Theory: Comparing dates and time

22 minutes 0 / 5 problems solved

The classes `LocalDate`, `LocalTime`, `LocalDateTime` have methods for comparing their instances according to their position on the timeline. The methods compare instances as they go in chronological order (or the order of time).

§1. The method compareTo

The method `compareTo` compares **this** instance and another one passed as the method's argument. It returns *0* if they are equal, a ***negative*** value if this instance is less than the other, and a ***positive*** value if this instance is greater.

Here is an example of comparing two instances of the class `LocalDate`:

```
LocalDate date1 = LocalDate.parse("2017-01-02");
LocalDate date2 = LocalDate.parse("2017-12-12");

date1.compareTo(date1); // 0, date1 and date1 are equal
date1.compareTo(date2); // -11, negative value => date1 is less than date2

date2.compareTo(date1); // 11, positive value => date2 is greater than date1
```

The class `LocalTime` has the same method, that returns either *0*, *1* or *-1*:

```
LocalTime time1 = LocalTime.parse("15:30:10");
LocalTime time2 = LocalTime.parse("17:50:30");

time1.compareTo(time1); // 0, time1 and time1 are equal
time1.compareTo(time2); // -1, time1 is less than time2
time2.compareTo(time1); // 1, time2 is greater than time1
```

as well as `LocalDateTime`:

```
LocalDateTime dateTime1 = LocalDateTime.parse("2017-01-01T20:30"); // 1 January 2017, 20:30
LocalDateTime dateTime2 = LocalDateTime.parse("2017-01-02T23:00"); // 2 January 2017, 23:00

dateTime1.compareTo(dateTime1); // 0, dateTime1 and dateTime are equal
dateTime1.compareTo(dateTime2); // -1, dateTime1 is less than dateTime2
dateTime2.compareTo(dateTime1); // 1, dateTime2 is greater than dateTime1
```

§2. The methods isAfter, isBefore and isEqual

The classes have also some concise methods to compare dates and time on a timeline that return `boolean` value.

- The method `isAfter` returns `true`, only if this instance is strictly after another instance passed as the argument, otherwise, the method returns `false`.
- The method `isBefore` returns `true`, only if this instance is strictly before an instance passed as the argument, otherwise, the method returns `false`.
- The method `isEqual` returns `true`, if instances are equal, otherwise, the method returns `false`.

Here is an example of comparing two instances of the `LocalDate` class.

```
LocalDate date1 = LocalDate.of(2017, 11, 30);
LocalDate date2 = LocalDate.of(2017, 12, 1);

date1.isEqual(date1); // true
date1.isEqual(date2); // false

date1.isBefore(date2); // true
date1.isBefore(date1); // false
date2.isBefore(date1); // false



date2.isAfter(date1); // true

date2.isAfter(date2); // false

date1.isAfter(date2); // false
```

Here is an example of comparing two instances of the `LocalTime` class.

```
LocalTime time1 = LocalTime.of(14, 20); // 14:20
LocalTime time2 = LocalTime.of(15, 55); // 15:55
LocalTime time3 = LocalTime.of(16, 40); // 16:40

time1.isBefore(time2); // true
time3.isBefore(time2); // false
time3.isBefore(time3); // false

time2.isAfter(time1);  // true

time2.isAfter(time3);  // false
```

And here is an example of comparing two instances of the `LocalDateTime` class.

```
LocalDateTime dateTime1 = LocalDateTime.parse("2017-12-
01T21:30"); // 1 December 2017, 21:30
LocalDateTime dateTime2 = LocalDateTime.parse("2017-12-
02T21:30"); // 2 December 2017, 21:30

dateTime1.isEqual(dateTime2); // false
dateTime2.isEqual(dateTime2); // true

dateTime1.isBefore(dateTime2); // true
dateTime1.isAfter(dateTime2); // false
dateTime2.isAfter(dateTime1); // true
```

Keep in mind, that the class `LocalTime` doesn't have the method `isEqual`. You can use the method `equals` instead.

We have learned now how to compare standard classes representing dates and time. Unsurprisingly, it can be done in almost the same unified way.

Report a typo

79 users liked this theory. **1** didn't like it. **What about you?**

Start practicing

Comments (5) Hints (0) Useful links (0) Show discussion