COM1003 Java Programming – Semester 2

Java Programming Assignment: *A Portuguese wine browser*

Submission deadline 3pm on Friday 10 May.

Note, this is an individual assignment. You are <u>not</u> allowed to work in groups and your code must be all <u>your own</u> 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:
<u>https://sites.google.com/sheffield.ac.uk/comughandbook-201819/general-information/assessment/unfair-means</u>

In this assignment, you will build a browser of Portuguese "Vinho Verde" wine samples. 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.

To make sure that this assignment is feasible within the time available, several classes are provided to help you to get started, and **<u>you must use and include these classes in your submission</u>**.

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, do not leave it until the last minute.

General Background

Assessing the quality of wines is critical to the wine industry in countries producing this good.
In this project, you will:

- Build a browser that will allow wine producers of the Portuguese "Vinho Verde" variety to explore the quality of their wine, based on the wine samples previously collected by them.

  Your browser will handle two wine types: red and white. Each wine sample is characterized by a list of wine properties. The wine properties were obtained from a battery of objective tests (e.g. pH values, acidity) and subjective ones based on sensory data (median of at least 3 evaluations made by wine experts). Each expert graded the wine quality with a numerical value between 0 (worst) and 10 (best). More details about the wine properties included in the dataset are provided in the section *'The wine samples data'*.

- Display on the console the answer to a number of questions describing the datasets containing the wine sample data. More details about the questions are provided in the section *'The questions'*.

- Provide the user the ability to systematically explore and evaluate the quality and properties of these wine samples. To do that, a list of queries will be provided to your browser via a text file. The results will be displayed on the console. You can assume that there will be no formatting errors in the provided list of queries. You can find the details describing the syntax of these queries and what they are meant to do in the section *'The queries'*.

- Provide the user the possibility of interacting via a GUI with the wine sample data by using user-defined filters (queries created by the user via GUI). 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'*.

The wine samples data

You can download the two original datasets for the red and white Portuguese "Vinho Verde" varieties from MOLE. Both datasets were collected as part of a research project and made publicly available by the creators in the UCI Machine Learning Repository[1]. The filenames of these two datasets will be provided to your main class (`WineSampleBrowser.java`). The first argument will refer to the file containing the red wine samples and the second argument to the one containing the white wine samples.

Both files are text and use the Comma Separated Values file format (CSV). In this case, the first row provides the twelve categories listed below and the remaining rows list the numerical values for each of the wine samples in the datasets. You are advised to use spreadsheet software to verify that your program is working as desired (e.g. Excel, Google Sheet, Numbers…). In Excel this means using its ability to load CSV files and Data-Filter capabilities. There are many online tutorials showing how to use this last feature (e.g. https://youtu.be/_OdsZR_rL1U).

Below you can find a brief description of the wine properties available for each wine sample (these are the twelve values listed in the first row of the CSV file):

Wine Properties - Objective physicochemical measurements:

1 - fixed acidity
2 - volatile acidity
3 - citric acid
4 - residual sugar
5 - chlorides
6 - free sulfur dioxide
7 - total sulfur dioxide
8 - density
9 - pH
10 - sulphates
11 – alcohol

Wine Properties - Subjective wine quality assessment (based on sensory data):

12 - quality (score between 0 and 10)

You do not need to know much about these and other terms to do this assignment, but if you wish to find out more, check the paper referenced in the UCI Machine Learning repository.

Note that the datasets do not provide a wine sample ID. You do not need to worry about that as the classes that you have been provided already assign an ID to each of the wine samples. Please, study the class `AbstractWineSampleCellar` and its `readWineFile` method.

[1] http://archive.ics.uci.edu/ml/datasets/Wine+Quality

The questions

Your application should be able to answer the following ten questions and display the results in the console:
Q1. How many wine samples are there?
Q2. How many red wine samples are there?
Q3. How many white wine samples are there?
Q4. Which wine samples were graded with the best quality?*
Q5. Which wine samples were graded with the worst quality?*
Q6. Which wine samples have the highest PH?*
Q7. Which wine samples have the lowest PH?*
Q8. What is the highest value of alcohol grade for the whole sample of red wines?
Q9. What is the lowest value of citric acid for the whole sample of white wines?
Q10. What is the average value of alcohol grade for the whole sample of white wines?

*At least for questions Q4-Q7, when you display the answer, you will also need to show the list of wine sample ID/s, their wine properties and wine type (red or white).

Note that for many of the questions above, there might be multiple wine samples providing the answer.

The output on the console could[2] appear as follows to answer questions Q1-Q10. Note that for some of these questions there might be multiple wine sample providing the answer (in the example below this happens for Q4).

```
Q1. Total number of wine samples: 234

Q2. Number of RED wine samples: 120 out of 234

Q3. Number of WHITE wine samples: 114 out of 234

Q4. The best quality wine samples are:
* Wine ID 3 of type WHITE with a quality score of 9.0
* Wine ID 23 of type WHITE with a quality score of 9.0
* Wine ID 45 of type WHITE with a quality score of 9.0

…
```

The queries

Your main class (`WineSampleBrowser.java`) will receive as third argument another filename. That file will contain a list of individual queries of the form:

```
select red where qual > 6
select white or red where qual < 6 and qual > 4 and pH > 2.8
```

In the example above, the first query should produce all red wine samples whose quality is greater than 6.
The second should find white or red wine samples whose quality is between 4 and 6, with a pH above 2.8.

In general, queries will be built as follows:

```
select [red | white | red or white | white or red] where [conditions]
```

where the `[conditions]` can have as many `and` clauses as specified by the user. Note that the `or` clause can only be used to select the type of wine samples (i.e. in the first part of the `select` statement when indicating "red `or` white", but not in the list of query conditions)[3].
For example, to select all red wine samples with fixed acidity above 8, quality below 5, and pH between 2.7 and 2.79, one would use a slightly more complex query such as:

```
select red where f_acid > 8 and qual < 5 and pH > 2.7 and pH < 2.79
```

[2] This is only indicative and the results shown do not correspond to the datasets you have been provided.
[3] This is meant to simplify your programming task.

The `[conditions]` in your queries will be your query conditions and they should support the following comparison operators:

Operator	Meaning
>	Larger than
>=	Larger than or equal to
=	Equal to
!=	Not equal to
<	Less than
<=	Less than or equal to

The notation used to represent the different wine properties in the queries is described in the following table:

Wine Property	Query keyword
Fixed acidity	`f_acid`
Volatile acidity	`v_acid`
Citric acid	`c_acid`
Residual sugar	`r_sugar`
Chlorides	`chlorid`
Free sulfur dioxide	`f_sulf`
Total sulfur dioxide	`t_sulf`
Density	`dens`
pH	`pH`
Sulphates	`sulph`
Alcohol	`alc`
Quality	`qual`

In most cases your queries will return multiple wine samples, and you should provide **self-informative and easy-to-read outputs to the console** to each of the queries. For example[4]:

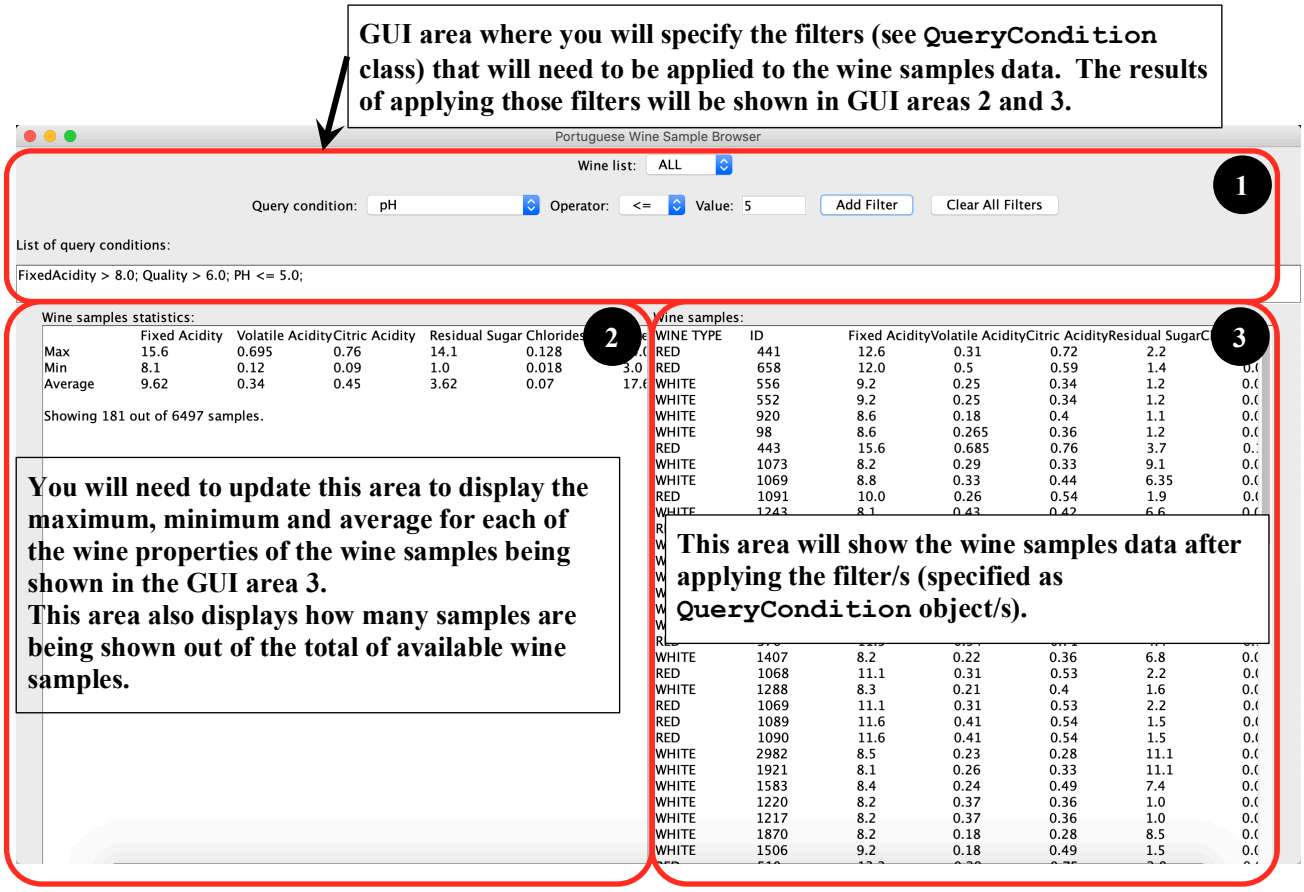
```
****** Query #1 ******
* select red where f_acid > 8 and qual < 5 and pH > 2.7 and pH < 2.79 *
In total, 3 red wine samples match your query
The list of red wine samples are:
[Red wine, sample #16, f_acid: 8.3, v_acid: 0.74, c_acid: 0.08,
r_sugar: 1.7, chlorid: 0.094, f_sulf: 10, t_sulf: 45, dens: 0.99576,
pH: 2.77, sulph: 0.5, alc: 9.8, qual: 4]
[Red wine, sample #45, f_acid: …]
[Red wine, sample #120, f_acid: …]
**********************
```

[4] Note that this example is only for demonstration purposes. This query will not produce the results being shown with the datasets you have been provided.

The GUI

Your next figure shows how the GUI could look like. You have been provided with a basic implementation of the main panel showing the three areas highlighted below.

- *Area 1 (top):* Allows the user to specify the filters that will need to be applied to the wine samples data.
- *Area 2 (bottom-left):* Shows basic statistics describing the resulting dataset after applying the filters listed in Area 1 (List of query conditions).
- *Area 3 (bottom-right):* Shows the subset of wine samples and their wine properties after applying the filters listed in Area 1 (List of query conditions).



Areas 2 and 3 will be updated whenever the user changes either the Wine list (red, white or all), or clicks the buttons Add filter or Clear All Filters.

In the example shown in the figure above, the user specified three filters, which were translated into three `QueryCondition` objects representing the conditions "Fixed Acidity > 8.0", "Quality > 6.0" and "pH <= 5.0", respectively. Every time the user clicked the button Add Filter, a new filter was added to the List of query conditions being shown in the GUI area 1. If the user clicks the button Clear All Filters, the List of query conditions needs to be cleared and no filters should be applied to the dataset being shown in areas 2 and 3. This means that the areas 2 and 3 would need to be updated to display the complete (unfiltered) original dataset for the wine type selected with the combobox named "Wine list".

Programming task

The overall aim of the assignment is to produce a Java program that allows browsing wine samples and exploring their wine properties. The user will be able to interact with the browser to indicate which wine samples 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 should work as follows.

There must be a `WineSampleBrowser` class (i.e. a file called `WineSampleBrowser.java`) with a `main` method. This will be the main class, and when invoked, the main method should do the following:

- Read the command line arguments to obtain the names of the three input files required by **`WineSampleBrowser`** (the note in the figure indicates in which order the three filenames need to be provided). You might use the following piece of code in your main:

```
public static void main(String[] args) {

    if (args.length == 0) {
        args = new String[] {
                "resources/winequality-red.csv",
                "resources/winequality-white.csv",
                "resources/queries.txt"};
    }

    // Instance variables
    String redWineFile = args[0];
    String whiteWineFile = args[1];
    String queriesFile = args[2];

    WineSampleCellar cellar = new WineSampleCellar(redWineFile, whiteWineFile, queriesFile);
```

Please, keep this bit of code AND respect the order in which the three filenames are specified:
- **First argument:**
  Red wine samples dataset (a CSV file)
- **Second argument:**
  White wine samples dataset (a CSV file)
- **Third argument:**
  List of queries as a text file (a TXT file)

Failing to comply with this specification will mean that your program will not provide the expected results and that the code will be returned to you.

- Construct a new **`WineSampleCellar`** object that will get as arguments the three input filenames provided to `WineSampleBrowser` (see the figure above). These files will contain the data that this class will be analysing. Note that you have been provided with an `AbstractWineSampleCellar` class. Your `WineSampleCellar` class will need to extend `AbstractWineSampleCellar` and provide an implementation for the methods defined as abstract. `AbstractWineSampleCellar` will be already doing for you the file reading of wine sample data files and queries file. You will not need to worry about that. You need to study and understand well `AbstractWineSampleCellar` to be able to use the collections filled in by the `readWineFile` and `readQueryFile` methods. Understanding them will help you writing `WineSampleCellar`.
- Construct the GUI. You will need to add to the GUI a panel. The panel will be the class **`WineSampleBrowserPanel`**. Note that you have been provided with a class called `AbstractWineSampleBrowserPanel`. Your `WineSampleBrowserPanel` will need to extend `AbstractWineSampleBrowserPanel`. `AbstractWineSampleBrowserPanel` is an abstract class and already provides a basic implementation of the main elements in the GUI shown in the section *"The GUI"*. Based on the interaction of the user with the GUI, the GUI will update the contents being displayed. Please, study and understand well the class `AbstractWineSampleBrowserPanel`. The class is full of comments to facilitate your work.
- Display on the console the answer to the questions Q1-Q10 specified in the section *"The questions"*. The class `AbstractWineSampleCellar` provides a number of methods that will be helpful to answer these questions. You will need to implement them in `WineSampleCellar`.
- Display on the console the results of running the list of queries provided as the third argument to your `WineSampleBrowser`. Details about how this functionality should operate are described in the section *"The queries"*. Note that the abstract class `AbstractWineSampleCellar` provides an abstract method named `displayQueryResults` that will be helpful to provide this functionality.

Classes provided

Several classes are provided on MOLE. Each is part of the `assignment2019.codeprovided` package. ***You must use these classes and you must <u>not</u> modify them***. The classes provided are these:

- `WineProperty.java` is a helper enum that provides a list of helpful constants describing the wine properties of a wine sample.
- `WineType.java` is a helper enum that provides a list of helpful constants describing a wine type.
- `WineSample.java` contains all the necessary member variables and methods to store a wine sample. This means that one row in the original CSV files provided will become a `WineSample` object in this project.
- `AbstractWineSampleCellar.java` is an abstract class that provides implementations of file reading utilities for the input datasets containing the wine samples and the query text file. The class stores in a `Map` all the wine samples read from the CSV files (variable `wineSampleRacks`). Please study well how to use the `Map` collection and what each of the abstract methods provided are meant to do. These abstract methods are the ones that your **`WineSampleCellar`** will need to override to provide answers to provide the functionalities described in this handout.
- `AbstractWineSampleBrowserPanel.java` is an abstract class that provides the basic code you will need to build a GUI similar to the one shown in the figure below. It defines a list of methods that your **`WineSampleBrowserPanel`** class will need to override to provide the functionalities described in this handout.
- `Query.java` is a class that represents one query. There will be one `Query` object for each of the queries contained in the text file with a list of queries provided to you. Please, study well this class and how it relates to the `QueryCondition` class described next.
- `QueryCondition.java` is a class that specifies one of the conditions that are part of a `Query` object. The GUI (in particular the class `WineSampleBrowserPanel`) will create objects of this type to represent the user-specified filters (using the GUI Area 1).

Deliverables

In this project, you must fully provide an implementation of the following classes. They will need to operate with the classes that have been provided to you. Each class should have the name specified in bold, although the way that these classes interact is up to you.

- **`WineSampleBrowser.java`** (your main class)
- **`WineSampleCellar.java`** (will need to extend `AbstractWineSampleCellar`)
- **`WineSampleBrowserPanel.java`** (will need to extend `AbstractWineSampleBrowserPanel`)

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. These should be uploaded to MOLE as described below.

Each of these classes provided are part of the `assignment2019.codeprovided` package. You must not change this, and you should place all of the other classes you write into a package called `assignment2019` (i.e. the first line of each of your classes should be `package assignment2019;`). If you are using Eclipse, then you should create a new package in your workspace called `assignment2019`. You can then import these files in the same way you imported the module code. ***Note that your package should be simply `assignment2019` and not `uk.ac.sheffield.com1003.assignment2019` or anything else.***

Your code must compile from the command line from the `src` folder with the command `javac assignment2019/WineSampleBrowser.java` (on MacOS/Linux operating systems) or `javac assignment2019\WineSampleBrowser.java` (on Windows). Test this before you submit your code.

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 `WineSampleBrowser` 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.

Submission and deadline

There will be an upload tool provided on MOLE, and the ***deadline for submission is 3 pm on Friday 10 May.*** You should upload your java source code as a compressed zip archive to MOLE before the submission deadline. ***You must upload a .zip file and not any other form of compressed file. For example, .tgz, and .rar files will not be accepted; these will be returned to you.***

The zip file should be named according to your CICS username, e.g. if your user name is aca18xxx then your file should be named aca18xxx.zip. The zip file should uncompress into a folder called 'assignment2019' containing your source code, with a sub-folder called 'assignment2019/codeprovided' (or 'assignment2019\codeprovided' on Windows) containing the provided classes. To create a zip file on Windows, navigate to your Eclipse workspace, then into the 'src' folder. Right click the folder 'assignment2019' and select 'Send To...' then select 'Compressed (zipped) Folder'.

Submission checklist

It is imperative that you adhere precisely to the required program structure outlined in this assignment. If you do not, your code may not be compatible with the marking system and will be returned to you.

- Embed your main method in a class called `WineSampleBrowser.java`.
- Ensure that your classes are part of the `assignment2019` package, and import the classes provided for you from the `assignment2019.codeprovided` package.
- Ensure that your submission has the following classes in the `assignment2019` package, with exact matches to filenames:
    - `WineSampleBrowser.java`
    - `WineSampleBrowserPanel.java`
    - `WineSampleCellar.java`
- Also ensure that your submission has the following unmodified classes in the `assignment2019.codeprovided` package:
    - `AbstractWineSampleBrowserPanel.java`
    - `AbstractWineSampleCellar.java`
    - `Query.java`
    - `QueryCondition.java`
    - `WineProperty.java`
    - `WineSample.java`
    - `WineType.java`

Late work

Late work will be penalised according to standard University procedure (a penalty of 5% per day late; work will be awarded a mark of zero if more than 5 days late). Be aware that sometimes MOLE is slow or unavailable, so make sure you upload your code well in advance of the deadline of 3pm, Friday 10th May.

Marking and feedback

Your submission will be marked out of 55, and marks will be awarded as follows:

Program operates correctly (10 marks) criteria include: providing the correct answers to Q1-Q10 (section *"The questions"*), providing the correct outputs to the queries (loaded from the text query file)(section *"The queries"*), and GUI operating as expected providing the correct answers (section *"The GUI"*).

Use of Java features (10 marks) criteria include; inheritance (e.g. for the main classes extending the abstract classes provided), use of Java collections methods to handle collections correctly, use of Java Swing and event handling, and choice of programming structures for loops and conditionals.

Good programming style (15 marks) criteria include; economy of code (*i.e.* methods are of reasonable length), class cohesion (*i.e.* classes have a clear set of responsibilities), readability (*i.e.* no long lines of code), sensible choice for class and variable names, and use of comments in each class.

Implementation and correct operation of `WineSampleBrowser` *and* `WineSampleCellar` *classes (5 marks for each)* criteria include; correct operations, quality and elegance of implementation.

Implementation of graphical display of `WineSampleBrowserPanel` *(10 marks)* criteria include; display updates correctly after the specification of filters, event handling is adequate, it makes good use of Java API components, and GUI controls work cleanly.

Questions

If you have any questions about the assignment, please, first check the list of Frequently Asked Questions in the Assignment section. If your question is not answered there, please, use the COM1003 forum to post your questions. The lecturer and the demonstrators will be available to answer questions about the assignment in the Tuesday and Friday labs of weeks 10 and 11.

Maria-Cruz Villa-Uriol, 6th April 2019