Instant

The Instant Class

One of the core classes of the Date-Time API is the `Instant` class, which represents the start of a nanosecond on the timeline. This class is useful for generating a time stamp to represent machine time.

```
Instant timestamp = Instant.now();
```

A value returned from the `Instant` class counts time beginning from the first second of January 1, 1970 (1970-01-01T00:00:00Z) also called the `EPOCH`. An instant that occurs before the epoch has a negative value, and an instant that occurs after the epoch has a positive value.

The other constants provided by the Instant class are `MIN`, representing the smallest possible (far past) instant, and `MAX`, representing the largest (far future) instant.

Invoking `toString()` on an `Instant` produces output like the following:

```
2013-05-30T23:38:23.085Z
```

This format follows the ISO-8601 standard for representing date and time.

The `Instant` class provides a variety of methods for manipulating an `Instant`. There are `plus()` and `minus()` methods for adding or subtracting time. The following code adds 1 hour to the current time:

```
Instant oneHourLater = Instant.now().plus(1, ChronoUnit.HOURS);
```

There are methods for comparing instants, such as `isAfter()` and `isBefore()`. The `until()` method returns how much time exists between two `Instant` objects. The following line of code reports how many seconds have occurred since the beginning of the Java epoch.

```
long secondsFromEpoch = Instant.ofEpochSecond(0L).until(Instant.now(),
                        ChronoUnit.SECONDS);
```

The `Instant` class does not work with human units of time, such as years, months, or days. If you want to perform calculations in those units, you can convert an `Instant` to another class, such as `LocalDateTime` or `ZonedDateTime`, by binding the `Instant` with a time zone. You can then access the value in the desired units. The following code converts an `Instant` to a `LocalDateTime` object using the `ofInstant()` method and the default time zone, and then prints out the date and time in a more readable form:

```
Instant timestamp;

LocalDateTime ldt = LocalDateTime.ofInstant(timestamp, ZoneId.systemDefault());
System.out.printf("%s %d %d at %d:%d%n", ldt.getMonth(), ldt.getDayOfMonth(),
                  ldt.getYear(), ldt.getHour(), ldt.getMinute());
```

The output will be similar to the following:

```
MAY 30 2021 at 18:21
```

Either a `ZonedDateTime` or an `OffsetDateTime` object can be converted to an `Instant` object, as each maps to an exact moment on the timeline. However, the reverse is not true. To convert an `Instant` object to a `ZonedDateTime` or an `OffsetDateTime` object requires supplying time zone, or time zone offset, information.

Last update: *January 27, 2022*

Time Zone and Offset ← **Instant** → Parsing and Formatting

Home > Tutorials > The Date Time API > Instant