Java 8 Date and Time API

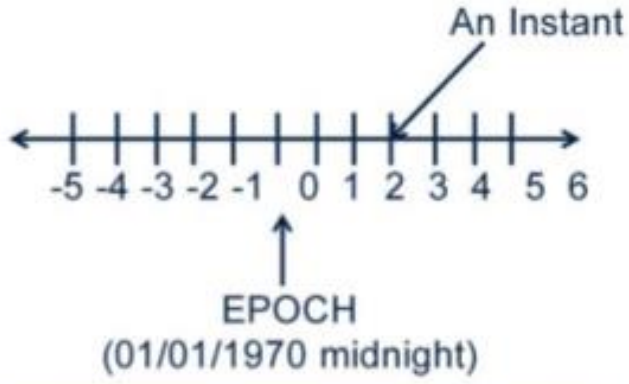
Date and Time API

- Added in Java SE 8 under java.time package.
- Enhanced API to make extremely easy to work with Date and Time.
- Immutable API to store date and time separately.
  - Instant
  - LocalDate
  - LocalTime
  - LocalDateTime
  - ZonedDateTime
- It has also added classes to measure date and time amount.
  - Duration
  - Period
- Improved way to represent units like day and months.
- Generalised parsing and formatting across all classes.

Instant class

The Instant Class

- An object of instant represent point on the time line.
- The reference point is the standard java epoch.
- This class is useful to represent machine timestamp.




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

- The static method "now" of Instant class is used to represent current time.

Instant Calculations

The Instant class also has several methods which can be used to make calculations relative to an Instant. Some (not all) of these methods are:

plusSeconds()
plusMillis()
plusNanos()
minusSeconds()
minusMillis()
minusNanos()

```
Instant now = Instant.now();
Instant later = now.plusSeconds(3);
Instant earlier = now.minusSeconds(3);
```

Instant class is useful for generating a time stamp to represent machine time. A value returned from the Instant class counts time beginning from the first second of January 1, 1970. This value is known as EPOCH.

LocalDate class

The LocalDate Class

- It represent date without time and zone.
- Useful to represent date events like birthdate .
- Following table shows important methods of LocalDate:

Method	Uses
now	A static method to return today's date.
of	Creates local date from year, month and date.
getXXX ()	Used to return various part of date.
plusXXX()	Add the specified factor and return a LocalDate.
minusXXX ()	Subtracts the specified factor and return a LocalDate.
isXXX()	Performs checks on LocalDate and returns Boolean value.
withXXX()	Returns a copy of LocalDate with the factor set to the given value.

```
LocalDate now = LocalDate.now();
LocalDate independence = LocalDate.of(1947, Month.AUGUST, 15);
System.out.println("Independence:"+ independence);
System.out.println("Today:"+now);
System.out.println("Tomorrow:"+ now.plusDays(1));
System.out.println("Last Month:"+ now.minusMonths(1));
System.out.println("Is leap?:"+ now.isLeapYear());
System.out.println("Move to 30th day of month:"+ now.withDayOfMonth(30));
```

ZonedDateTime class

The ZonedDateTime Class

- It stores all date and time fields, to a precision of nanoseconds, as well as a time-zone and zone offset.
- Useful to represent arrival and departure time in airline applications.
- Following table shows important methods of ZonedDateTime:

Method	Uses
now	A static method to return today's date.
of	Overloaded static method to create zoned date time object.
getXXX ()	Used to return various part of ZonedDateTime.
plusXXX()	Add the specified factor and return a ZonedDateTime.
minusXXX( )	Subtracts the specified factor and return a ZonedDateTime.
isXXX()	Performs checks on ZonedDateTime and returns Boolean value.
withXXX()	Returns ZonedDateTime with the factor set to the given value.

```
ZonedDateTime currentTime = ZonedDateTime.now();
ZonedDateTime currentTimeInParis = ZonedDateTime.now(ZoneId.of("Europe/Paris"));
System.out.println("India:" + currentTime);
System.out.println("Paris:" + currentTimeInParis);
```

Period and Duration classes

Period and Duration

- The Period class models a date-based amount of time, such as five days, a week or three years.
- Duration class models a quantity or amount of time in terms of seconds and nanoseconds. It is used represent amount of time between two instants.
- Following table shows important and common methods of both:

Method	Uses
between	Use to create either Period or Duration between LocalDates.
of	Creates Period/Duration based on given year, months & days.
ofXXX()	Creates Period/Duration based on specified factors.
getXXX ()	Used to return various part of Period/Duration.
plusXXX()	Add the specified factor and return a LocalDate.
minusXXX(	Subtracts the specified factor and return a LocalDate.
isXXX()	Performs checks on LocalDate and returns Boolean value.
withXXX()	Returns a copy of LocalDate with the factor set to the given value.

```
LocalDate start = LocalDate.of(1947, Month.AUGUST, 15);
LocalDate end = LocalDate.now(); //18/02/2015
Period period = start.until(end);

System.out.println("Days:"+ period.get(ChronoUnit.DAYS));
System.out.println("Months:"+period.get(ChronoUnit.MONTHS));
System.out.println("Years:"+ period.get(ChronoUnit.YEARS));
```

Formatting and Parsing Date and Time

Formatting and Parsing Date and Time

- Java SE 8 adds DateTimeFormatter class which can be used to format and parse the date and time.
- To either format or parse, the first step is to create instance of DateTimeFormatter.
- Following are few important methods available on this to create DateTimeFormatter.

Method	Uses
ofLocalizedDate(dateStyle)	Date style formatter from locale
ofLocalizedTime(timeStyle)	Time style formatter from locale
ofLocalizedDateTime(dateTimeStyle)	Date and time style formatter from locale
ofPattern(StringPattern)	Custom style formatter from string

- Once formatter object created, parsing/formatting is done by using parse() and format() methods respectively. These methods are available on all major date and time classes .

```
DateTimeFormatter formatter = DateTimeFormatter.ofLocalizedDate(FormatStyle.MEDIUM);
LocalDate currentDate = LocalDate.now();
System.out.println(currentDate.format(formatter));
```

The following example shows how to parse a text string into date.

```
DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
String text = "12/02/2015";
LocalDate date = LocalDate.parse(text, formatter);
System.out.println(date);
```

Conversions

1. To convert from **java.time.LocalDate to java.sql.Date**:

```
java.sql.Date.valueOf(localDate);
```

2. To convert from **java.sql.Date to java.time.LocalDate**:

```
sqlDate.toLocalDate();
```

3. To convert **java.sql.Date to java.util.Date**

Ex.

```
java.sql.Date mdate = resultSet.getDate("hiredate");
java.util.Date udate = new java.util.Date(mdate.getTime());
```

4. To convert **java.util.Date to java.sql.Date**

Ex.

```
java.util.Date udate=new Date();
java.sql.Date mdate = new java.sql.Date(udate.getTime());
```

Thank You!