Exercises module 9 – Date & time

9.1 - Scheduling

In programming, we often need to execute tasks (code) at regular intervals. In the exercise below we need to execute something that runs about once every 7 days.

1. Create a **LocalDate** object for the date **11-november 2016** named it **lastRunDate**

2. Create a second **LocalDate** object named **today** with the date **16-November 2016**

3. Make a test if the **today** date is within 7 days of the **lastRunDate.**

    a. If the **today** date is within **lastRunDate plus 7 days** then print out **"Not time yet"**
    b. If **more** than **7 days** have passed since **lastRunDate**, we print out **"Time to run again"**

4. Verify that the function works with different dates, like 20-November 2016.

5. Add so that the program also prints out the **number of days** since the **LastRunDate.**

9.2 – Period

1. In Java, we have three different methods to deal with a duration of time.
    a. Duration
    b. Period
    c. ChronoUnit.between

    Visit this page to read more about these classes and what the difference is between them:
    https://docs.oracle.com/javase/tutorial/datetime/iso/period.html

2. Let's write a program that can calculate the following:

```
Next year's christmas: 2018-12-24
Current date: 2017-01-22

Next year's christmas is in 1 years 11 months and 2 days

A total of 701 days!
```

3. First create two LocalDate variables that contains the current date and the date of Christmas next year and print them to the screen.

4. Create **Period** instance that contains the difference between these two dates.

5. Print out the number of **years**, **months** and **days** to next year's Christmas

6. It is also fun to know the total number of days and to get that we can use the **ChronoUnit.DAYS.between** method. Print out the result!

9.3 Formatting date and time

1. When we run this code:

```
LocalDateTime now = LocalDateTime.of(2017, 9, 19, 14, 5, 0);
System.out.println(now);
```

   We get the following output: **2017-09-19T14:05**

   In this exercise, we will learn how to format and customize the data and time.

2. To customize the printout, we need to first create a **DateTimeFormatter** and pass a formatting pattern string to it like:

```
DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyyMMdd HHmm");
```

3. Then we can use the format method to convert and format the date to a string using:

```
System.out.println(now.format(formatter));
```

   Try the code above and make sure it works!

   You can simplify the code above by writing:

```
System.out.println(now.sformat(DateTimeFormatter.ofPattern("pattern")));
```

4. Visit
   http://docs.oracle.com/javase/8/docs/api/java/time/format/DateTimeFormatter.html#patterns
   and explore the various patterns for formatting the datetime.

5. Try the difference between different number of formatting characters like:

```
System.out.println(now.format(DateTimeFormatter.ofPattern("y")));
System.out.println(now.format(DateTimeFormatter.ofPattern("yy")));
System.out.println(now.format(DateTimeFormatter.ofPattern("yyy")));
System.out.println(now.format(DateTimeFormatter.ofPattern("yyyy")));
```

   Try the behaviour of using 1-4 repeated characters with the ‘M’, ‘E’, ‘h’, ‘d’….

   Can you format the above date so that it matches the following output?

```
Tue-19, 2017
Tuesday-19, 2017
19-September, 2017
02:05 PM
2017/19/9
2017/19/09
```

Questions and concepts to study further on your own:

- What is UTC time?
- What is JodaTime?
- What happens when we get summer and winter time?
- How do you parse a string into a LocalDateTime object?
- What is ISO-8601?
- What happens on the 19-January, 2038?
  https://www.youtube.com/watch?v=QJQ691PTKsA
  https://en.wikipedia.org/wiki/Unix_time
- What is a leap-second?