Date and Time API (JSR 310)

Intro

Problem

- До Java 8 для работы с датой и временем использовались классы:
  - `java.util.Date`
  - `java.util.Calendar`
- Какие проблемы при их использовании
  - не потокобезопасные
  - изменяемые объекты
  - временная зона даты – это временная зона JVM по умолчанию
  - месяца начинаются с нуля

Problem

- Использовали сторонние библиотеки (например: **Joda-Time**)
- *Joda-Time is the de facto standard date and time library for Java prior to Java SE 8* (c) **joda.org**

Solution

- JSR 310: Date and Time API
- Содержит классы:
  - **неизменные** (**immutable**),
  - **потокобезопасные** (**thread-safe**)
  - с **продуманным дизайном**

java.time

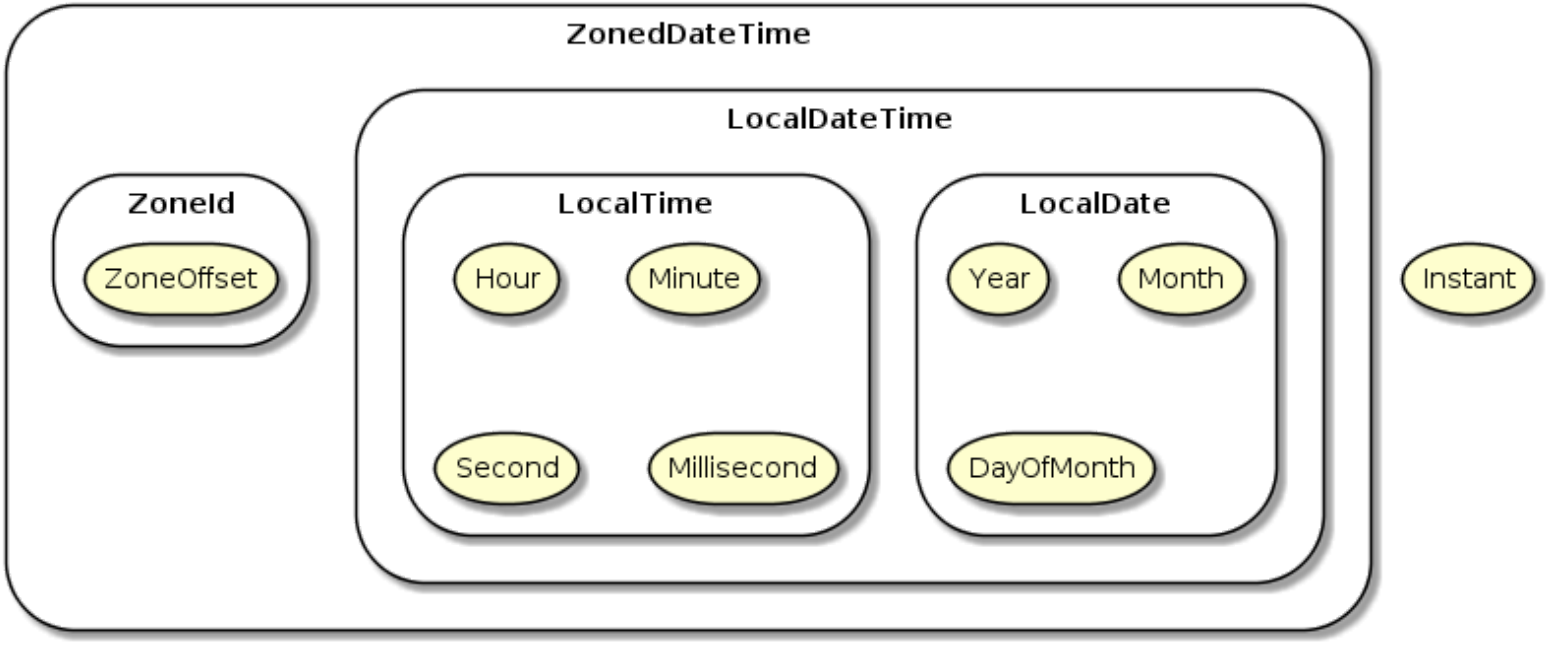
Basics

- `java.time` — содержит, часто используемые классы.
- `java.time.format` — класс для преобразования даты и форматирования.
- `java.time.temporal` — класс для конвертирования даты и время, и для внесения в них корректировок (например добавить 30 дней к текущей дате).

Basics

- java.time.temporal.Temporal
  - LocalTime
  - LocalDate
  - LocalDateTime
  - Instant
- java.time.temporal.TemporalAmount
  - Period
  - Duration

Basics


ZonedDateTime
LocalDateTime
ZoneId
ZoneOffset
LocalTime
Hour
Minute
Second
Millisecond
LocalDate
Year
Month
DayOfMonth
Instant

Temporal

Temporal

- isSupported(TemporalUnit): boolean
- with(TemporalAdjuster): Temporal
- with(TemporalField, long): Temporal
- plus(TemporalAmount): Temporal
- plus(long, TemporalUnit): Temporal
- minus(TemporalAmount): Temporal
- minus(long, TemporalUnit): Temporal
- until(Temporal, TemporalUnit): long

ChronoField

- HOUR_OF_AMPM
- HOUR_OF_DAY
- MINUTE_OF_DAY
- MINUTE_OF_HOUR
- SECOND_OF_DAY
- SECOND_OF_MINUTE
- NANO_OF_DAY
- NANO_OF_SECOND

ChronoUnit

- NANOS
- MICROS
- MILLIS
- SECONDS
- MINUTES
- HOURS
- HALF_DAYS
- DAYS

ChronoUnit

- WEEKS
- MONTHS
- YEARS
- DECADES
- CENTURIES
- MILLENNIA
- ERAS
- FOREVER

LocalTime

Creating `LocalTime`

- `LocalTime.now(): LocalTime`
- `LocalTime.of(int, int): LocalTime`
- `LocalTime.of(int, int, int): LocalTime`
- `LocalTime.of(int, int, int, int): LocalTime`

Methods

- `getHour(): int`
- `getMinute(): int`
- `getSecond(): int`
- `getNano(): int`
- `get(ChronoField): int`

Compare `LocalTime`

- `isAfter(LocalTime): boolean`
- `isBefore(LocalTime): boolean`
- `equals(LocalTime): boolean`

Set units

- `with(ChronoField, int): LocalTime`
- …

Change units

- `plus(int, ChronoUnit): LocalTime`
- `minus(int, ChronoUnit): LocalTime`
- …

`LocalDate`

Same as `LocalTime` + few methods

`LocalDateTime`

Same as `LocalTime` + few methods

`Instant`

Same as `LocalTime` + few methods

TemporalAmount

TemporalAmount

- `get(TemporalUnit): long`
- `getUnits(): List<TemporalUnit>`
- `addTo(Temporal): Temporal`
- `subtractFrom(Temporal): Temporal`

Duration

Duration

- time range
- use for:
  - hours
  - minutes
  - seconds
  - millis
  - nanos

`Period`

- time range
- use for:
  - years
  - months
  - days

DateTimeFormatter

`DateTimeFormatter`

Класс `DateTimeFormatter` используется в Java 8 при форматировании и разборе даты.

Creating

- `ofPattern(String, Locale)`: DateTimeFormatter

```
DateTimeFormatter formatter =
        DateTimeFormatter.ofPattern("MMMM, dd, yyyy HH:mm:ss",
```

Pattern

Символ	Что означает	Пример
`y`	год в эре	`2014`, `14`
`M/L`	месяц (название или номер)	`9`, `09`, `Sep`, `September`, `S`
`d`	день месяца	`17`
`E`	день недели	`Вт`, `вторник`
`h`	время в 12-часовом формате	`1`
`H`	часы в 24-часовом формате	`13`
`m`	минуты	`32`
`s`	секунды	`11`

Символ	Что означает	Пример
`S`	миллисекунды	`109`

Parsing

- `parse(CharSequence text)` - конвертация строки, которая содержит дату и время, в объект `LocalDateTime`. При этом используется формат строки вида `2007-12-03T10:15:30`.
- `parse(CharSequence text, DateTimeFormatter formatter)` - конвертация строки, которая содержит дату и время, в объект `LocalDateTime` с использованием указанного формата.

Example

```
DateTimeFormatter formatter1 =
        DateTimeFormatter.ofPattern("MMMM d, yyyy HH:mm:ss");
LocalDateTime localDateTime =
        LocalDateTime.parse("июня 5, 2018 12:10:56", formatter1
System.out.println(localDateTime);

DateTimeFormatter formatter2 =
        DateTimeFormatter.ofPattern("MMMM d, yyyy");
LocalDate localDate = LocalDate.parse("июня 5, 2018", formatter
System.out.println(localDate);
```

Example

```
LocalDateTime dateTime = LocalDateTime.now();
DateTimeFormatter formatter =
        DateTimeFormatter.ofPattern("MMMM, dd, yyyy HH:mm:ss",
System.out.println(dateTime.format(formatter));
```