LocalDateTime

The class LocalDateTime is a combination of LocalDate and LocalTime that keeps such values as 2017-12-03T22:30.

It still doesn't store information on a time-zone. It could be used to store a date and time of a transaction in a payment system.
As in the LocalTime class, time is represented to nanosecond precision.

Creating LocalDateTime and current time

An instance of LocalDateTime that represents **this** current moment can be obtained as below:
LocalDateTime now = LocalDateTime.now(); // this moment

The class has static methods of and parse to create instances:

LocalDateTime dt1 = LocalDateTime.of(2017, 11, 25, 22, 30); // 25 November 2017, 22:30
LocalDateTime dt2 = LocalDateTime.parse("2017-11-25T22:30"); // 25 November 2017, 22:30

It's also possible to obtain an instance from the instances of LocalDate and LocalTime, like this:

LocalDate date = LocalDate.of(2017, 11, 25); // 2017-11-25
LocalTime time = LocalTime.of(21, 30); // 21:30

LocalDateTime dateTime = LocalDateTime.of(date, time); // 2017-11-25T21:30

or by using special instance methods of LocalDate and LocalTime:

LocalDate date = LocalDate.of(2017, 11, 25); // 2017-11-25
LocalTime time = LocalTime.of(21, 30); // 21:30

LocalDateTime dateTime1 = date.atTime(time); // 2017-11-25T21:30
LocalDateTime dateTime2 = time.atDate(date); // 2017-11-25T21:30

LocalDateTime: from years to minutes

Now let's observe some methods of the LocalDateTime class. We've already created an instance dateTime to represent the 25 of November, 2017, 10:30 pm :
LocalDateTime dateTime = LocalDateTime.of(2017, 11, 25, 22, 30); // 25 November 2017, 22:30

The class LocalDateTime has methods for obtaining units of date

and time, such as a month, day of the month, hour and minute:

```
int month = dateTime.getMonthValue(); // 11
int day = dateTime.getDayOfMonth(); // 25
int hour = dateTime.getHour(); // 22
int minute = dateTime.getMinute(); // 30
```

The class also has instance methods toLocalDate and toLocalTime to get the date and time as the whole parts of LocalDateTime:

```
LocalDate dateOf = dateTime.toLocalDate(); // 2017-11-25
LocalTime timeOf = dateTime.toLocalTime(); // 22:30
```

Arithmetic methods of LocalDateTime

The class has methods to add, subtract and alter years, months, days, hours, minutes, seconds as well as LocalDate and LocalTime. Let's explore them with a different example:

```
LocalDateTime endOf2017 = LocalDateTime.of(2017, 12, 31, 23, 59, 59); //
2017-12-31T23:59:59
```

This is how by adding one second we get into another year or move by years:

```
LocalDateTime beginningOf2018 = endOf2017.plusSeconds(1); //
2018-01-01T00:00
LocalDateTime beginningOf2020 = beginningOf2018.plusYears(2); //
2020-01-01T00:00
```

We can also alter the constituents of the LocalDateTime by indicating its values:

```
LocalDateTime beginningOf2020 = beginningOf2018.withYear(2020); //
2020-01-01T00:00
```

As you see, LocalDateTime is another immutable class from the java.time package. It represents a combination of LocalDate and LocalTime.