Edinburgh Napier
UNIVERSITY

**School of Computing, Edinburgh Napier University**

| | |
|---|---|
| **1. Module number** | *SET07102* |
| **2. Module title** | *Software Development 1* |
| **3. Module leader** | *Kevin Sim* |
| **4. Tutor with responsibility for this Assessment**<br>Student's first point of contact | *Kevin Sim*<br>*k.sim@napier.ac.uk* |
| **5. Assessment** | *Assessment 2 – Weather Data*<br><br>*You should answer FIVE questions from ONE of the three sections provided.*<br><br>*Questions in section 1 attract a maximum of 60% of the available marks.*<br><br>*Questions in section 2 attract a maximum of 80% of the available marks.*<br><br>*Questions in section 3 attract a maximum of 100% of the available marks.* |
| **6. Weighting** | *60% of the overall module mark* |
| 7. Size and/or time limits for assessment | |
| **8. Deadline of submission** | *Hand in to Moodle by 16:00 on Friday 2nd December 2016.* |
| **9. Arrangements for submission** | *Submit a Word document and source code via Moodle. A short (~5 minute) demonstration will be required prior to hand-in during your scheduled practical session on either Thursday 1st or Friday 2nd December.* |
| **10. Assessment Regulations**<br>All assessments are subject to the University Regulations**.** | |
| **11. The requirements for the assessment** | *Write Java programs to answer questions in **one** of the sections attached.* |

| | |
|---|---|
| **12. Special instructions** | *You are required to submit two files. A Word document (or a PDF) that includes:*<br>• *The output from your code and any screenshots*<br><br>*A Zip file containing your Java source code*<br><br>*A short (~5 minute) demonstration of your code will be required prior to hand-in. Failure to demonstrate your code will result in a mark of Zero* |
| **13. Return of work** | *You will receive immediate feedback during the practical session where you demonstrate your work. Further feedback will be supplied via Moodle within 3 weeks of submission* |
| **14. Assessment criteria** | *You will be assessed on the accuracy and the quality of your code. See attached document for marking criteria* |

## Introduction

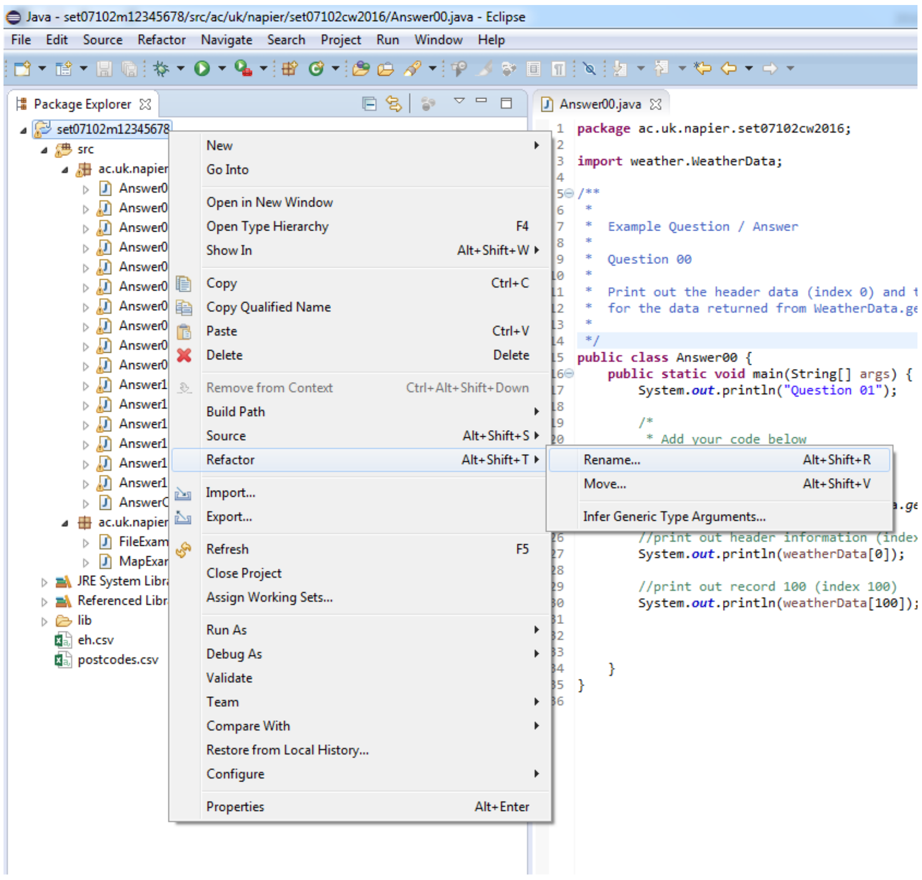
You are supplied with a Java library containing weather data recorded at different locations in the UK during the year 2015. A Second Java library is provided that allows geographic coordinates to be plotted on a world map. You are required to write code to answer questions about the data and output the correct answers.

## Setting up Your Coursework Java Project

You are supplied with a zip file named `set07102m12345678.zip` that contains an eclipse Java project.

Import the project to an eclipse workspace (you should know how by now) and rename the project you imported by replacing the numbers 12345678 with your matriculation number *(To rename a project right click on the project in the package explorer and select refactor → rename from the menu as shown in Figure 1 below)*

Your project folder should now be named as `set07102m` followed by your matriculation number i.e. `set07102m40004938`



```
package ac.uk.napier.set07102cw2016;

import weather.WeatherData;

/**
 *
 *  Example Question / Answer
 *
 *  Question 00
 *
 *  Print out the header data (index 0) and t
 *  for the data returned from WeatherData.ge
 *
 */
public class Answer00 {
    public static void main(String[] args) {
        System.out.println("Question 01");

        /*
         * Add your code below

        //print out header information (index
        System.out.println(weatherData[0]);

        //print out record 100 (index 100)
        System.out.println(weatherData[100]);



    }
}
```

**Figure 1**

The supplied project contains a number of class files and libraries as shown in Figure 1

The package `ac.uk.napier.set07102cw2016` contains 16 files named `Answer00.java` to `Answer15.java.`

You are required to add code to the main method of each file corresponding to the numbers of the five questions you attempt. For example, if you attempt question 9 your code must be invoked in the main method of `Answer09.java`.

Your code should output an answer for each question to the system console.

`Answer00.java` is provided as an example.

Three Java libraries are described below. The ones that you use will depend on which questions that you choose to answer.

**WeatherData**

The project includes a Java library named `WeatherData.jar` which contains weather data observed at weather stations located across the UK during the year 2015. The library contains one class with two public static methods.

The method *`WeatherData.getData()`* returns an array of strings. The data can be read into an array of Strings using the following code

```
String[] weatherData = WeatherData.getData();
```

The first element of the array returned, at index 0, contains header information with descriptions of the data. Each subsequent element contains a single data record with details recorded at a single weather station at a given date and time.

There are a different number of records supplied for each weather station. Potentially a weather station can have a record for each hour during the year but in reality, many locations have far fewer entries. The first 7 elements of the array are printed below.

| | |
|---|---|
| Element 0 | SiteId,SiteName,Latitude,Longitude,Year,Month,Date,Hour,WindSpeed,Temperature |
| Element 1 | 3002,BALTASOUND (3002),60.7490,-0.8540,2015,01,01,0,21,8.70 |
| Element 2 | 3005,LERWICK (S. SCREEN) (3005),60.1390,-1.1830,2015,01,01,0,30,8.60 |
| Element 3 | 3017,KIRKWALL (3017),58.9540,-2.9000,2015,01,01,0,18,8.90 |
| Element 4 | 3023,SOUTH UIST RANGE (3023),57.3580,-7.3970,2015,01,01,0,36,9.50 |
| Element 5 | 3026,STORNOWAY (3026),58.2140,-6.3250,2015,01,01,0,26,9.60 |
| Element 6 | 3031,LOCH GLACARNOCH SAWS (3031),57.7250,-4.8960,2015,01,01,0,30,7.50 |

Each element, other than the first, is a comma-delimited string that needs to be split and parsed to the correct data type before any queries can be run on the data.

You can use the split method of String to split a line of data into an array containing the 10 individual fields. Each record should be split using a comma as the delimiter.

The following description of the split method is taken from the String API

https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#split-java.lang.String-

```
public String[] split(String regex)
```

Splits this string around matches of the given regular expression.

This method works as if by invoking the two-argument `split` method with the given expression and a limit argument of zero. Trailing empty strings are therefore not included in the resulting array.

The string `"boo:and:foo"`, for example, yields the following results with these expressions:

| Regex | Result |
|---|---|
| `:` | `{ "boo", "and", "foo" }` |
| `o` | `{ "b", "", ":and:f" }` |

**Parameters:**

`regex - the delimiting regular expression`

**Returns:**

`the array of strings computed by splitting this string around matches of the given regular expression`

The split data array should be parsed to the data types shown in Table 1 using the parse methods of the appropriate primitive wrapper class

- `public static int parseInt(String s)`
- `public static double parseDouble(String s)`

**Table 1**

| Field Index | Data Type | Column Header | Description |
|---|---|---|---|
| 0 | int | SiteId | Unique Identifier identifying the weather station |
| 1 | String | SiteName | Name of the Weather Station |
| 2 | double | Latitude | Latitude of the Weather Station |
| 3 | double | Longitude | Longitude of the Weather Station |
| 4 | int | Year | 2015 for all records supplied |
| 5 | int | Month | Integer representing the month of the year in the range 1 – 12 |
| 6 | int | Date | Integer representing the day of the month in the range 1 – 31 |
| 7 | int | Hour | Integer representing the hour of the day in the range 0 – 23 |
| 8 | int | WindSpeed | Integer representing wind speed in Km/h >= 0 |
| 9 | double | Temperature | Double representing the temperature in Celsius. Can be negative |

The file `Answer00.java` is supplied as an example of how to use the method *`WeatherData.getData()`*

The second public method supplied with the `WeatherData` package uses the Haversine formula to calculate the distance between 2 geographic coordinates. This is required for questions in section 3. The following Java Documentation describes its use.

```
double getDistanceBetweenPoints(double lat1, double
                    lon1, double lat2, double lon2)

Implementation of the Haversine formula for
calculating the distance between two geographic
coordinates

Parameters:
     lat1 Latitude of coordinate 1
     lon1 Longitude of coordinate 1
     lat2 Latitude of coordinate 2
     lon2 Longitude of coordinate 2
Returns:
     The distance in KM between coordinate 1 and
     coordinate 2
```

**MapGui**

If you decide to attempt the questions in sections 2 or 3 then you will need to use the supplied MapGui Class. The MapGui Class is located in the JMapViewer package and includes two public methods that allow you to display geographic coordinate(s) on a Map.

This is built on the open source package
http://wiki.openstreetmap.org/wiki/JMapViewer

```
void showMap(Coordinate coordinate)


Displays a Map showing the location of the
coordinate

Parameters:
     coordinate A single Coordinate


void showMap(ArrayList<Coordinate> coordinates)


Displays a Map showing the location of each
coordinate in the coordinates ArrayList

Parameters:
     coordinates An ArrayList of type Coordinate
```

The class `MapExample` that is located in the package `ac.uk.napier.set07102cw2016.Examples` shows how to use the libraries described above. Note that the Coordinate class is supplied with the JMapViewer package.

**PostcodeExample**

If you choose to answer questions in section 3 that require the use of the supplied postcode data then an example of how to access the data is provided in the `PostcodeExample` class which is located in the package `ac.uk.napier.set07102cw2016.Examples.`

Two postcode data files are provided. `postcodes.csv` contains 1.6 million UK postcodes and their geographic locations and file `eh.csv` contains the subset of postcodes that start with EH. (Edinburgh)

**Instructions**

You should answer FIVE questions from ONE of the three sections supplied. All of the questions that you answer MUST be from the SAME section. Sections attract different maximum marks.

- Questions in Section 1 attract a maximum of 12% each
- Questions in Section 2 attract a maximum of 16% each
- Questions in Section 3 attract a maximum of 20% each

You are required to modify 5 files from the 15 files named `Answer01.java` – `Answer15.java` supplied. The main method of each of these files is to be the entry point for your answer. Each answer should print output to the system console.

File `AnswerChecker.java` will be executed automatically to check your answers and it is therefore crucial that you follow the steps above and structure your project correctly. Failure to follow these instructions will be penalised.

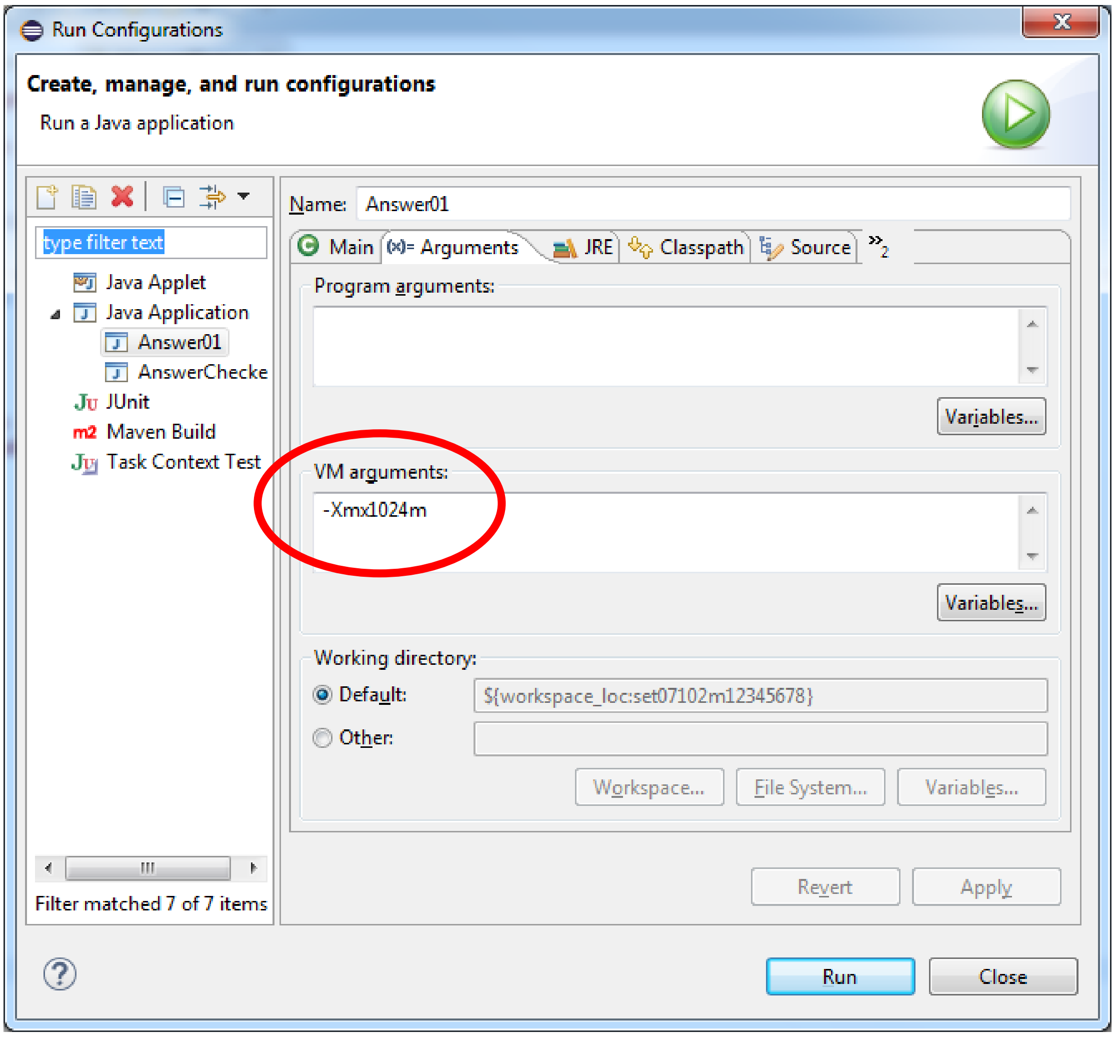
You can design and use as many extra classes and methods as you feel necessary as long as all source code is provided. No additional external libraries or resources should be used (anything included with Java is OK). User defined classes should be placed into separate package(s)

You are supplied with class AnswerChecker to check your results. Running this class will start the 15 answer classes in order and should print something for each of the 5 questions that you attempt.

Due to the large amount of data records supplied, you may encounter memory problems depending on how you choose to answer the questions. To increase the memory available to the JVM add the following VM Arguments to the run configuration.

```
-Xmx1024m
```

This increases the memory available to the JVM to 1024MB. The screenshot below shows where to add this parameter. This is accessed from the `run→run configurations` menu.



Name: Answer01

Program arguments:

VM arguments: -Xmx1024m

Working directory:

Default: ${workspace_loc:set07102m12345678}

Other:

## submission

You are required to submit 2 files no later than the 2nd December 2016 via Moodle.

A word document should be submitted containing the output printed for each of your answers along with any screenshots that are asked for. The word document should be named using the same prefix as your project directory (i.e `set07102m12345678.doc`). A correctly named pdf is also acceptable.

A single zip file should be submitted containing your project directory. The zip file should be named using the same prefix as your project directory (i.e `set07102m12345678.zip`).

## Section 1

**Questions in this section attract a maximum mark of 12% each.**

**You will be marked primarily on the approach taken to answering the questions.**

**Answers, or partial answers, are provided for some of the questions to enable you to gain confidence in your approach.**

The main methods in Answer01.java – Answer05.java must be used to invoke your code.

1) How many different weather stations does the data cover? The Site Id is a unique identifier that identifies a weather station. (Each weather station has thousands of records)

There are 136 Sites

2) What was the lowest temperature recorded? Output the site id, site name, temperature, Year, Month, Date and Hour that the lowest temperature was recorded.

3041,AONACH MOR (3041), -26.70, 2015, 03, 06, 13

3) What were the minimum, maximum and mean temperatures recorded at 6 AM during January 2015 for the weather station named "HEATHROW (3772)"?

min = -6.2 max = 12.4 mean = ???? There are 31 entries for Weather station 3772 on this date

4) How many days of the year did the temperature drop to zero or below anywhere in the UK? The practical and tutorial given during weeks 8 and 9 should help you with this.

Hint : More than 200 but less than 250:

5) What is the most northerly weather station? Print out the ID, name, latitude and longitude of the weather station.

Located at a Latitude of 60.749

## Section 2

**Questions in this section attract a maximum mark of 16% each.**

**You will be marked primarily on the approach taken to answering the questions.**

**Answers or partial answers are provided for some of the questions to enable you to gain confidence in your approach.**

**Questions 6 – 10 must be implemented using an Object Oriented design that uses aggregation between WeatherStation and WeatherReading, as detailed below.**

Additionally for some questions, screenshots are required as detailed below. In addition to any screenshots, your code MUST still output the answer to the system console.

Your object-oriented model should be implemented using two classes.

- WeatherStation
- WeatherReading

Each WeatherStation should maintain a list of WeatherReadings obtained at that location. All fields contained in the data should be stored in the appropriate class. You are free to implement whatever additional classes that you feel necessary. For example a collection of WeatherStation objects can be maintained in your main method or you can encapsulate these in another class if you wish. The main methods in Answer06.java – Answer10.java must be used to invoke your code.

Marks will be awarded for the quality of your object-oriented design, as detailed at the end of this document, as well as for the general approach that you use to answer each question. Marks will also be awarded for correct use of variables, methods, naming conventions, code clarity and appropriate comments as detailed at the end of this document.

6) Which weather station has the fewest weather readings? Output the Id, name and coordinates of the weather station to the system console. Plot the location using the MapGui class supplied. Include a screenshot in your Word document.

```
WeatherStation [siteId=3874 name=SOLENT MRSC (3874)
lat=50.807 lon=-1.208]
```

7) What was the highest temperature recorded? Output the site id, site name, temperature, Year, Month, Date and Hour that the temperature was recorded to the console. Plot the Weather Station where the highest temperature was recorded using the MapGui class supplied. Include a screenshot in your Word document.

```
siteId=3772
```

8) What were the minimum, maximum and mean temperatures recorded at 11 AM during July 2015 for the weather named "DUNKESWELL AERODROME (3840)"? Print the answer to the console. In addition, plot the location of the Weather Station on a map using the MapGUI class supplied, and include a screenshot of this in your Word document.

```
min = 12.1 max = 19.8 mean = ?????
```

9) How many weather stations never recorded a temperature of zero or below? Print the number to the console. Plot the location(s) on a map using the supplied MapGUI class and supply a screenshot in your Word document. The practical and tutorial given during weeks 8 and 9 should help you with this.

10) What is the most southerly weather station? Print out the ID, name, Latitude and Longitude of the weather station to the console. Plot the most southerly weather station on a map using the supplied MapGUI class. Include a screenshot of the plot in your Word document.

```
Located at latitude 49.913
```

## Section 3

**Questions in this section attract a maximum mark of 20% each.**

**You will be marked primarily on the approach taken to answering the questions.**

**Questions 11 – 15 must be implemented using an Object Oriented design that uses aggregation and inheritance as specified below.**

Additionally for some questions, screenshots are required as detailed below. Your code should still output the answer to the system console.

Your model should be implemented using three classes.

- WeatherStation
- WeatherReading
- Postcode

Your classes WeatherStation and Postcode MUST both **extend Coordinate** which is a class included in the JMapViewer package. You will need to override the constructor `Coordinate (double lat, double lon)` in your subclasses.

Each WeatherStation should maintain a list of WeatherReadings obtained at that location. All fields contained in the data should be stored in the appropriate class. You are free to implement whatever additional classes that you feel necessary. For example a collection of WeatherStation Objects can be maintained in your main method or you can encapsulate these in another class if you wish. The main methods in Answer11.java – Answer15.java must be used to invoke your code.

Marks will be awarded for the quality of your object-oriented design, as detailed at the end of this document, as well as for the general approach that you use to answer each question. Marks will also be awarded for correct use of variables, methods, naming conventions, code clarity and appropriate comments as detailed at the end of this document.

11) How many postcodes are within a 5KM radius of the most northerly weather station? Print the answer to the system console and plot these locations on a map using the supplied MapGUI class. Include a screenshot of the map at a suitable zoom level in your Word document.

`16 Post Codes`

12) Which location sustained a wind speed of more than 50 MPH for the most consecutive readings? The data supplied is given in chronological order. Print the answer to the system console and plot the location on a map using the supplied MapGUI class. Include a screenshot at a suitable zoom level in your Word document.

13) What was the difference in MEDIAN temperature during July between FARNBOROUGH (3768) and EDINBURGH/GOGARBANK (3166)? Print the answer to the system console and plot these locations on a map using the supplied MapGUI class. Include a screenshot at a suitable zoom level in your Word document.

14) Which is the most isolated EH postcode? This is calculated as the postcode with the largest distance to its closest neighbour. Use the `eh.csv` file supplied for this task. Output the postcode and location to the console and plot the location using the supplied MapGui class. Include a screenshot in your Word Document.

EH27 ???

15) Which EH postcode is the most densely populated? This is determined as the postcode with the most neighbouring postcodes within a 0.1KM radius (<=). Use the `eh.csv` file supplied for this task. Output the postcode and location to the console and plot the location using the supplied MapGui class. Include a screenshot in your Word Document.

## Advice on coding and Marking Criteria

Your code is being submitted for assessment.

- Your code will be read by a compiler:
    - It should work; it should compile correctly, run without errors and print results to the console.
- Your code will be read by an human (or a lecturer)
    - Your code should be readable and understandable by your lecturers.
    - Tidy the code up before you submit it (eclipse can format your code for you).
        - Delete commented out code.
        - Fix indentation
    - Keep your lines short.
    - Avoid redundant code.
- Use standard Java naming conventions for class names, variables and methods
- Use appropriate comments where required (do not overuse)

If you have chosen to answer questions in section 2 or section 3 then your Object Oriented design will be assessed in terms of

- Naming conventions
- Correct use of getters and setters for accessing private fields
- Implementation of expected Object methods and Constructors
- Use of comments
- Elegance of your solutions and clarity of your code

You should submit your code as a single zip file containing your project directory. You should submit a word document containing your project output and screenshots. We will make comments on the quality of your code and return these to you via Moodle.

Please follow the instructions for naming your files

- Your project folder should now be named as set07102m followed by your matriculation number i.e. set07102m40004938

- Your zip should be named set07102m followed by your matriculation number i.e. set07102m40004938.zip

- Your word document should be named set07102m followed by your matriculation number i.e. set07102m40004938.docx

To zip your project directory select the project folder
(i.e. set07102m40004938) from your eclipse workspace and from the right click context menu select `send to → compressed (zipped) folder`.

Upload a **single zip** file along with your word document to Moodle by the deadline.