COMP3222 Mobile App Development

Coursework 2: Kotlin Programming

The aim of this assignment is to assess your understanding of the Kotlin programming language alone. The solution you submit should be implemented as a command line application, *not* as an Android app. (The final assignment will focus purely on app development.)

This assignment contributes 20% to your overall module grade.

1 Preparation

Download `cwk2files.zip` from the Coursework 2 folder in Minerva and unpack the archive. This will give you a directory `cwk2`, configured as a Gradle project. You can use Gradle from the command line to build and run your Kotlin application. You can also import it as a Gradle project into the IntelliJ IDE if you prefer. Consult the `README` file in the project directory for further details.

There is also a subdirectory `data` containing a number of data files. Some of these are provided for testing purposes, whereas others are real datasets containing meteorological data collected at several different weather stations around the country. See the `README` file in the `data` directory for further details.

2 Task

2.1 Basic Solution

Create an application in Kotlin that extracts data on rainfall from any of the provided files and then analyses the data. Your application should require the filename to be specified on the command line. If no filename is provided, the application should display a helpful message to the user and then terminate. If a valid filename is provided, the application should

- Extract weather station name, latitude, longitude and elevation from the file header (the first four lines), then display this information
- Find the records in the dataset having the highest and lowest rainfall measurements, then display the month, year and rainfall value for these records

Here's an example of the expected output when the application is run on the file `bradford.txt`:

```
Station: Bradford
Latitude: 53.813°
Longitude: -1.772°
Elevation: 134 m
Number of records: 1068
Wettest month: June 2007 (261.4 mm)
Driest month: April 2011 (2.0 mm)
```

Make sure that your application behaves sensibly for files containing errors and files containing a valid header but no data—see the `README` file in the `data` directory for further information.

There are many different ways in which this task could be approached, some of which will earn you more marks than others. See Section 3 for advice on this.

2.2 Advanced Solution

1. Modify your application so that displays the range of years contained within a dataset.
2. Modify the application further so that it identifies the wettest and driest *years* in a dataset. At this point, the output generated for `bradford.txt` should look approximately like this:

```
Station: Bradford
Latitude: 53.813°
Longitude: -1.772°
Elevation: 134 m
Number of records: 1068
Years spanned: 1930 to 2018
Wettest year: 2012 (1233.1 mm)
Driest year: 1955 (621.6 mm)
Wettest month: June 2007 (261.4 mm)
Driest month: April 2011 (2.0 mm)
```

3. Modify the application so that it can accept a year as a second command line argument. If this second argument is supplied, the application should display a bar chart of rainfall for that year instead of displaying all the other information. For example, if you supply arguments of `data/bradford.txt` and `2018`, it should produce output like this:

```
  January (107.8): ##################################################
 February ( 46.0): ####################
    March ( 84.8): ########################################
    April ( 82.6): #######################################
      May ( 23.4): ##########
     June (  6.0): ##
     July ( 57.8): ##########################
   August ( 43.2): ####################
September ( 63.0): ############################
  October ( 34.8): ################
 November ( 65.2): ##############################
 December ( 87.2): ########################################
```

The figures in parentheses here are the actual rainfall values for those months.

3 Hints & Tips

30% of the marks in this assignment are for code structure and intelligent use of Kotlin features. You could certainly implement your solution as one large `main` function, but this wouldn't score highly for structure and Kotlin usage. A reasonable solution would define a data structure of some kind to represent a single record, then store all of the data from the file in a collection of these structures. Functions could be used to read the data and perform each analysis task.

A good solution would be highly object-oriented, using classes to represent individual records and the entire dataset, with the latter providing methods to perform each of the analysis tasks. A very good solution would make full use of Kotlin's support for processing collections in a functional style, using lambda functions. There are opportunities in this task for using other Kotlin features such as `when` expressions, nullable types, scope functions, etc. You'll get credit for using such features sensibly.

Feel free to call out to the Java API where you consider it helpful to do so. For example, you may find it easier to use `String.format` or `System.out.printf` to generate formatted output. You could also use Java's `Scanner` class to help you read lines or individual numerical values from the file. Other approaches to reading the data are possible. For example, you could use the `readLines` method of Kotlin's `File` class to read file contents into a list of strings.

4 Submission

Submit your source code as single Zip archive, via the link provided for this purpose in Minerva. You should use Gradle to generate this Zip archive—see the README file for further details.

The deadline for submissions is **10 am on Thursday 5 March**. The standard university penalty of 5% of available marks per day will apply to late work, unless an extension has been arranged with the SSO due to genuine extenuating circumstances.

5 Marking

A total of **40 marks** are available, awarded as follows:

16	Basic functionality
8	Advanced functionality
12	Structure and intelligent use of Kotlin
4	Code style & commenting