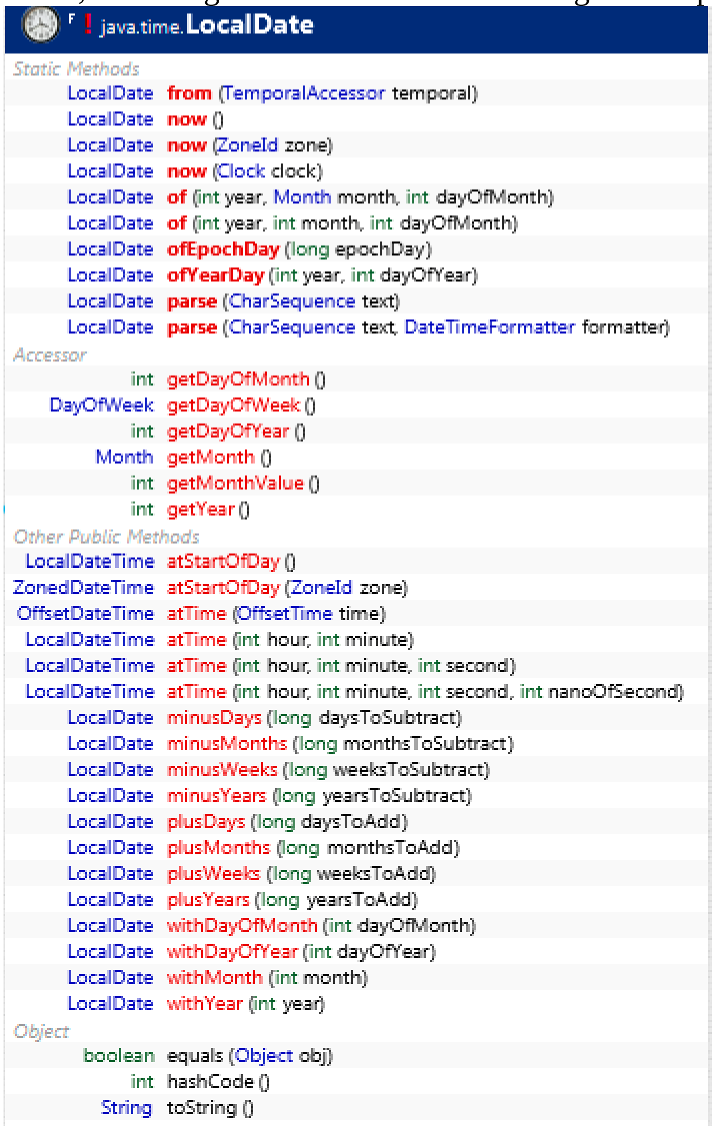
# Java 8 - LocalDate Class API Guide

A `LocalDate` represents a year-month-day in the `ISO` calendar and is useful for representing a date without a time. You might use a `LocalDate` to track a significant event, such as a birth date or wedding date.
This class does not store and represent a time or time-zone. Instead, it is a description of the date which can be used for a birthday or holiday.
`LocalDate` Class provides lots of APIs/Methods to deal with local dates so in this post we will discuss a few important and frequently used `LocalDate` APIs/Methods with Examples.
In order to understand better, let's categories methods into their usage and explained with examples.



```
F ! java.time.LocalDate
Static Methods
        LocalDate  from (TemporalAccessor temporal)
        LocalDate  now ()
        LocalDate  now (ZoneId zone)
        LocalDate  now (Clock clock)
        LocalDate  of (int year, Month month, int dayOfMonth)
        LocalDate  of (int year, int month, int dayOfMonth)
        LocalDate  ofEpochDay (long epochDay)
        LocalDate  ofYearDay (int year, int dayOfYear)
        LocalDate  parse (CharSequence text)
        LocalDate  parse (CharSequence text, DateTimeFormatter formatter)
Accessor
              int  getDayOfMonth ()
        DayOfWeek  getDayOfWeek ()
              int  getDayOfYear ()
            Month  getMonth ()
              int  getMonthValue ()
              int  getYear ()
Other Public Methods
    LocalDateTime  atStartOfDay ()
    ZonedDateTime  atStartOfDay (ZoneId zone)
   OffsetDateTime  atTime (OffsetTime time)
    LocalDateTime  atTime (int hour, int minute)
    LocalDateTime  atTime (int hour, int minute, int second)
    LocalDateTime  atTime (int hour, int minute, int second, int nanoOfSecond)
        LocalDate  minusDays (long daysToSubtract)
        LocalDate  minusMonths (long monthsToSubtract)
        LocalDate  minusWeeks (long weeksToSubtract)
        LocalDate  minusYears (long yearsToSubtract)
        LocalDate  plusDays (long daysToAdd)
        LocalDate  plusMonths (long monthsToAdd)
        LocalDate  plusWeeks (long weeksToAdd)
        LocalDate  plusYears (long yearsToAdd)
        LocalDate  withDayOfMonth (int dayOfMonth)
        LocalDate  withDayOfYear (int dayOfYear)
        LocalDate  withMonth (int month)
        LocalDate  withYear (int year)
Object
          boolean  equals (Object obj)
              int  hashCode ()
           String  toString ()
```

## LocalDate APIs/Methods

### 1. LocalDate APIs to Create the Current Date and Specific Date

- `static LocalDate now()` - Obtains the current date from the system clock in the default time-zone.
- `static LocalDate now(Clock clock)` - Obtains the current date from the specified clock.

•`static LocalDate now(ZoneId zone)` - Obtains the current date from the system clock in the specified time-zone.
•`static LocalDate of(int year, int month, int dayOfMonth)` - Obtains an instance of LocalDate from a year, month and day.
•`static LocalDate of(int year, Month month, int dayOfMonth)` - Obtains an instance of LocalDate from a year, month and day.

## 2. LocalDate APIs to get Year, Month, Day from LocalDate

•`Month getMonth()` - Gets the month-of-year field using the Month enum.
•`int getMonthValue()` - Gets the month-of-year field from 1 to 12.
•`int getYear()` - Gets the year field.
•`int getDayOfMonth()` - Gets the day-of-month field.
•`DayOfWeek getDayOfWeek()` - Gets the day-of-week field, which is an enum DayOfWeek.
•_int getDayOfYear() _- Gets the day-of-year field.

## 3. LocalDate APIs to Add or Subtract Days, Months, Weeks and Years to LocalDate

•`LocalDate plusDays(long daysToAdd)` - Returns a copy of this LocalDate with the specified number of days added.
•`LocalDate plusMonths(long monthsToAdd)` - Returns a copy of this LocalDate with the specified number of months added.
•`LocalDate plusWeeks(long weeksToAdd)` - Returns a copy of this LocalDate with the specified number of weeks added.
•`LocalDate plusYears(long yearsToAdd)` - Returns a copy of this LocalDate with the specified number of years added.

## 4. LocalDate APIs to compare LocalDate objects

•`boolean isAfter(ChronoLocalDate other)` - Checks if this date is after the specified date.
•`boolean isBefore(ChronoLocalDate other)` - Checks if this date is before the specified date.
•`boolean isEqual(ChronoLocalDate other)` - Checks if this date is equal to the specified date.

## 5. LocalDate APIs to Get Number of Days from Month or Year from LocalDate

•`int lengthOfMonth()` - Returns the length of the month represented by this date.
•`int lengthOfYear()` - Returns the length of the year represented by this date.

## 6. LocalDate API to Check If a Given Year Is Leap Year or Not

•`boolean isLeapYear()` - Checks if the year is a leap year, according to the ISO proleptic calendar system rules.

### 7. LocalDate APIs to Convert String to LocalDate in java

- *static LocalDate parse(CharSequence text)* - Obtains an instance of LocalDate from a text string such as 2007-12-03.
- *static LocalDate parse(CharSequence text, DateTimeFormatter formatter)* - Obtains an instance of LocalDate from a text string using a specific formatter.

### 8. LocalDate API to Convert LocalDate to String in java

- *String format(DateTimeFormatter formatter)* - Formats this date using the specified formatter.

Let's discuss all the above APIs with examples.

## 1. LocalDate APIs to Create the Current Date and Specific Date

*LocalDate* class provides below APIs to create current and specific date using LocalDate class.

- *static LocalDate now()* - Obtains the current date from the system clock in the default time-zone.
- *static LocalDate now(Clock clock)* - Obtains the current date from the specified clock.
- *static LocalDate now(ZoneId zone)* - Obtains the current date from the system clock in the specified time-zone.
- *static LocalDate of(int year, int month, int dayOfMonth)* - Obtains an instance of LocalDate from a year, month and day.
- *static LocalDate of(int year, Month month, int dayOfMonth)* - Obtains an instance of LocalDate from a year, month and day.

```java
import java.time.Clock;
import java.time.LocalDate;
import java.time.Month;
import java.time.ZoneId;
/**
 * Program to demonstrate LocalDate Class APIs.
 * @author javaguides.net
 *
 */
public class LocalDateExamples {

    public static void main(String[] args) {
        createLocalDate();
    }
    private static void createLocalDate() {
        // Current date
        LocalDate localDate = LocalDate.now();
        System.out.println(localDate);

        LocalDate localDate1 = LocalDate.now(Clock.systemDefaultZone());
        System.out.println(localDate1);

        LocalDate localDate3 =
LocalDate.now(Clock.system(ZoneId.of("Indian/Cocos")));
        System.out.println(localDate3);

        // Specific date
        LocalDate localDate2 = LocalDate.of(1991, Month.MARCH, 24);
        System.out.println(localDate2);

        LocalDate localDate5 = LocalDate.of(1991, 12, 24);
        System.out.println(localDate5);
    }
}
```

Output:

```
2018-08-10
2018-08-10
2018-08-10
1991-03-24
1991-12-24
```

## 2. LocalDate APIs to get Year, Month, Day from LocalDate

*LocalDate* class provides below APIs to get a year, month and day respectively.

- *Month getMonth()* - Gets the month-of-year field using the Month enum.
- *int getMonthValue()* - Gets the month-of-year field from 1 to 12.
- *int getYear()* - Gets the year field.
- *int getDayOfMonth()* - Gets the day-of-month field.
- *DayOfWeek getDayOfWeek()* - Gets the day-of-week field, which is an enum DayOfWeek.
- *int getDayOfYear()* - Gets the day-of-year field.

```
import java.time.LocalDate;
/**
 * Program to demonstrate LocalDate Class APIs.
 * @author javaguides.net
 *
 */
public class LocalDateExamples {

    public static void main(String[] args) {
        getYearMonthDay();
    }

    private static void getYearMonthDay() {
        LocalDate localDate = LocalDate.now();
        System.out.println("Year : " + localDate.getYear());
        System.out.println("Month : " + localDate.getMonth().getValue());
        System.out.println("Day of Month : " + localDate.getDayOfMonth());
        System.out.println("Day of Week : " + localDate.getDayOfWeek());
        System.out.println("Day of Year : " + localDate.getDayOfYear());
    }
}
```

Output:

```
Year : 2018
Month : 8
Day of Month : 10
Day of Week : FRIDAY
Day of Year : 222
```

## 3. LocalDate APIs to Add or Subtract Days, Months, Weeks and Years to LocalDate

*LocalDate* class provides below APIs to add or subtract Days, Months, Weeks and Years to LocalDate.

- *LocalDate plusDays(long daysToAdd)* - Returns a copy of this LocalDate with the specified number of days added.
- *LocalDate plusMonths(long monthsToAdd)* - Returns a copy of this LocalDate with the specified number of months added.
- *LocalDate plusWeeks(long weeksToAdd)* - Returns a copy of this LocalDate with the specified number of weeks added.
- *LocalDate plusYears(long yearsToAdd)* - Returns a copy of this LocalDate with the specified number of years added.

```
import java.time.LocalDate;
/**
 * Program to demonstrate LocalDate Class APIs.
 * @author javaguides.net
 *
 */
public class LocalDateExamples {

    public static void main(String[] args) {
        addOrSubstractUsingLocalDate();
    }

    public static void addOrSubstractUsingLocalDate() {

        LocalDate localDate = LocalDate.now();

        // LocalDate's plus methods
        System.out.println("Addition of days : " + localDate.plusDays(5));
        System.out.println("Addition of months : " + localDate.plusMonths(15));
        System.out.println("Addition of weeks : " + localDate.plusWeeks(45));
        System.out.println("Addition of years : " + localDate.plusYears(5));

        // LocalDate's minus methods
        System.out.println("Subtraction of days : " + localDate.minusDays(5));
        System.out.println("Subtraction of months : " + localDate.minusMonths(15));
        System.out.println("Subtraction of weeks : " + localDate.minusWeeks(45));
        System.out.println("Subtraction of years : " + localDate.minusYears(5));
    }
}
```

Output:

```
Addition of days : 2018-08-15
Addition of months : 2019-11-10
Addition of weeks : 2019-06-21
Addition of years : 2023-08-10
Subtraction of days : 2018-08-05
Subtraction of months : 2017-05-10
Subtraction of weeks : 2017-09-29
Subtraction of years : 2013-08-10
```

## 4. LocalDate APIs to compare LocalDate objects

`LocalDate` class provides below APIs to compare `LocalDate` objects in Java.

- *`boolean isAfter(ChronoLocalDate other)`* - Checks if this date is after the specified date.

•*boolean isBefore(ChronoLocalDate other)* - Checks if this date is before the specified date.
•*boolean isEqual(ChronoLocalDate other)* - Checks if this date is equal to the specified date.

```java
import java.time.LocalDate;
import java.time.Month;
/**
 * Program to demonstrate LocalDate Class APIs.
 * @author javaguides.net
 *
 */
public class LocalDateExamples {

    public static void main(String[] args) {
        compareLocalDate();
    }

    private static void compareLocalDate() {
        LocalDate localDate1 = LocalDate.now();
        LocalDate localDate2 = LocalDate.of(2017, Month.MAY, 14);
        LocalDate localDate3 = LocalDate.of(2016, Month.MAY, 15);

        // isEqual() example
        if (localDate1.isEqual(localDate2)) {
            System.out.println("localDate1 and localDate2 are equal");
        } else {
            System.out.println("localDate1 and localDate2 are not equal");
        }

        // ifAfter() example
        if (localDate2.isAfter(localDate3)) {
            System.out.println("localDate2 comes after localDate3");
        }

        // isBefore() example
        if (localDate3.isBefore(localDate1)) {
           System.out.println("localDate3 comes before localDate1");
        }
    }
}
```

Output:

```
localDate1 and localDate2 are not equal
localDate2 comes after localDate3
localDate3 comes before localDate1
```

## 5. LocalDate APIs to Get Number of Days from Month or Year from LocalDate

`LocalDate` class provides below APIs to get a number of days from Month or Year from LocalDate.

- *int lengthOfMonth()* - Returns the length of the month represented by this date.
- *int lengthOfYear()* - Returns the length of the year represented by this date.

```java
import java.time.LocalDate;
import java.time.Month;
/**
 * Program to demonstrate LocalDate Class APIs.
 * @author javaguides.net
 *
 */
public class LocalDateExamples {
    public static void main(String[] args) {
          getDaysFromMonthAndYear();
    }
    private static void getDaysFromMonthAndYear() {
         LocalDate localDate1 = LocalDate.of(2017, Month.JANUARY, 1);
         LocalDate localDate2 = LocalDate.of(2016, Month.FEBRUARY, 1);

         // Number of days in a month
         System.out.println("Number of days in " + localDate1.getMonth() + " : " +
localDate1.lengthOfMonth());
         System.out.println("Number of days in " + localDate2.getMonth() + " : " +
localDate2.lengthOfMonth());

         // Number of days in a year
         System.out.println("Number of days in " + localDate1.getYear() + " : " +
localDate1.lengthOfYear());
         System.out.println("Number of days in " + localDate2.getYear() + " : " +
localDate2.lengthOfYear());
    }
}
```

Output:

```
Number of days in JANUARY : 31
Number of days in FEBRUARY : 29
Number of days in 2017 : 365
Number of days in 2016 : 366
```

## 6. LocalDate API to Check If a Given Year Is Leap Year or Not

*LocalDate* class provides below API to check if a given year is a leap year or not.

- *boolean isLeapYear()* - Checks if the year is a leap year, according to the ISO proleptic calendar system rules.

```java
package com.ramesh.java8.datetime.api;

import java.time.LocalDate;
import java.time.Month;

/**
 * Program to demonstrate LocalDate Class APIs.
 * @author javaguides.net
 *
 */
public class LocalDateExamples {

    public static void main(String[] args) {
        checkIfYearIsLeapYear();
    }

    private static void checkIfYearIsLeapYear() {
        LocalDate localDate1 = LocalDate.of(2017, Month.JANUARY, 1);
        LocalDate localDate2 = LocalDate.of(2016, Month.JANUARY, 1);

        if (localDate1.isLeapYear()) {
            System.out.println(localDate1.getYear() + " is a leap year");
        } else {
            System.out.println(localDate1.getYear() + " is not a leap year");
        }

        if (localDate2.isLeapYear()) {
            System.out.println(localDate2.getYear() + " is a leap year");
        } else {
            System.out.println(localDate2.getYear() + " is not a leap year");
        }
    }
}
```

Output:

```
2017 is not a leap year
2016 is a leap year
```

## 7. LocalDate API to Convert String to LocalDate in java

*LocalDate* class provides below API to convert from String to LocalDate in Java.

- •*static LocalDate parse(CharSequence text)* - Obtains an instance of LocalDate from a text string such as 2007-12-03.
- •*static LocalDate parse(CharSequence text, DateTimeFormatter formatter)* - Obtains an instance of LocalDate from a text string using a specific formatter.

```
package com.ramesh.java8.datetime.api;

import java.time.LocalDate;
import java.time.format.DateTimeFormatter;

/**
 * Program to demonstrate LocalDate Class APIs.
 * @author javaguides.net
 *
 */
public class LocalDateExamples {

    public static void main(String[] args) {
        convertStringToLocalDate();
    }

    private static void convertStringToLocalDate() {
        // ISO Date
        LocalDate localDate = LocalDate.parse("2017-05-03",
DateTimeFormatter.ISO_LOCAL_DATE);
        System.out.println(localDate);

        // yyyy/MM/dd pattern
        LocalDate localDate1 = LocalDate.parse("2017/05/12",
DateTimeFormatter.ofPattern("yyyy/MM/dd"));
        System.out.println(localDate1);
        // MMM dd, yyyy pattern
        LocalDate localDate2 = LocalDate.parse("May 05, 2017",
DateTimeFormatter.ofPattern("MMM dd, yyyy"));
        System.out.println(localDate2);

        // dd-MMM-yyyy pattern
        LocalDate localDate3 = LocalDate.parse("12-Jan-2017",
DateTimeFormatter.ofPattern("dd-MMM-yyyy"));
        System.out.println(localDate3);

        // dd-LL-yyyy pattern
        LocalDate localDate4 = LocalDate.parse("12-01-2017",
DateTimeFormatter.ofPattern("dd-LL-yyyy"));
        System.out.println(localDate4);
    }
}
```

Output:

```
2017-05-03
2017-05-12
2017-05-05
2017-01-12
2017-01-12
```

## 8. LocalDate APIs to Convert LocalDate to String in java

`LocalDate` class provides below API to convert from LocalDate to String in Java.

- *`String format(DateTimeFormatter formatter)`* - Formats this date using the specified formatter.

```
package com.ramesh.java8.datetime.api;

import java.time.LocalDate;
import java.time.format.DateTimeFormatter;

/**
 * Program to demonstrate LocalDate Class APIs.
 * @author javaguides.net
 *
 */
public class LocalDateExamples {
    public static void main(String[] args) {
        convertLocalDatetoString();
    }

    private static void convertLocalDatetoString() {
        // ISO Date
        LocalDate localDate = LocalDate.now();
        DateTimeFormatter dateFormatter = DateTimeFormatter.ISO_LOCAL_DATE;
        System.out.println(localDate.format(dateFormatter));

        // yyyy/MM/dd pattern
         DateTimeFormatter dateFormatter1 =
DateTimeFormatter.ofPattern("yyyy/MM/dd");
         System.out.println(localDate.format(dateFormatter1));

        // MMMM dd, yyyy pattern
        DateTimeFormatter dateFormatter2 = DateTimeFormatter.ofPattern("MMMM dd,
yyyy");
        System.out.println(localDate.format(dateFormatter2));

        // dd-MMM-yyyy pattern
        DateTimeFormatter dateFormatter3 = DateTimeFormatter.ofPattern("dd-MMM-
yyyy");
        System.out.println(localDate.format(dateFormatter3));

        // dd-LL-yyyy pattern
        DateTimeFormatter dateFormatter4 = DateTimeFormatter.ofPattern("dd-LL-
yyyy");
        System.out.println(localDate.format(dateFormatter4));
    }
}
```

Output:

```
2018-08-10
2018/08/10
August 10, 2018
10-Aug-2018
10-08-2018
```

## References

https://docs.oracle.com/javase/8/docs/api/java/time/LocalDate.html

## Related Java 8 Date and Time Posts

- Date and Time API Guide
- Java 8 - LocalTime Class API Guide
- Java 8 - LocalDate Class API Guide
- Java 8 - LocalDateTime Class API Guide
- Java 8 - ZonedDateTime Class API Guide
- Java 8 - Duration Class API Guide
- Java 8 - Instant Class API Guide