COM1003 Java Programming – Semester 2

Java Programming Assignment:
A Fitness activity tracker data explorer

Submission deadline 3pm on Friday 22 May.

Note, this is an individual assignment. You are not allowed to work in groups and your code must be all your own work. You may not use other peoples' code or ideas, e.g., taken from the Internet.
Any form of unfair means (plagiarism or collusion) will be treated as a serious academic offence and will be processed under our University Discipline Regulations. These might result in a loss of marks, a possible failure of the module or even expulsion from our University.
For further details, please visit the UG student handbook section on this topic:
https://sites.google.com/sheffield.ac.uk/comughandbook/general-information/assessment/unfair-means

In this assignment, you will build a programme (a fitness activity tracker data explorer) to visualise and analyse fitness activity data collected for a number of individuals who participated in a study aiming to understand the accuracy of commercial fitness tracker measurements. Participants were monitored using five off-the-shelf fitness activity trackers able to measure distance, number of steps, energy expenditure and heart rate[1]. Gold standard[2] measurements were obtained to assess the accuracy of each fitness activity tracker.

This project will make substantial use of the material you have been taught in the course. In particular, you will employ OO programming, the Java Collections Framework, Java Swing and event handling for the GUI-building.

You will be provided with a substantial number of classes to help you get started. They will take care of various tasks on your behalf and will facilitate your work. They are also meant to facilitate assessing your code.

You must use and include these classes in your submission.
You must not modify the classes that have been provided to you.
You are expected to strictly follow all the guidance provided in this handout otherwise you will incur in unnecessary mark losses.

Before you begin, read all of this document carefully. It is intended to provide you with sufficient detail to get started with the assignment. You need to read this document along with the classes that have been provided to you. They need to be studied together. The comments provided in the code are written to help you. You are also strongly recommended to begin work on the assignment soon, and not leaving it until the last minute.

General Background

In this assignment, you will be using a dataset that was collected with the aim of assessing the accuracy of five fitness activity trackers under controlled conditions (walking / running for 30 minutes on a treadmill). You will be asked to:

- Parse the fitness activity tracker dataset, which is composed of a number of text files that will be used to populate the relevant data structures used in the assignment. Detailed guidance about the fitness activity data and the participants in the dataset is provided in the section *'The fitness activity tracker data'*.
- Display on the console the answer to a number of questions about the dataset. Detailed guidance about the questions is provided in the section *'The questions'*.
- Build a Graphical User Interface (GUI) that will allow exploring the fitness activity data in the dataset provided, according to certain criteria. Your GUI will visualise the fitness activity data for all participants. For each participant, data will include up to four type of measurements: distance, steps, energy expenditure and heart rate. You should be able to display the results interactively. You can find the details about how the GUI will need to operate in the section *'The GUI'*.

[1] Hear rate measurements only available for a subset of trackers.
[2] In this context, we will consider as the "gold standard" measurement the most accurate measurement obtainable with conventional means.

The fitness tracker activity data

In this assignment, you will read a number of text files containing a wide range of measurements collected for each participant. The dataset[3] can be downloaded from MOLE, and your programme will be responsible for loading it from the folder of your choice.

Data for each participant will be in a single text file, and each text file will follow the following format:

Format	Example
Participant: [Participant ID]	**Participant:** P1
Age: [Age]	**Age:** 67
Gender: [F]	**Gender:** F
--------------------------	-----------------------------------
Heart Rate	**Heart Rate**
Count; GS; FT1; FT2; FT3;	**Count; GS; FT1; FT2; FT3;**
[List of recordings*]	1; 59.0; 60.0; 61.0; 67.0;
	2; 66.0; 67.0; 66.0; 69.0;
	3; 69.0; 70.0; 71.0; 77.0;
	4; 44.0; 45.0; 47.0; 50.0;
	5; 87.0; 88.0; 87.0; 85.0;
	6; 62.0; 65.0; 61.0; 62.0;
--------------------------	-----------------------------------
Steps	**Steps**
Count;GS;FT1;FT2;FT3;FT4;FT5;	**Count;GS;FT1;FT2;FT3;FT4;FT5;**
[List of recordings*]	1;1100.0;1100.0;1090.0;990.0;1055.0;1052.0;
	2;1324.0;1320.0;1300.0;1100.0;1299.0;1288.0;
	3;1465.0;1467.0;1500.0;1255.0;1501.0;1499.0;
	4;1890.0;1886.0;1950.0;1735.0;1995.0;1998.0;
--------------------------	-----------------------------------
Energy Expenditure	**Energy Expenditure**
Count;GS;FT1;FT2;FT3;FT4;FT5;	**Count;GS;FT1;FT2;FT3;FT4;FT5;**
[List of recordings*]	1;333.0;334.0;200.0;0.0;150.0;290.0;
	2;258.0;278.0;244.0;20.0;245.0;279.0;
	3;169.0;168.0;201.0;10.0;159.0;156.0;
	4;556.0;554.0;350.0;206.0;455.0;592.0;
--------------------------	-----------------------------------
Distance	**Distance**
Count;GS;FT1;FT2;FT3;FT4;FT5;	**Count;GS;FT1;FT2;FT3;FT4;FT5;**
[List of recordings*]	1;9099.0;9100.0;9090.0;8990.0;9055.0;9020.0;
***** `List of recordings` is an enumeration of numbers following the format specified in the previous line (header). All the provided data files will be following the format shown in this example, i.e. the measurement count (`Count`), the gold standard measurement (`GS`), and the measurements for each fitness tracker, for either fitness trackers `FT1`, `FT2` and `FT3`, or for fitness trackers `FT1`, `FT2`, `FT3`, `FT4` and `FT5`. This will depend on the ability that each fitness tracker has to record a particular measurement type.	

To facilitate your work, you have been already provided with a number of classes that will read the data in the provided text files but that does not parse yet the text files and fills in the appropriate collections to store the fitness tracker activity data. Please, study the class `AbstractDataLoader`. You will need to write the class `DataLoader` that will extend the `AbstractDataLoader` class.

Please, study well the class `AbstractDataLoader` class in combination with this handout. You need follow ALL the specifications provided, otherwise you will lose marks unnecessarily.

[3] The original dataset was collected as part of a research project that obtained ethics approval from the Department of Computer Science at the University of Sheffield. All participants consented to have their data used for research and educational purposes. The dataset in this assignment has been slightly altered to facilitate the tasks requested.

The questions

Your application should be able to answer the following twenty questions[4] and display the results in the console:

Q1. Provide the total number of participants in this dataset.
Q2. Provide the total number of participants that have heart rate measurements.
Q3. Provide the total number of participants that have steps measurements.
Q4. Provide the total number of participants that have distance measurements.
Q5. Provide the total number of participants that have energy expenditure measurements.
Q6. Provide the total count of heart rate measurements in the whole dataset.
Q7. Provide the total count of steps measurements in the whole dataset.
Q8. Provide the total count of distance measurements in the whole dataset.
Q9. Provide the total count of energy expenditure measurements in the whole dataset.
Q10. Provide the total count of heart rate measurements for each fitness tracker (not including the gold standard measurement) for the whole dataset.
Q11. Provide the total count of energy expenditure measurements for fitness tracker FT1 for the whole dataset.
Q12. Provide the total count of steps measurements for fitness trackers FT2, FT3 and FT4 for the whole dataset.
Q13. Provide the total count of distance measurements for fitness tracker FT5 for the whole dataset.
Q14. Provide the list of participant/s with the highest single measurement of steps (across trackers, including GS) and the corresponding number of steps.
Q15. Provide the list of participant/s with the lowest single measurement of steps (across trackers, including GS) and the corresponding number of steps.
Q16. Provide the list of participant/s with the highest single measurement of walked distance (across trackers, including GS) and the corresponding distance.
Q17. Provide the list of participant/s with the lowest single measurement of walked distance (across trackers, including GS) and the corresponding distance.
Q18. Provide the global average heart rate for the whole dataset.
Q19. Provide the list of participant/s with an average individual participant heart rate above the global average heart rate (provide average heart rate value per participant).
Q20. Provide the list of participant/s with an average individual participant heart rate below the global average heart rate (provide average heart rate value per participant).

Note that for many of the questions above, there might be multiple values returned (e.g. Q10 and Q12) or multiple participants satisfying the condition specified (e.g. Q14-Q17 and Q19-Q20).

The output on the console can[5] follow a similar format to the one provided as an example below. It is important that your answers are well reported and intelligible to the user of your programme as this will be assessed.

```
Q1.  Total number of participants: 32

Q2.  Number of participants with heart rate measurements: 12 out of 32

Q10. Total count of heart rate measurements for FT1: 212
     Total count of heart rate measurements for FT2: 134
     Total count of heart rate measurements for FT3: 4
     Total count of heart rate measurements for FT4: 45
     Total count of heart rate measurements for FT5: 4500

Q11. Total count of energy expenditure measurements for FT1: 230

Q12. Total count of energy expenditure measurements for FT2: 258
     Total count of energy expenditure measurements for FT3: 0
     Total count of energy expenditure measurements for FT4: 984

Q14. 3 participant/s with the highest number of steps:
     * Participant ID P3 with number of steps: 22450
     * Participant ID P13 with number of steps: 22450
     * Participant ID P15 with number of steps: 22450
```

4 Questions Q2-Q9 and Q14-Q20 should include gold standard measurements, question Q10-Q13 should NOT include gold standard measurements, and for question Q1, gold standard measurements are actually irrelevant.

5 This guidance is only for illustration purposes. The results shown do not correspond to the dataset you have been provided.

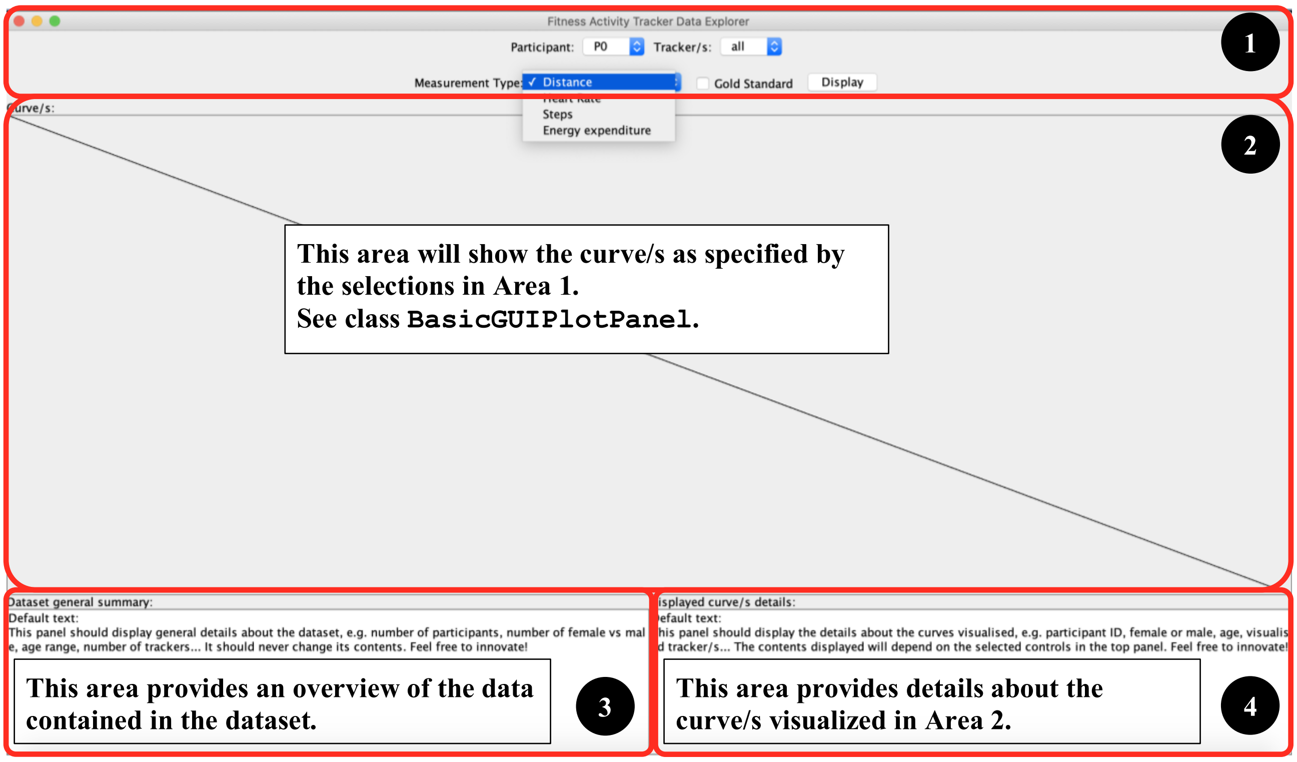
```
        Q19. 5 participant/s with heart rate above global average heart rate (74.1):
             * Participant ID P5 with individual average heart rate 74.3
             * Participant ID P12 with individual average heart rate 78.6
             * Participant ID P25 with individual average heart rate 79.1
             * Participant ID P27 with individual average heart rate 74.15
             * Participant ID P31 with individual average heart rate 75.4
```

You have been provided with the class `AbstractFitnessQuestions` which contains the list of methods that you will need to implement in the class `FitnessQuestions` to answer these questions.
Please, follow really carefully ALL the specifications provided, otherwise you will lose marks unnecessarily.

The GUI

The next figure shows how the GUI could look like. You have been provided with a basic implementation of the main panel (class `AbstractGUIPanel`) showing the four areas highlighted below.

- *Area 1 (top):* Allows the user to specify which data should be shown in Areas 3 and 4. The selections in this panel will control the contents being shown in Areas 2 and 4. Please, make sure that you understand from the provided code which selections are possible. When the button "Display" is clicked, areas 2 and 4 should be updated accordingly.
- *Area 2 (centre):* Visualises the relevant curve/s based on the selections made using the GUI controls in Area 1. The contents of this panel will dynamically change according to the selections in Area 1. Please, make sure that you understand from the provided code which curve/s are expected to be visualised. The class `BasicGUIPlotPanel` provides a basic implementation of the panel where the curves should be plotted.
- *Area 3 (bottom-left):* Shows a general summary describing the dataset. The contents of this panel will not change.
- *Area 4 (bottom-right):* Shows details describing the curves visualised in Area 2 after applying the selections in Area 1. The contents of this panel will dynamically change according to the selections in Area 1.


Fitness Activity Tracker Data Explorer
Participant: P0
Tracker/s: all
Measurement Type: ✓ Distance
Heart Rate
Steps
Energy expenditure
Gold Standard
Display
1
Curve/s:
2
This area will show the curve/s as specified by the selections in Area 1.
See class **BasicGUIPlotPanel**.
Dataset general summary:
Displayed curve/s details:
This area provides an overview of the data contained in the dataset.
3
This area provides details about the curve/s visualized in Area 2.
4

Default text:
This panel should display general details about the dataset, e.g. number of participants, number of female vs male, age range, number of trackers... It should never change its contents. Feel free to innovate!

Default text:
This panel should display the details about the curves visualised, e.g. participant ID, female or male, age, visualised tracker/s... The contents displayed will depend on the selected controls in the top panel. Feel free to innovate!

Areas 2 and 4 will be updated whenever the user clicks on the button "Display" according to the selections in Area 1.

In the example shown in the figure above, the user specified that for participant P0, the distance measurements should be shown for all available trackers for that participant, and gold standard curve should not be shown. When the user clicks the button "Display", the contents of areas 2 and 4 should be updated.

In case the user selects a combination of criteria where there is no data available to be visualised, a relevant message should be displayed in Areas 2 and 4. The programme should NOT crash.

Programming task

The overall aim of the assignment is to produce a Java program that allows browsing fitness activity tracker data for a number of individuals and visually explore the accuracy of the different trackers in the provided dataset. The user will be able to interact with the browser to indicate which participant data, which measurement type, which tracker (all or a specific one), and if the gold standard measurements need to be shown. You will need to write several new classes that operate together with a set of classes that are provided (see below for details).

Your program will work as follows.

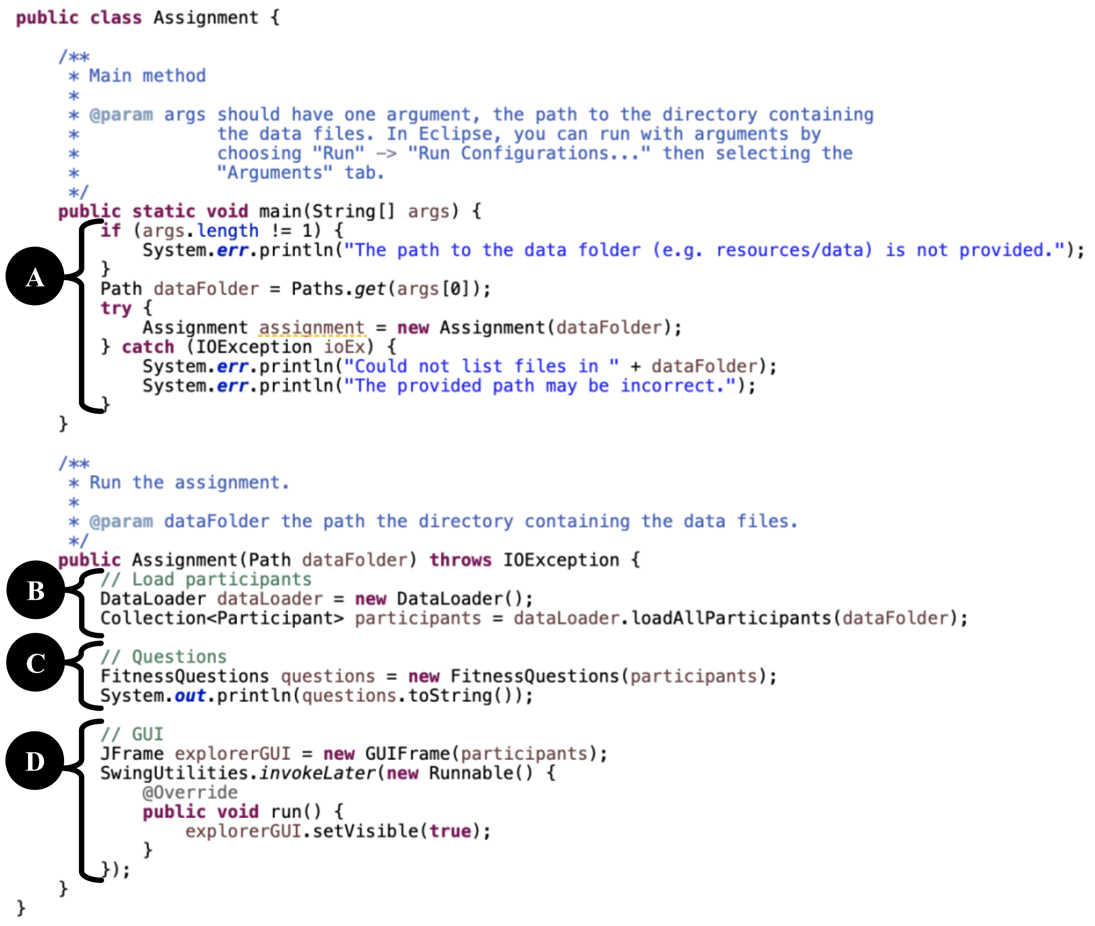
The main class will be **`Assignment`** (i.e. a file called **`Assignment.java`**). You have already been provided with this class, and it already has a `main` method. You SHOULD NOT modify this class. This will be the main class used to run your programme, and when invoked, the main will:

A. Read the command line arguments to obtain the path to the folder containing the data files. This task has been already done on your behalf in the classes provided. An instance of the class `Assignment` is created.

B. Load the participants data from the provided text files containing the participants fitness activity tracker data. To achieve this, you will need to work on the `DataLoader.java` class.

C. Answer the questions provided in "The questions" section in this handout. For that, you will need to work on the `FitnessQuestions.java` class. Note that the answers to Q1-Q20 will be printed to the console thanks to the `System.out.println(questions.toString())` statement.

D. Build and open the GUI, according to the specifications provided in "The GUI" section in this handout. For that, you will need to work on the `GUIPanel.java` and `GUIPlotPanel.java` classes.



```java
public class Assignment {

    /**
     * Main method
     *
     * @param args should have one argument, the path to the directory containing
     *             the data files. In Eclipse, you can run with arguments by
     *             choosing "Run" -> "Run Configurations..." then selecting the
     *             "Arguments" tab.
     */
    public static void main(String[] args) {
        if (args.length != 1) {
            System.err.println("The path to the data folder (e.g. resources/data) is not provided.");
        }
        Path dataFolder = Paths.get(args[0]);
        try {
            Assignment assignment = new Assignment(dataFolder);
        } catch (IOException ioEx) {
            System.err.println("Could not list files in " + dataFolder);
            System.err.println("The provided path may be incorrect.");
        }
    }

    /**
     * Run the assignment.
     *
     * @param dataFolder the path the directory containing the data files.
     */
    public Assignment(Path dataFolder) throws IOException {
        // Load participants
        DataLoader dataLoader = new DataLoader();
        Collection<Participant> participants = dataLoader.loadAllParticipants(dataFolder);

        // Questions
        FitnessQuestions questions = new FitnessQuestions(participants);
        System.out.println(questions.toString());

        // GUI
        JFrame explorerGUI = new GUIFrame(participants);
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                explorerGUI.setVisible(true);
            }
        });
    }
}
```

Classes provided

In addition to the main assignment class (`Assignment.java`) provided in package `assignment2020`, several classes are provided. ***You must use these classes and you must not modify any of the classes in the package*** **`assignment2020.codeprovided`*****. They have been carefully designed to help you accomplish the programming tasks set in this assignment.***
The contents in package `assignment2020.codeprovided` have been organised as follows:

- Package `assignment2020.codeprovided.dataloading`:
  - `AbstractDataLoader.java` is an abstract class that provides implementations of basic file reading utilities for the input text files containing the fitness tracker activity data. Please study well this class as your **`DataLoader.java`** class will need to extend this class and implement some of its abstract methods.
  - `DataParsingException.java` is a helper class that should be thrown whenever a parsing error occurs when loading the data for a participant.
- Package `assignment2020.codeprovided.fitnesstracker`:
  - `Participant.java` is a core class for your assignment. This class stores the participant details including their `name`, `age` and `gender`. The class also stores in a `Map` all the fitness tracker data for a single participant. The `Map` 'maps' the tracker's name (a `String`) to a `Tracker` object.
  - `Tracker.java` is another core class for your assignment. This class uses a `Map` to map a measurement type to a list of measurements (see enum `MeasurementType` and class `Measurement`).
- Package `assignment2020.codeprovided.fitnesstracker.measurements`:
  This package contains a set of classes that are core to represent the data contained in the dataset provided:
  - `MeasurementType.java` is a helper enum that should be used to represent the possible types of measurements in the dataset.
  - `Measurement.java` is a helper abstract class used to represent any of the measurements in the dataset. It stores a measurement `count` and its associated `value`.
  - `Distance.java`, `EnergyExpenditure.java`, `HeartRate.java` and `Steps.java` are the concrete classes representing all the possible measurement types in the dataset. They all extend the abstract class `Measurement`.
  - `MeasurementFactory.java` is a helper class containing a single static method that creates the appropriate `Measurement` instance (`Distance`, `EnergyExpenditure`, `HeartRate` or `Steps`) according to the `MeasurementType` provided.
- Package `assignment2020.codeprovided.handoutquestions`:
  - `AbstractFitnessQuestions.java` is an abstract class that provides the specification of the methods that your **`FitnessQuestions.java`** will need to implement to answer "The Questions". Please, study well this class before implementing the class `FitnessQuestions`.
- Package `assignment2020.codeprovided.gui`:
  - `AbstractGUIPanel.java` is an abstract class that builds the main GUI panel shown in "The GUI" section in this handout. The class **`GUIPanel.java`** will extend this class.
  - `BasicGUIPlotPanel.java` is a concrete class that draws a Rectangle into the Area 2 in the GUI. The class **`GUIPlotPanel.java`** will extend this class and will be responsible for drawing the curve/s as specified in Area 1 in the GUI.
  - `GUIFrame.java` is the class in charge of creating the GUI main `JFrame`. You do not need to modify this class.

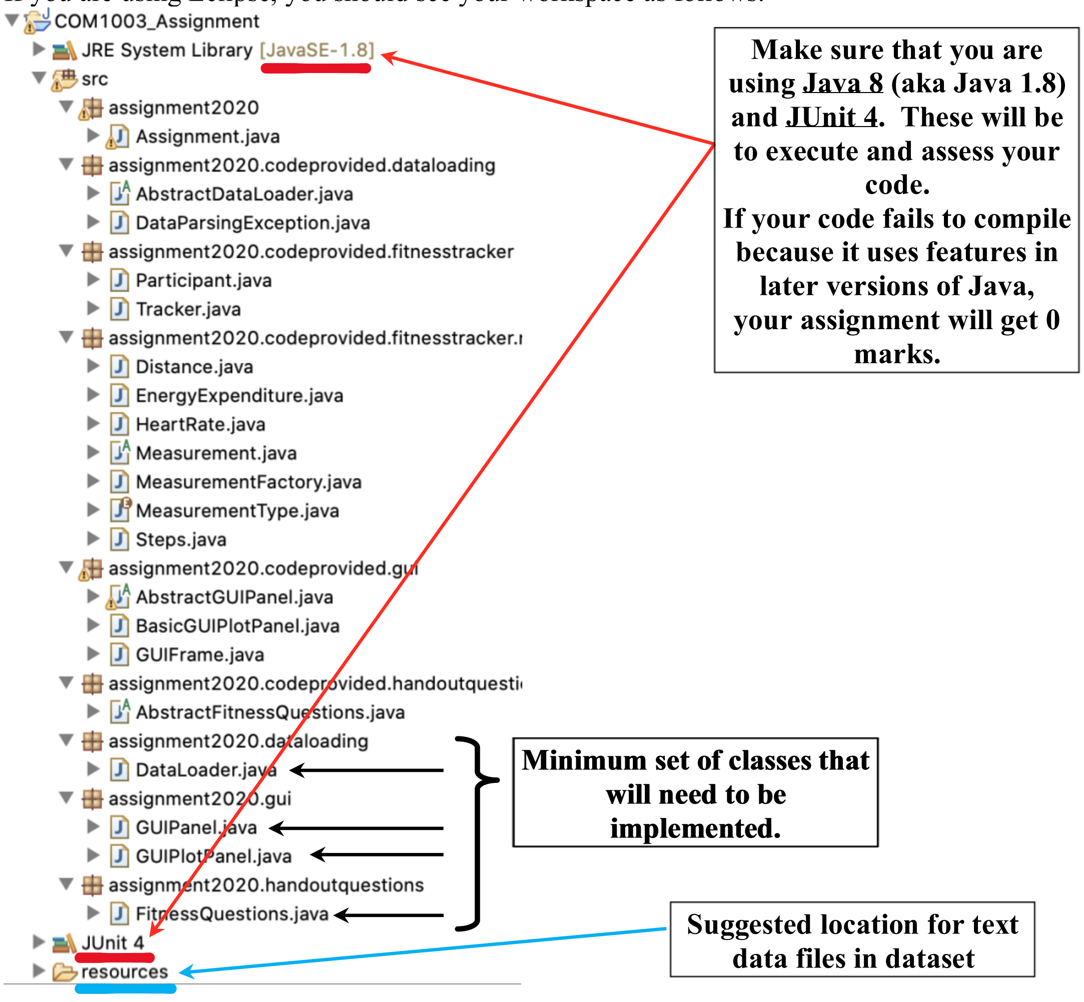
Deliverables

In this assignment, you must fully provide an implementation of four classes. They will need to operate with the classes that have been provided to you. Each class should have the name specified, although the way that these classes interact is up to you. If they need any additional helper classes or methods, you should implement them.

- **`DataLoader.java`** (in package `assignment2020.dataloading`):
  This class will read the participant data in the specified directory and will return a collection of participants. This class should extend `AbstractDataLoader`.
- **`FitnessQuestions.java`** (in package `assignment2020.handoutquestions`):
  This class will provide the answers to "The Questions" and it should extend `AbstractFitnessQuestions`.
- **`GUIPanel.java`** (in package `assignment2020.gui`):
  This class will be the main panel in `GUIFrame` and should extend `AbstractGUIPanel`.
- **`GUIPlotPanel.java`** (in package `assignment2020.gui`):
  This class will be the panel where curves will be plotted and should extend `BasicGUIPlotPanel`.

For convenience, you have already been provided a first empty implementation of these 4 classes.
The deliverable for this assignment is all of the `.java` files needed to run the application, i.e. both the classes you have written, and the classes provided (see separate guidance about "How to submit your code").

All the classes in this assignment will be part of the `assignment2020` package. This package, in addition to the subpackages in `assignment2020.codeprovided.*`, contains the additional three subpackages named: `assignment2020.dataloading`, `assignment2020.handoutquestions`, and `assignment2020.gui`.
You must not change this structure and respect where each of the provided classes have been placed. If you need to create additional classes, you should place them logically within the package structure provided.
If you are using Eclipse, you should see your workspace as follows:



```
▼ COM1003_Assignment
  ▶ JRE System Library [JavaSE-1.8]
  ▼ src
    ▼ assignment2020
      ▶ Assignment.java
    ▼ assignment2020.codeprovided.dataloading
      ▶ AbstractDataLoader.java
      ▶ DataParsingException.java
    ▼ assignment2020.codeprovided.fitnesstracker
      ▶ Participant.java
      ▶ Tracker.java
    ▼ assignment2020.codeprovided.fitnesstracker.ı
      ▶ Distance.java
      ▶ EnergyExpenditure.java
      ▶ HeartRate.java
      ▶ Measurement.java
      ▶ MeasurementFactory.java
      ▶ MeasurementType.java
      ▶ Steps.java
    ▼ assignment2020.codeprovided.gui
      ▶ AbstractGUIPanel.java
      ▶ BasicGUIPlotPanel.java
      ▶ GUIFrame.java
    ▼ assignment2020.codeprovided.handoutquesti
      ▶ AbstractFitnessQuestions.java
    ▼ assignment2020.dataloading
      ▶ DataLoader.java
    ▼ assignment2020.gui
      ▶ GUIPanel.java
      ▶ GUIPlotPanel.java
    ▼ assignment2020.handoutquestions
      ▶ FitnessQuestions.java
  ▶ JUnit 4
  ▶ resources
```

**Make sure that you are using <u>Java 8</u> (aka Java 1.8) and <u>JUnit 4</u>. These will be to execute and assess your code.
If your code fails to compile because it uses features in later versions of Java, your assignment will get 0 marks.**

Minimum set of classes that will need to be implemented.

Suggested location for text data files in dataset

Your package should be `assignment2020` and not `uk.ac.sheffield.com1003.assignment2020` or anything else.

Your code must compile from the command line from the `src` folder with the command `javac assignment2020/Assignment.java` (on MacOS/Linux operating systems) or `javac assignment2020\Assignment.java` (on Windows).
From the same folder, your code will be run from command line as follows:

```
java assignment2020.Assignment resources/data
```

Test this before you submit your code and double-check that your code works.

Coding style

For coding style, readability is of great importance. Read through the Java coding guidelines used by Google, https://google.github.io/styleguide/javaguide.html. Take care with line breaks and whitespace, use comments only when required (i.e. not excessively, but you should have a JavaDoc comment block at the head of each class), use indentation consistently with 4 whitespaces per indent. Before you hand in your code, ask yourself whether the code is understandable by someone marking it. In your design you should aim to have classes that balance cohesion and coupling, so that each class has clear and natural responsibilities within the program.

Your code should adhere to the following style – note that in Eclipse, you can format your code automatically, and a quick internet search will tell you how to do this:

- Indentation 4 spaces (not tabs) per block of text.
- As a minimum you should provide a JavaDoc block and comment block at the head of each class.
- Other comments indented to the same level as surrounding code, with correctly formatted JavaDoc.
- Empty lines and whitespace used to maximize code readability.
- One variable declared per line of code.
- Sensible and descriptive names for methods and variables.
- No empty code blocks or unreachable code, no empty catch blocks.
- No big chunks of commented (unused) code.
- All class names start with Capital letter, i.e. in CamelCase, all method, member, and parameter names in lowerCamelCase.
- Sensible balance of coupling and cohesion in each class (look up cohesion and coupling if you don't know what these terms mean).

How to proceed

This may seem like a complex task and the handout is quite long. The first thing you should do is to examine the classes that are provided. Do not start coding straight away. Think carefully about what each of these classes knows (what are the instance variables?), and what it does (what are the methods?). Then consider the associations between the classes.

Next, think through the logical series of operations that the `Assignment` class should do to work as described in the handout. Which methods and objects are involved at each stage? Try to break each stage down into small chunks. Again, don't do any coding. Write your ideas down on a piece of paper. Design your new classes.
What does each class need to know (what are the instance variables?), and what does it need to do (what are the methods?)? Go back over your design and see if there are any problems.

Now you can start coding. Make sure you write your code incrementally. You are not requested to provide any test classes, but it is a good idea to write small test classes to create objects from the code provided and verify that they work correctly. Remember that you can use a spreadsheet software to verify your answers. Add small sections of code to your new classes and then test to see if they will compile and runs. Write test classes and go back to refine your design if you need to.

Maria-Cruz Villa-Uriol, 16th April 2020