3. Essential Java Classes

3a. Date and Time in Java SE 8

Some New Date and Time Classes

Package java.time

- LocalDate - a date without a time-zone in the ISO-8601 calendar system, such as 2007-12-03
- LocalTime - a time without time-zone, such as 10:15:30
- LocalDateTime - a date-time without a time-zone, such as 2007-12-03T10:15:30
- Duration - a time-based amount of time, such as '34.5 seconds‘
- Period - a date-based amount of time, such as '2 years, 3 months and 4 days‘

Some Date Enums

- java.time.DayOfWeek - a day-of-week, such as 'Tuesday‘
- java.time.Month – a month-of-year, such as 'July‘
- java.time.**temporal**.ChronoField – a standard set of fields provide field-based access to manipulate a date, time or date-time
- java.time.**temporal**.ChronoUnit – a standard set of date periods units

LocalDate Class

- Is an immutable date-time object that represents a date, often viewed as year-month-day
- Other date fields, such as day-of-year, day-of-week and week-of-year, can also be accessed
- This class does not store or represent a time or time-zone
- The ISO-8601 calendar system is the modern civil calendar system used today in most of the world. It is equivalent to the Gregorian calendar system
- The LocalDate class doesn't have a public constructor

How to Create LocalDate

- now() - obtains the current date from the system clock in the default time-zone
- of(int year, int month, int dayOfMonth) -obtains an instance of LocalDate from a year, month and day
- of(int year, **Month** month, int dayOfMonth) - obtains an instance of LocalDate from a year, month and day

Date Formatting

- format(DateTimeFormatter formatter) method formats the date using the specified formatter
- Formatter can be created as follows

DateTimeFormatter formatter =

DateTimeFormatter.ofPattern("dd.MM.yyyy");

- Details are here:

http://docs.oracle.com/javase/8/docs/api/java/time/format/DateTimeFormatter.html

Exercise

- Print current date, please

Print Current Date

See 33a1CurrentDate project for the full text

LocalDate Comparison Methods

- compareTo(LocalDate other) - compares this date to another date
- equals(Object obj) - checks if this date is equal to another date
- isAfter(LocalDate other) - checks if this date is after the specified date
- isBefore(LocalDate other) - checks if this date is before the specified date

LocalDate Check/Length Methods

- isLeapYear() - checks if the year is a leap year
- lengthOfMonth() - returns the length of the month represented by this date
- lengthOfYear() - returns the length of the year represented by this date

LocalDate Plus Methods

- plusDays(long daysToAdd) - returns a copy of this LocalDate with the specified number of days added
- plusMonths(long monthsToAdd) - adds the specified period in months
- plusWeeks(long weeksToAdd) - adds the specified period in weeks
- plusYears(long yearsToAdd) – adds the specified period in years

Exercise

- Print the following:
  - Current date
  - Date in 6 weeks
  - Date 4 month before
  - Date in 45 days

Date Manipulations

See 33a2DateManipulations project for the full text

Get Field of a Date (1 of 2)

- get(ChronoField field) - gets the value of the specified field from this date as an int
- ChronoField enum is a standard set of date fields: DAY_OF_MONTH, DAY_OF_WEEK, DAY_OF_YEAR, HOUR_OF_DAY, MONTH_OF_YEAR, YEAR, SECOND_OF_DAY etc.
- See for details:

http://docs.oracle.com/javase/8/docs/api/java/time/temporal/ChronoField.html

Get Field of a Date (2 of 2)

- getDayOfWeek() - gets the day-of-week field, which is an enum DayOfWeek (MONDAY, TUESDAY, WEDNESDAY etc.)
- getDayOfMonth() - gets the day-of-month field
- getDayOfYear() - gets the day-of-year field
- getMonth() - gets the month-of-year field using the Month enum (JANUARY, FEBRUARY, etc.)

Exercise

- Create a method that gets some date and returns next bank day

Exercise

See 33a3NextBankDate project for the full text

Some Other LocalDate Methods

- until(LocalDate endDate, ChronoUnit unit) - calculates the amount of time until endDate in terms of the specified unit
- withDayOfMonth(int dayOfMonth) - returns a copy of this date with the day-of-month altered
- withMonth(int month) - returns a copy of this date with the month-of-year altered
- withYear(int year) - returns a copy of this date with the year altered

Home Tasks

1. How old are you in days and months? What day of week is your birthday?
2. Calculate Orthodox Easter and Trinity dates for the current decade

http://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%93%D0%B0%D1%83%D1%81%D1%81%D0%B0_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B4%D0%B0%D1%82%D1%8B_%D0%9F%D0%B0%D1%81%D1%85%D0%B8

LocalDateTime Class

- LocalDateTime is an immutable date-time object that represents a date-time, often viewed as year-month-day-hour-minute-second
- Time is represented to nanosecond precision
- It is a description of the date, as used for birthdays, combined with the local time as seen on a wall clock
- It cannot represent an instant on the time-line without additional information such as an offset or time-zone

LocalDateTime Methods I

- now()
- compareTo(), equals(), isAfter(), isBefore()
- plusDays(), plusMonths(), plusWeeks(), plusYears()
- plusHours(), plusMinutes(), plusSeconds()
- get(ChronoField field), getDayOfWeek(), getDayOfMonth(), getDayOfYear(), getMonth()
- getHour(), getMinute(), getSecond()

LocalDateTime Methods II

- of(int year, int month, int dayOfMonth, int hour, int minute, int second) - obtains an instance of LocalDateTime from year, month, day, hour, minute and second, setting the nanosecond to zero
- format(DateTimeFormatter formatter) - formats this date-time using the specified formatter

Print Current Date and Time

```
public static void main(String[] args){
   LocalDateTime currDateTime = LocalDateTime.now();
   DateTimeFormatter formatter =
     DateTimeFormatter.ofPattern("dd.MM.yyyy HH:mm:ss");
   String txt = currDateTime.format(formatter);
   System.out.println("Current date and time:" + txt);
}
```

See 33a6CurrentDateTime project for the full text

Converting from java.util.Date

- Convert java.util.Date to java.time.LocalDateTime:

```
Date ts = new Date();
Instant instant = Instant.ofEpochMilli(ts.getTime());
LocalDateTime res = LocalDateTime.ofInstant(instant,
    ZoneId.systemDefault());
```

- Convert java.util.Date to java.time.LocalDate:

```
Date date = new Date();
Instant instant = Instant.ofEpochMilli(date.getTime());
LocalDate res = LocalDateTime.ofInstant(instant,
    ZoneId.systemDefault()).toLocalDate();
```

Converting to java.util.Date

- Convert java.time.LocalDateTime to java.util.Date:

```
LocalDateTime ldt = LocalDateTime.now();
Instant instant =
    ldt.atZone(ZoneId.systemDefault()).toInstant();
Date res = Date.from(instant);
```

- Convert java.time.LocalDate to java.util.Date:

```
LocalDate ld = LocalDate.now();;
Instant instant = ld.atStartOfDay().
    atZone(ZoneId.systemDefault()).toInstant();
Date res = Date.from(instant);
```

Manuals

- http://docs.oracle.com/javase/tutorial/datetime/index.html