](http://docs.google.com/java/util/Calendar.html#set(int,%20int))

### getMinimum

public int **getMinimum**(int field)

Returns the minimum value for the given calendar field of this GregorianCalendar instance. The minimum value is defined as the smallest value returned by the [get](http://docs.google.com/java/util/Calendar.html#get(int)) method for any possible time value, taking into consideration the current values of the [getFirstDayOfWeek](http://docs.google.com/java/util/Calendar.html#getFirstDayOfWeek()), [getMinimalDaysInFirstWeek](http://docs.google.com/java/util/Calendar.html#getMinimalDaysInFirstWeek()), [getGregorianChange](http://docs.google.com/java/util/GregorianCalendar.html#getGregorianChange()) and [getTimeZone](http://docs.google.com/java/util/Calendar.html#getTimeZone()) methods.

**Specified by:**[getMinimum](http://docs.google.com/java/util/Calendar.html#getMinimum(int)) in class [Calendar](http://docs.google.com/java/util/Calendar.html) **Parameters:**field - the calendar field. **Returns:**the minimum value for the given calendar field.**See Also:**[getMaximum(int)](http://docs.google.com/java/util/GregorianCalendar.html#getMaximum(int)), [getGreatestMinimum(int)](http://docs.google.com/java/util/GregorianCalendar.html#getGreatestMinimum(int)), [getLeastMaximum(int)](http://docs.google.com/java/util/GregorianCalendar.html#getLeastMaximum(int)), [getActualMinimum(int)](http://docs.google.com/java/util/GregorianCalendar.html#getActualMinimum(int)), [getActualMaximum(int)](http://docs.google.com/java/util/GregorianCalendar.html#getActualMaximum(int))

### getMaximum

public int **getMaximum**(int field)

Returns the maximum value for the given calendar field of this GregorianCalendar instance. The maximum value is defined as the largest value returned by the [get](http://docs.google.com/java/util/Calendar.html#get(int)) method for any possible time value, taking into consideration the current values of the [getFirstDayOfWeek](http://docs.google.com/java/util/Calendar.html#getFirstDayOfWeek()), [getMinimalDaysInFirstWeek](http://docs.google.com/java/util/Calendar.html#getMinimalDaysInFirstWeek()), [getGregorianChange](http://docs.google.com/java/util/GregorianCalendar.html#getGregorianChange()) and [getTimeZone](http://docs.google.com/java/util/Calendar.html#getTimeZone()) methods.

**Specified by:**[getMaximum](http://docs.google.com/java/util/Calendar.html#getMaximum(int)) in class [Calendar](http://docs.google.com/java/util/Calendar.html) **Parameters:**field - the calendar field. **Returns:**the maximum value for the given calendar field.**See Also:**[getMinimum(int)](http://docs.google.com/java/util/GregorianCalendar.html#getMinimum(int)), [getGreatestMinimum(int)](http://docs.google.com/java/util/GregorianCalendar.html#getGreatestMinimum(int)), [getLeastMaximum(int)](http://docs.google.com/java/util/GregorianCalendar.html#getLeastMaximum(int)), [getActualMinimum(int)](http://docs.google.com/java/util/GregorianCalendar.html#getActualMinimum(int)), [getActualMaximum(int)](http://docs.google.com/java/util/GregorianCalendar.html#getActualMaximum(int))

### getGreatestMinimum

public int **getGreatestMinimum**(int field)

Returns the highest minimum value for the given calendar field of this GregorianCalendar instance. The highest minimum value is defined as the largest value returned by [getActualMinimum(int)](http://docs.google.com/java/util/GregorianCalendar.html#getActualMinimum(int)) for any possible time value, taking into consideration the current values of the [getFirstDayOfWeek](http://docs.google.com/java/util/Calendar.html#getFirstDayOfWeek()), [getMinimalDaysInFirstWeek](http://docs.google.com/java/util/Calendar.html#getMinimalDaysInFirstWeek()), [getGregorianChange](http://docs.google.com/java/util/GregorianCalendar.html#getGregorianChange()) and [getTimeZone](http://docs.google.com/java/util/Calendar.html#getTimeZone()) methods.

**Specified by:**[getGreatestMinimum](http://docs.google.com/java/util/Calendar.html#getGreatestMinimum(int)) in class [Calendar](http://docs.google.com/java/util/Calendar.html) **Parameters:**field - the calendar field. **Returns:**the highest minimum value for the given calendar field.**See Also:**[getMinimum(int)](http://docs.google.com/java/util/GregorianCalendar.html#getMinimum(int)), [getMaximum(int)](http://docs.google.com/java/util/GregorianCalendar.html#getMaximum(int)), [getLeastMaximum(int)](http://docs.google.com/java/util/GregorianCalendar.html#getLeastMaximum(int)), [getActualMinimum(int)](http://docs.google.com/java/util/GregorianCalendar.html#getActualMinimum(int)), [getActualMaximum(int)](http://docs.google.com/java/util/GregorianCalendar.html#getActualMaximum(int))

### getLeastMaximum

public int **getLeastMaximum**(int field)

Returns the lowest maximum value for the given calendar field of this GregorianCalendar instance. The lowest maximum value is defined as the smallest value returned by [getActualMaximum(int)](http://docs.google.com/java/util/GregorianCalendar.html#getActualMaximum(int)) for any possible time value, taking into consideration the current values of the [getFirstDayOfWeek](http://docs.google.com/java/util/Calendar.html#getFirstDayOfWeek()), [getMinimalDaysInFirstWeek](http://docs.google.com/java/util/Calendar.html#getMinimalDaysInFirstWeek()), [getGregorianChange](http://docs.google.com/java/util/GregorianCalendar.html#getGregorianChange()) and [getTimeZone](http://docs.google.com/java/util/Calendar.html#getTimeZone()) methods.

**Specified by:**[getLeastMaximum](http://docs.google.com/java/util/Calendar.html#getLeastMaximum(int)) in class [Calendar](http://docs.google.com/java/util/Calendar.html) **Parameters:**field - the calendar field **Returns:**the lowest maximum value for the given calendar field.**See Also:**[getMinimum(int)](http://docs.google.com/java/util/GregorianCalendar.html#getMinimum(int)), [getMaximum(int)](http://docs.google.com/java/util/GregorianCalendar.html#getMaximum(int)), [getGreatestMinimum(int)](http://docs.google.com/java/util/GregorianCalendar.html#getGreatestMinimum(int)), [getActualMinimum(int)](http://docs.google.com/java/util/GregorianCalendar.html#getActualMinimum(int)), [getActualMaximum(int)](http://docs.google.com/java/util/GregorianCalendar.html#getActualMaximum(int))

### getActualMinimum

public int **getActualMinimum**(int field)

Returns the minimum value that this calendar field could have, taking into consideration the given time value and the current values of the [getFirstDayOfWeek](http://docs.google.com/java/util/Calendar.html#getFirstDayOfWeek()), [getMinimalDaysInFirstWeek](http://docs.google.com/java/util/Calendar.html#getMinimalDaysInFirstWeek()), [getGregorianChange](http://docs.google.com/java/util/GregorianCalendar.html#getGregorianChange()) and [getTimeZone](http://docs.google.com/java/util/Calendar.html#getTimeZone()) methods.

For example, if the Gregorian change date is January 10, 1970 and the date of this GregorianCalendar is January 20, 1970, the actual minimum value of the DAY\_OF\_MONTH field is 10 because the previous date of January 10, 1970 is December 27, 1996 (in the Julian calendar). Therefore, December 28, 1969 to January 9, 1970 don't exist.

**Overrides:**[getActualMinimum](http://docs.google.com/java/util/Calendar.html#getActualMinimum(int)) in class [Calendar](http://docs.google.com/java/util/Calendar.html) **Parameters:**field - the calendar field **Returns:**the minimum of the given field for the time value of this GregorianCalendar**Since:** 1.2 **See Also:**[getMinimum(int)](http://docs.google.com/java/util/GregorianCalendar.html#getMinimum(int)), [getMaximum(int)](http://docs.google.com/java/util/GregorianCalendar.html#getMaximum(int)), [getGreatestMinimum(int)](http://docs.google.com/java/util/GregorianCalendar.html#getGreatestMinimum(int)), [getLeastMaximum(int)](http://docs.google.com/java/util/GregorianCalendar.html#getLeastMaximum(int)), [getActualMaximum(int)](http://docs.google.com/java/util/GregorianCalendar.html#getActualMaximum(int))

### getActualMaximum

public int **getActualMaximum**(int field)

Returns the maximum value that this calendar field could have, taking into consideration the given time value and the current values of the [getFirstDayOfWeek](http://docs.google.com/java/util/Calendar.html#getFirstDayOfWeek()), [getMinimalDaysInFirstWeek](http://docs.google.com/java/util/Calendar.html#getMinimalDaysInFirstWeek()), [getGregorianChange](http://docs.google.com/java/util/GregorianCalendar.html#getGregorianChange()) and [getTimeZone](http://docs.google.com/java/util/Calendar.html#getTimeZone()) methods. For example, if the date of this instance is February 1, 2004, the actual maximum value of the DAY\_OF\_MONTH field is 29 because 2004 is a leap year, and if the date of this instance is February 1, 2005, it's 28.

**Overrides:**[getActualMaximum](http://docs.google.com/java/util/Calendar.html#getActualMaximum(int)) in class [Calendar](http://docs.google.com/java/util/Calendar.html) **Parameters:**field - the calendar field **Returns:**the maximum of the given field for the time value of this GregorianCalendar**Since:** 1.2 **See Also:**[getMinimum(int)](http://docs.google.com/java/util/GregorianCalendar.html#getMinimum(int)), [getMaximum(int)](http://docs.google.com/java/util/GregorianCalendar.html#getMaximum(int)), [getGreatestMinimum(int)](http://docs.google.com/java/util/GregorianCalendar.html#getGreatestMinimum(int)), [getLeastMaximum(int)](http://docs.google.com/java/util/GregorianCalendar.html#getLeastMaximum(int)), [getActualMinimum(int)](http://docs.google.com/java/util/GregorianCalendar.html#getActualMinimum(int))

### clone

public [Object](http://docs.google.com/java/lang/Object.html) **clone**()

**Description copied from class:** [**Calendar**](http://docs.google.com/java/util/Calendar.html#clone()) Creates and returns a copy of this object.

**Overrides:**[clone](http://docs.google.com/java/util/Calendar.html#clone()) in class [Calendar](http://docs.google.com/java/util/Calendar.html) **Returns:**a copy of this object.**See Also:**[Cloneable](http://docs.google.com/java/lang/Cloneable.html)

### getTimeZone

public [TimeZone](http://docs.google.com/java/util/TimeZone.html) **getTimeZone**()

**Description copied from class:** [**Calendar**](http://docs.google.com/java/util/Calendar.html#getTimeZone()) Gets the time zone.

**Overrides:**[getTimeZone](http://docs.google.com/java/util/Calendar.html#getTimeZone()) in class [Calendar](http://docs.google.com/java/util/Calendar.html) **Returns:**the time zone object associated with this calendar.

### setTimeZone

public void **setTimeZone**([TimeZone](http://docs.google.com/java/util/TimeZone.html) zone)

**Description copied from class:** [**Calendar**](http://docs.google.com/java/util/Calendar.html#setTimeZone(java.util.TimeZone)) Sets the time zone with the given time zone value.

**Overrides:**[setTimeZone](http://docs.google.com/java/util/Calendar.html#setTimeZone(java.util.TimeZone)) in class [Calendar](http://docs.google.com/java/util/Calendar.html) **Parameters:**zone - the given time zone.

### computeFields

protected void **computeFields**()

Converts the time value (millisecond offset from the [Epoch](http://docs.google.com/Calendar.html#Epoch)) to calendar field values. The time is *not* recomputed first; to recompute the time, then the fields, call the complete method.

**Specified by:**[computeFields](http://docs.google.com/java/util/Calendar.html#computeFields()) in class [Calendar](http://docs.google.com/java/util/Calendar.html) **See Also:**[Calendar.complete()](http://docs.google.com/java/util/Calendar.html#complete())

### computeTime

protected void **computeTime**()

Converts calendar field values to the time value (millisecond offset from the [Epoch](http://docs.google.com/Calendar.html#Epoch)).

**Specified by:**[computeTime](http://docs.google.com/java/util/Calendar.html#computeTime()) in class [Calendar](http://docs.google.com/java/util/Calendar.html) **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if any calendar fields are invalid.**See Also:**[Calendar.complete()](http://docs.google.com/java/util/Calendar.html#complete()), [Calendar.computeFields()](http://docs.google.com/java/util/Calendar.html#computeFields())

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/GregorianCalendar.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/util/FormatterClosedException.html)   [**NEXT CLASS**](http://docs.google.com/java/util/HashMap.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/GregorianCalendar.html)    [**NO FRAMES**](http://docs.google.com/GregorianCalendar.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | [CONSTR](#tyjcwt) | [METHOD](#3dy6vkm) | DETAIL: [FIELD](#2s8eyo1) | [CONSTR](#26in1rg) | [METHOD](#1y810tw) |

[Submit a bug or feature](http://bugs.sun.com/services/bugreport/index.jsp)

For further API reference and developer documentation, see [Java SE Developer Documentation](http://docs.google.com/webnotes/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Copyright 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to [license terms](http://docs.google.com/legal/license.html). Also see the [documentation redistribution policy](http://java.sun.com/docs/redist.html).