

# Java 8 - LocalDateTime Class API Guide


The *LocalDateTime* is used to represent a combination of date and time.

This is the most commonly used class when we need a combination of date and time. The class offers a variety of APIs and we will look at some of the most commonly used ones.

The *java.time.LocalDateTime* class is an immutable class which represents a date-time without time-zone information such as '2018-08-12T10:35:55'.

*LocalDateTime* class provides lots of APIs/Methods to deal with local date and time so in this post we will discuss few important and frequently used *LocalDateTime* APIs/Methods with Examples.

In order understand better, let's categories methods into their usage and explained with examples.

```
<<Java Class>>  
 LocalDateTime  
java.time
```

## LocalDateTime APIs/Methods

### 1. LocalDateTime APIs to Current Date-Time and Specific Date-Time Object

- *static LocalDateTime now()* - Obtains the current date-time from the system clock in the default time-zone.
- *static LocalDateTime now(Clock clock)* - Obtains the current date-time from the specified clock.
- *static LocalDateTime now(ZoneId zone)* - Obtains the current date-time from the system clock in the specified time-zone.

### 2. LocalDateTime APIs to get year, month, day from LocalDateTime

- *int getYear()* - Gets the year field.
- *Month getMonth()* - Gets the month-of-year field using the Month enum.
- *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. LocalDateTime APIs to get Hour, Minute, Second from LocalDateTime

- *int getHour()* - Gets the hour-of-day field.
- *int getMinute()* - Gets the minute-of-hour field.
- *int getNano()* - Gets the nano-of-second field.
- *int getSecond()* - Gets the second-of-minute field.

#### 4. LocalDateTime APIs to add or subtract years, months, days, hours, minutes and seconds to LocalDateTime

- *LocalDateTime plusDays(long days)* - Returns a copy of this LocalDateTime with the specified number of days added.
- *LocalDateTime plusHours(long hours)* - Returns a copy of this LocalDateTime with the specified number of hours added.
- *LocalDateTime plusMinutes(long minutes)* - Returns a copy of this LocalDateTime with the specified number of minutes added.
- *LocalDateTime plusMonths(long months)* - Returns a copy of this LocalDateTime with the specified number of months added.
- *LocalDateTime plusNanos(long nanos)* - Returns a copy of this LocalDateTime with the specified number of nanoseconds added.
- *LocalDateTime plusSeconds(long seconds)* - Returns a copy of this LocalDateTime with the specified number of seconds added.
- *LocalDateTime plusWeeks(long weeks)* - Returns a copy of this LocalDateTime with the specified number of weeks added.
- *LocalDateTime plusYears(long years)* - Returns a copy of this LocalDateTime with the specified number of years added.
- *LocalDateTime minusDays(long days)* - Returns a copy of this LocalDateTime with the specified number of days subtracted.
- *LocalDateTime minusHours(long hours)* - Returns a copy of this LocalDateTime with the specified number of hours subtracted.
- *LocalDateTime minusMinutes(long minutes)* - Returns a copy of this LocalDateTime with the specified number of minutes subtracted.
- *LocalDateTime minusMonths(long months)* - Returns a copy of this LocalDateTime with the specified number of months subtracted.
- *LocalDateTime minusNanos(long nanos)* - Returns a copy of this LocalDateTime with the specified number of nanoseconds subtracted.
- *LocalDateTime minusSeconds(long seconds)* - Returns a copy of this LocalDateTime with the specified number of seconds subtracted.
- *LocalDateTime minusWeeks(long weeks)* - Returns a copy of this LocalDateTime with the specified number of weeks subtracted.
- *LocalDateTime minusYears(long years)* - Returns a copy of this LocalDateTime with the specified number of years subtracted.

#### 5. LocalDateTime APIs to compare LocalDateTime objects in Java

- *boolean isAfter(ChronoLocalDateTime<?> other)* - Checks if this date-time is after the specified date-time.
- *boolean isBefore(ChronoLocalDateTime<?> other)* - Checks if this date-time is before the specified date-time.
- *boolean isEqual(ChronoLocalDateTime<?> other)* - Checks if this date-time is equal to the specified date-time.

- `int compareTo(ChronoLocalDateTime<?> other)` - Compares this date-time to another date-time.

## 6. LocalDateTime API to convert from LocalDateTime to LocalDate in Java

- `LocalDate toLocalDate()` - Gets the LocalDate part of this date-time.

## 7. LocalDateTime APIs to convert from LocalDateTime to LocalTime in Java

- `LocalTime toLocalTime()` - Gets the LocalTime part of this date-time.

Let's discuss above APIs with examples

# 1. LocalDateTime APIs to Current Date-Time and Specific Date-Time Object

LocalDateTime class provides below APIs to create the current date-time and specific date-time object respectively.

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

```
import java.time.Clock;
import java.time.LocalDateTime;
import java.time.Month;
import java.time.ZoneId;

/**
 * Program to demonstrate LocalDateTime Class APIs.
 * @author javaguides.net
 *
 */

public class LocalDateTimeExample {

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

    private static void createLocalDateTime() {
        // Current date time
        LocalDateTime dateTime = LocalDateTime.now();
    }
}
```

```

System.out.println(dateTime);

// Current date time from specified time-zone
LocalDateTime dateTime2 = LocalDateTime.now(ZoneId.of("UTC"));
System.out.println(dateTime2);

// Current date time from specified clock
LocalDateTime dateTime3 = LocalDateTime.now(Clock.systemUTC());
System.out.println(dateTime3);

// Specific date time
LocalDateTime dateTime4 = LocalDateTime.of(2017, Month.JULY, 12, 10, 35, 55);
System.out.println(dateTime4);

}

}

```

Output:

```

2018-08-10T18:08:43.787
2018-08-10T12:38:43.789
2018-08-10T12:38:43.789
2017-07-12T10:35:55

```

## 2. LocalDateTime APIs to get year, month, day from LocalDateTime

LocalDateTime class provides below APIs to get year, month, day from LocalDateTime.

- *int getYear()* - Gets the year field.
- *Month getMonth()* - Gets the month-of-year field using the Month enum.
- *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.LocalDateTime;

/**
 * Program to demonstrate LocalDateTime Class APIs.
 * @author javaguides.net
 *
 */

```

```

public class LocalDateTimeExample {

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

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

```

Output:

```

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

```

### 3. LocalDateTime APIs to get Hour, Minute, Second from LocalDateTime

LocalDateTime class provides below APIs to get Hour, Minute, Second from LocalDateTime.

- *int getHour()* - Gets the hour-of-day field.
- *int getMinute()* - Gets the minute-of-hour field.
- *int getNano()* - Gets the nano-of-second field.
- *int getSecond()* - Gets the second-of-minute field.

```

import java.time.LocalDateTime;

/**
 * Program to demonstrate LocalDateTime Class APIs.
 * @author javaguides.net
 *
 */

public class LocalDateTimeExample {

```

```

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

private static void getHourMinuteSecondfromLocalDateTime() {
    LocalDateTime dateTime = LocalDateTime.now();
    System.out.println("Hour : " + dateTime.getHour());
    System.out.println("Minute : " + dateTime.getMinute());
    System.out.println("Second : " + dateTime.getSecond());
    System.out.println("Nano : " + dateTime.getNano());
}

}

```

## 4. LocalDateTime APIs to add or subtract years, months, days, hours, minutes and seconds to LocalDateTime

LocalDateTime class provides below APIs to add or subtract years, months, days, hours, minutes and seconds to LocalDateTime.

- *LocalDateTime plusDays(long days)* - Returns a copy of this LocalDateTime with the specified number of days added.
- *LocalDateTime plusHours(long hours)* - Returns a copy of this LocalDateTime with the specified number of hours added.
- *LocalDateTime plusMinutes(long minutes)* - Returns a copy of this LocalDateTime with the specified number of minutes added.
- *\_LocalDateTime plusMonths(long months) \_* - Returns a copy of this LocalDateTime with the specified number of months added.
- *LocalDateTime plusNanos(long nanos)* - Returns a copy of this LocalDateTime with the specified number of nanoseconds added.
- *LocalDateTime plusSeconds(long seconds)* - Returns a copy of this LocalDateTime with the specified number of seconds added.
- *LocalDateTime plusWeeks(long weeks)* - Returns a copy of this LocalDateTime with the specified number of weeks added.
- *LocalDateTime plusYears(long years)* - Returns a copy of this LocalDateTime with the specified number of years added.
- *LocalDateTime minusDays(long days)* - Returns a copy of this LocalDateTime with the specified number of days subtracted.
- *LocalDateTime minusHours(long hours)* - Returns a copy of this LocalDateTime with the specified number of hours subtracted.
- *LocalDateTime minusMinutes(long minutes)* - Returns a copy of this LocalDateTime with the specified number of minutes subtracted.

- *LocalDateTime minusMonths(long months)* - Returns a copy of this LocalDateTime with the specified number of months subtracted.
- *LocalDateTime minusNanos(long nanos)* - Returns a copy of this LocalDateTime with the specified number of nanoseconds subtracted.
- *LocalDateTime minusSeconds(long seconds)* - Returns a copy of this LocalDateTime with the specified number of seconds subtracted.
- *LocalDateTime minusWeeks(long weeks)* - Returns a copy of this LocalDateTime with the specified number of weeks subtracted.
- *LocalDateTime minusYears(long years)* - Returns a copy of this LocalDateTime with the specified number of years subtracted.

```
/**
 * Program to demonstrate LocalDateTime Class APIs.
 * @author javaguides.net
 *
 */
public class LocalDateTimeExample {

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

    private static void addOrSubtractUsingLocalDateTime() {
        LocalDateTime dateTime = LocalDateTime.now();
        // LocalDateTime's plus methods
        System.out.println("Addition of days : " + dateTime.plusDays(5));
        System.out.println("Addition of months : " + dateTime.plusMonths(15));
        System.out.println("Addition of years : " + dateTime.plusYears(5));
        System.out.println("Addition of Hours : " + dateTime.plusHours(2));
        System.out.println("Addition of Minutes : " + dateTime.plusMinutes(30));
        System.out.println("Addition of Seconds : " + dateTime.plusSeconds(20));

        // LocalDateTime's minus methods
        System.out.println("Subtraction of days : " + dateTime.minusDays(5));
        System.out.println("Subtraction of months : " + dateTime.minusMonths(15));
        System.out.println("Subtraction of years : " + dateTime.minusYears(5));
        System.out.println("Subtraction of Hours : " + dateTime.minusHours(2));
        System.out.println("Subtraction of Minutes : " + dateTime.minusMinutes(30));
        System.out.println("Subtraction of Seconds : " + dateTime.minusSeconds(20));
    }

}
```

Output:

```
Addition of days : 2018-08-15T18:14:13.385
Addition of months : 2019-11-10T18:14:13.385
Addition of years : 2023-08-10T18:14:13.385
Addition of Hours : 2018-08-10T20:14:13.385
Addition of Minutes : 2018-08-10T18:44:13.385
Addition of Seconds : 2018-08-10T18:14:33.385
Subtraction of days : 2018-08-05T18:14:13.385
Subtraction of months : 2017-05-10T18:14:13.385
Subtraction of years : 2013-08-10T18:14:13.385
Subtraction of Hours : 2018-08-10T16:14:13.385
Subtraction of Minutes : 2018-08-10T17:44:13.385
Subtraction of Seconds : 2018-08-10T18:13:53.385
```

## 5. LocalDateTime APIs to compare LocalDateTime objects in Java

LocalDateTime class provides below APIs compare LocalDateTime objects in Java.

- *`boolean isAfter(ChronoLocalDateTime<?> other)`* - Checks if this date-time is after the specified date-time.
- *`boolean isBefore(ChronoLocalDateTime<?> other)`* - Checks if this date-time is before the specified date-time.
- *`boolean isEqual(ChronoLocalDateTime<?> other)`* - Checks if this date-time is equal to the specified date-time.
- *`int compareTo(ChronoLocalDateTime<?> other)`* - Compares this date-time to another date-time.

```
import java.time.LocalDateTime;

/**
 * Program to demonstrate LocalDateTime Class APIs.
 * @author javaguides.net
 *
 */

public class LocalDateTimeExample {

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



```

private static void compareLocalDateTimeObjects() {
    LocalDateTime dateTime1 = LocalDateTime.of(2017, 05, 22, 10, 55, 25);
    LocalDateTime dateTime2 = LocalDateTime.of(2017, 06, 11, 05, 35, 26);
    LocalDateTime dateTime3 = LocalDateTime.of(2017, 05, 22, 10, 55, 25);

    // Using isBefore() method
    if (dateTime1.isBefore(dateTime2)) {
        System.out.println("dateTime1 is before dateTime2");
    }

    // Using isAfter() method
    if (dateTime2.isAfter(dateTime3)) {
        System.out.println("dateTime2 is after dateTime3");
    }

    // Using isEqual() method
    if (dateTime1.isEqual(dateTime3)) {
        System.out.println("dateTime1 is equal to dateTime3");
    }

    // Using compareTo() method
    if (dateTime1.compareTo(dateTime3) == 0) {
        System.out.println("dateTime1 is equal to dateTime3");
    }

}
}

```

Output:

```

dateTime1 is before dateTime2
dateTime2 is after dateTime3
dateTime1 is equal to dateTime3
dateTime1 is equal to dateTime3

```

## 6. LocalDateTime API to convert from LocalDateTime to LocalDate in Java

LocalDateTime class provides below APIs to convert from LocalDateTime to LocalDate in Java.

- *LocalDate toLocalDate()* - Gets the LocalDate part of this date-time.

```

import java.time.LocalDate;
import java.time.LocalDateTime;

/**
 * Program to demonstrate LocalDateTime Class APIs.
 * @author javaguides.net
 *
 */
public class LocalDateTimeExample {

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

    private static void convertLocalDateTimeToLocalDate() {
        LocalDateTime dateTime = LocalDateTime.now();
        System.out.println(dateTime);

        LocalDate localDate = dateTime.toLocalDate();
        System.out.println(localDate);
    }
}

```

Output:

```

2018-08-10T18:16:35.802
2018-08-10

```

## 7. LocalDateTime APIs to convert from LocalDateTime to LocalTime in Java

LocalDateTime class provides below APIs to convert from LocalDateTime to LocalTime in Java.

- *LocalTime toLocalTime()* - Gets the LocalTime part of this date-time.

```

import java.time.LocalDateTime;
import java.time.LocalTime;

/**
 * Program to demonstrate LocalDateTime Class APIs.
 * @author javaguides.net
 *
 */

```

```
public class LocalDateTimeExample {  
  
    public static void main(String[] args) {  
        convertLocalDateTimeToLocalTime();  
    }  
  
    private static void convertLocalDateTimeToLocalTime() {  
        LocalDateTime dateTime = LocalDateTime.now();  
        System.out.println(dateTime);  
  
        LocalTime localDate = dateTime.toLocalTime();  
        System.out.println(localDate);  
    }  
  
}
```

Output:

```
2018-08-10T18:17:08.598  
18:17:08.598
```

## References

<https://docs.oracle.com/javase/8/docs/api/java/time/LocalDateTime.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](#)