Java Workbook

06. DateTime and localization

~by Andrzej "Andret2344" Chmiel

Exercise 1: Date range

Given the following code:

```
class MainDateRange {
    public static void main(final String[] args) {
        final LocalDate begin = LocalDate.of(2020, 1, 2);
        final LocalDate end = LocalDate.of(2020, 3, 15);
        getDateRagne(begin, end).forEach(System.out::println);
    }
}
```

Create the `getDateRagne` method, so the code compiles and prints out this:

```
2020-01-22
2020-01-23
2020-01-24
2020-01-25
2020-01-26
2020-01-27
2020-01-28
2020-01-29
2020-01-30
2020-01-31
2020-02-01
2020-02-02
2020-02-03
2020-02-04
2020-02-05
2020-02-06
2020-02-07
2020-02-08
2020-02-09
2020-02-10
2020-02-11
2020-02-12
2020-02-13
2020-02-14
2020-02-15
```

Try to not use any loops.

Exercise 2: Timezone differences

Given the following code:

```
class MainTimeZoneDiff {
    public static void main(final String[] args) {
        System.out.println(timeZonesDiff(
                ZoneId.of("Europe/Warsaw"),
                ZoneId.of("America/Chicago")));

        System.out.println(timeZonesDiff(
                ZoneId.of("Asia/Tokyo"),
                ZoneId.of("America/Los_Angeles")));

        System.out.println(timeZonesDiff(
                ZoneId.of("America/Argentina/Buenos_Aires"),
                ZoneId.of("Europe/Rome")));
    }
}
```

Not changing the `main` method, create the missing `timeZonesDiff` method so the code compiles and prints out this:

```
-7
-17
4
```

Exercise 3: Time distance

Given the following code:

```
class MainTimeDistance {
    public static void main(final String[] args) {
        printTimeDifference(
                LocalDateTime.of(2017, 11, 11, 0, 0),
                LocalDateTime.of(2017, 11, 12, 20, 0));

        printTimeDifference(
                LocalDateTime.of(2017, 11, 21, 7, 14),
                LocalDateTime.of(2017, 11, 25, 2,7));

        printTimeDifference(
                LocalDateTime.of(2017, 11, 22, 10, 30),
                LocalDateTime.of(2017,12, 24, 18, 0));
    }
}
```

Not changing the `main` method, create the missing `printTimeDifference` method so the code compiles and prints out this:

```
1 days 20 hours 0 minutes
3 days 18 hours 53 minutes
32 days 7 hours 30 minutes
```

Exercise 4: Daylight saving time

Given the following code:

```
class MainDaylightSavingTime {
    public static void main(final String[] args) {
        System.out.println(daylightSavingsTime(2016));
        System.out.println(daylightSavingsTime(2017));
        System.out.println(daylightSavingsTime(2018));
    }
}
```

Not changing the `main` method, create the missing `daylightSavingsTime` that for the given year finds the summer to winter daylight saving day in timezone identified by `Europe/Warsaw` without using any loops. So running the code prints out this:

```
2016-10-30
2017-10-29
2018-10-28
```

Exercise 5: Constant random

Given the following code:

```
public class MainConstantRandom {
    public static void main(final String[] args) {
        System.out.println(getRandomOfDate(LocalDateTime.now()));
        System.out.println(getRandomOfDate(LocalDateTime.now()));
        System.out.println(getRandomOfDate(LocalDateTime.now()));
    }
}
```

Create the missing `getRandomOfDate` method that generates a random integer $x \in [0, 1000]$ that will always be the same on a certain day of a year (for instance, the number generated on 23$^{\text{rd}}$ of March will be the same every year). So the method uses `Random` class for randomization and prints out three times the same number.

Exercise 6: Time measurement

Create a `Timer` class that is available in the following use:

```
try (final Timer timer = new Timer()) {
    // do something
}
```

And after the `try` block it automatically prints how many milliseconds the `try`'s body lasted.

Exercise 7: Properties

In an adequate place create a `*.properties` file and using `Properties` file read its content. Keys that are to be in the file:

- `greeting`
- `username`
- `password`

Use the loaded data to print out the greeting and the credentials.

Exercise 8: Localization

Given the following code:

```
record CountryInfo(String name, String capitalCity) {
}
```

Create multiple `*.properties` files (for Germany, Poland, GB, and USA) that contain the full country code in their name, i.e., `country_en_US` and fill them in with keys present in `CountryInfo` class. Using the `ResourceBundle` class, the program should read the file that matches to your system locale and fill in the `CountryInfo` object with them, so printing out the `CountryInfo` object prints out i.e., this:

```
CountryInfo[name=Poland, capitalCity=Warsaw]
```

To manually change the locale, use:

```
Locale.setDefault(new Locale("en", "GB"));
```

Exercise 9: Default localization

In the previous exercise, there is a possibility that the program is run from a location that is not predicted by the localization files. Fix the code so if it is run from such a location, it prints this:

```
CountryInfo[name=unknown, capitalCity=unknown]
```

Make as little changes to the java code as possible.

Exercise 10: Various currencies

Given the following code:

```
class MainCurrencies {
    public static void main(final String[] args) {
        Stream.of(
                        new Locale("pl", "PL"),
                        new Locale("en", "US"),
                        new Locale("en", "GB"),
                        new Locale("de", "DE"),
                        new Locale("cz", "CZ"))
                .map(MainCurrencies::getCurrency)
                .forEach(System.out::println);
    }
}
```

Not changing the `main` method, create the missing `getCurrency` method so the program compiles and prints out this:

```
PLN (Polish Zloty)
USD (US Dollar)
GBP (British Pound)
EUR (Euro)
CZK (Czech Koruna)
```