Theory: LocalTime

17 minutes 0 / 5 problems solved

Skip this topic Start practicing

1527 users solved this topic. Latest completion was about 3 hours ago.

Current topic:

LocalTime

Topic depends on:

✓ Objects Stage 3

Topic is required for:

LocalDateTime

Table of contents:

There's a special class to represent the time of day in Java. The `LocalTime` class represents daytime in hours-minutes-seconds format, such as `06:30` or `11:45:30`. It doesn't store information about the date or a time zone. Time is stored with nanosecond precision (for example, `13:45:30.123456789`).
The `LocalTime` class could be used to store things like an opening and closing time of a shop or a train schedule.

This class belongs to the `java.time` package and pretty much resembles the `LocalDate` class.

§1. Creating LocalTime, current time and constants

First of all, let's see how do we create an instance of the `LocalTime` class. An instance that stores the current time can be created as shown below:

```
LocalTime now = LocalTime.now();
```

If we want to pass a specific time to an instance, we should employ either of the two static methods `of` and `parse` to create an instance of `LocalTime` :

```
LocalTime.of(11, 45);        // 11:45
LocalTime.of(11, 45, 30);    // 11:45:30
LocalTime.parse("11:45:30"); // 11:45:30 (hours, minutes, seconds)
```

In the first line, second and nanosecond fields will be set to zero.

The hour of a day is an `int` number from 0 to 23, while minutes and seconds are numbers from 0 to 59. Nanoseconds can be any integer numbers from 0 to 999,999,999. The following code throws an exception:

```
LocalTime.of(24, 1, 1); // it throws DateTimeException (24 is an invalid value for hours)
```

It's also possible to create an instance of `LocalTime` by using static methods `ofSecondOfDay` and `ofNanoOfDay`. In this case, we should indicate seconds and nanoseconds of a day respectively:

```
LocalTime time = LocalTime.ofSecondOfDay(12345); // 03:25:45
LocalTime nanotime = LocalTime.ofNanoOfDay(1234567890); // 00:00:01.234567890
```

There are some predefined constants in the `LocalTime` class as well :

```
LocalTime.MIN; // 00:00
LocalTime.MAX; // 23:59:59.999999999
LocalTime.NOON; // 12:00
LocalTime.MIDNIGHT; // 00:00
```

Remember that it is a good practice to use constants whenever it is possible!

§2. LocalTime: hours, minutes and seconds

Now let's discuss a bunch of useful methods of the `LocalTime` class.

Let's suppose we have an instance of `LocalTime` :

```
LocalTime time = LocalTime.of(11, 45, 30); // 11:45:30
```

By using the following methods we can get hours, minutes, seconds and nanoseconds of it:

```
time.getHour();   // 11
time.getMinute(); // 45
time.getSecond(); // 30
time.getNano();   // 0, nanoseconds
```

Another useful method is `toSecondOfDay`. It returns the time in seconds of the day from the given `LocalTime` instance. For our example, it'll be:

```
time.toSecondOfDay(); // 42330
```

§3. Arithmetic methods of LocalTime

The class also has methods to add and subtract hours, minutes, seconds and nanoseconds:

```
LocalTime time1 = time.plusHours(5); // 16:45:30
LocalTime time2 = time.plusHours(22); // 09:45:30
LocalTime time3 = time.minusMinutes(10); // 11:35:30
LocalTime time4 = time.minusSeconds(30); // 11:45
```

The following methods return a copy of an instance with one altered part of the time:

```
LocalTime time1 = time.withHour(23); // 23:45:30
LocalTime time2 = time.withMinute(50); // 11:50:30
LocalTime time3 = time.withSecond(0); // 11:45
```

Keep in mind, that `LocalTime` class is **immutable** and all considered methods return a new instance.

As you can see, the `LocalTime` class provides enough methods that will simplify your job and save your time when you'll have to deal with hours, minutes and seconds.

Report a typo

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

Start practicing

Comments (2) Hints (0) Useful links (0) Show discussion