Assignment OO Programming Sem 2 2019
Java

The aim of this assignment is to develop a java application that puts into practice the programming skills taught on your OO programming course.

You have three different options for your project:

- You can pick one of the ideas described in this document. You should read them anyway to give you an idea of the type and scale of assignment expected.
- You can create your own idea e.g. a game, some sort of analysis tool, a solution to a puzzle, etc.. your choice – but you need to get your idea approved first, to check it's suitable.
- You can do a team project, with up to 3 people, and pick your own idea as a group. This will need to be a bigger project, with each team member doing an identifiable, equal amount of the development.

This assignment is worth 60% of your Sem 2 CA (and therefore, 30% of your OVERALL CA mark.

What next?

If you are doing one of the 5 ideas below - just go ahead and get started.

If you are doing your OWN project idea, or a group idea, please submit a description of your idea to me by Friday March 15thth. Your idea must have some sort of algorithm in it – just as the sample projects shown here do – so keep that in mind. You should divide it into the core functionality, and the optional advanced features. I will approve the project or send it for redrafting if it is not suitable. The sooner you send it to me, the sooner you get it returned.

Assignment criteria

- All or most of your assignment should be written in Java. If there are extra parts (e.g. database) that is fine.
- Your code should demonstrate the use of OO concepts. - including using classes for separate entities (as opposed to dumping everything into a single class), methods, encapsulated attributes, constructors, inheritance, interfaces, polymorphism.
- Code should follow java naming standards, be well indented, bracket aligned , comment headers, comments.
- You need to include a link to a short video with your submission to demo it, explain how you have coded it, what classes you have used and so on, how your algorithm worked . *No more than 2 minutes.* (It should be of your actual screens with your voice - as opposed to you!).
- Include a readme file to briefly explain your project.
- Use BitBucket or GIT to manage your source code. Your code and video link will be collected from your whatever code repository you use (more on this). If you haven't used GIT, we can cover in class.

Assignment OO Programming Sem 2 2019
Java

- If you make use of any code directly from a book or online source, **you must show this in the comments.** You will be marked on code that is your own code.
- Your will demo your code in the lab after Easter as part of the marking.
- The assignment is due in on by end of day **Friday April 12th**

Marking

The marking scheme will be:

- 10% Project management (demonstrating regular commits using GIT over the lifetime of the assignment)
- 10% video explaining your assignment – available via YouTube or any video hosting service
- 40% Basic core functionality: for well functioning, well implemented, using code that follows coding standard *core* functionality – i.e. the core of whatever problem you are solving.
- 40% Optional advanced features: for added functionality that enhances whatever your idea is.

Plagiarism means attempting to pass off someone else's work
as your own or deliberately allowing another student to copy your work

Be careful: PLAGIARISM WILL NOT BE TOLERATED AND WILL BE DEALT WITH SEVERELY

If you are not picking your own idea, the following ideas are available to choose from if you wish..

1. Topic Modeller

This tool will allow you to detect whether a set of documents are "about" the same topic or not.

The tool will analyse the words in each document – and decide what the most common words are in each document.

"Stop" words should be excluded from the analysis (this is the list of words that don't really add meaning which are not included in the similarity measure – such as "the", "a", "of" "he".. etc. Sets of stop words can be found online (for example at : http://www.lextek.com/manuals/onix/stopwords1.html)

Basic on the overlap in the top X words (e.g. 10), a grade of likelihood of it being about the same topic can be produced (e.g. 70% overlap in the top ten words = 70% likely to be about the same topic

or whatever way you decide to do this). Likewise, documents with small overlap on most comment words (e.g. < 40%) are definitely different topics.

Additional features that could be included to include marks would be things such as:

- Having a GUI to run things and display the results.
- Having the ability to select documents through a "File Chooser" GUI, with graphical results rather than just console
- Allowing words which appear on the list as common across documents - but which are obviously not informative about the topic – to be added to the stop word list, and the analysis re-run.
- Displaying the results - so that the user can easily see how many, and what topics (i.e. what word groups) are returned.
- Saving down the results persistently to a file, with the overlapping words written out too.
- Extra features that you think will enhance the application.

2. Data Explorer

Note: I will cover SQL database connection (briefly) in class.

Data analysis is a major area within Computer Science . Apart from the Big Data generated by social media and internet generally – there is an insatiable desire on behalf of companies to analyse their data in order to reveal "knowledge" hidden in the data. For example, an Optical chain analysed their sales data to determine that sales of high end high profit glasses peaked on Friday afternoons – so they made sure that they had enough staff to service this demand.

The Irish Government have put 1000s of datasets into public use at a portal site:
https://data.gov.ie/data

This has data about a whole plethora of public interests and government control information e.g. about crime rates, hospitals, schools, transport, environment, energy use and so on.

The purpose of this project is to take **ONE** of these datasets – and build a tool that shows interesting facts from the dataset. The dataset formats include Comma Separated (CSV) which is probably the easier to work at – so these datasets are at : https://data.gov.ie/data/search?res_format=CSV

You don't have to use the full dataset if it is too big. But to query it, you will need to LOAD the dataset, or a subset of it, into a relational database yourself – and get a connection working between your java code and the database (using JDBC).

You could also just read in the file and do simple operations on it (e.g. how many of ??).. but searching is very limited if you stick just to file format without a database.

Your project will need to have a GUI that allows query parameters to be put in.

Extras

- Ability to see the results through the GUI too.
- Flexible queries – not just one or two hard coded

Whatever else you decide might enhance the application.

3. LanguageAnalyser

This tool will analyse text samples (e.g. a file of user posts, individual sentences) – and analyse it to decide whether it is

- Slang- style language: Think yourself of how this could be identified. e.g. Typical style is shorter sentences, shorter average word length, containing slang words or bad language?
- Formal business language: Longer sentences, average longer word length, "clean" language, good punctuation.

Ideas for additional features that could be included to include marks would be things such as:

- Have an editable dictionary list so that slang words can be extended.
- Having a GUI to run things and display the results in a suitable way.
- Having the ability to select documents through a "File Chooser" GUI, with graphical results rather than just uploading a file or sentences from the console.
- Extending the idea of how to identify slang versus formal language.
- Making it multi lingual.

Make sure you can demo this properly .

4. My Search Engine

This tool will allow you to search for a term across a set of text sources – e.g. groups of text files.

The user puts in a search term and the tool will check the contents of a set of text files and tell you which ones contain the search term. The files that have the "strongest match" against the search term should be returned at the top of the list.

The user should be able to search on a single word. But to make the search better, you should be able to search on multiple words – e.g. "Christmas day".. although what rules you apply as to whether these are assumed to be together or separate words is up to you; Maybe you can use "*" and logic (e.g. this word AND this word OR this word) to make the search smarter.

To make this more advanced, you will need to:

- Have a GUI to take the search request parameters and the search results.
- Have more sophisticated searching - e.g. exact phrase matches, comma separate words, wild cards (such as walk* to find walked, walking, walk etc).
- Have a way for the user to pick the search space (i.e. the text files to be searched).
- Have a ranking mechanism so the strongest match is returned first – and a ranking metric (e.g. a %) is calculated and the user can see this.

- Spelling correction where it can correct wrong spelling of search terms
- What else can you come up with?

5. Machine learning model, using Naïve Bayes

This assignment allows you to learn about machine learning, including building a predictive/classification model.

Background: Machine learning is a huge component of Artificial Intelligence. It allows computers to perform tasks automatically, using algorithms to spot patterns (very similar to how humans do). A typical use of machine learning is to build classification models – which classify or predict what an input is about. For example, machine learning models can be created that are "trained" to a recognise a fraudulent insurance claim. This is done often done by looking at past examples insurance claims that were fraudulent and claims that weren't fraudulent – and spotting the patterns associated with each. This is an example of "supervised learning" - where you allow a computer to learn from examples – and it's very similar to how we learn as humans. We "train" a machine learning algorithm with examples (e.g. 100 "good" insurance claims, and 100 "fraudulent" claims). The algorithm figures out the patterns – and is able to then classify or predict whether a new claim is fraudulent or not, having sussed out the pattern.

This assignment will use a popular machine learning algorithm called Naïve Bayes – to be explained in class. A simple dataset is provided below showing previous cases of whether patients had tonsillitis or not (based on assessing temperature, aches and pains and sore throat). Using java, the assignment should "train" a model using the dataset provided. Don't hard code the classifier in anyway – make sure it is driven by the data dynamically in your code. Using a GUI, allow for patient symptoms to be put entered in, and the prediction model will give the probability of whether they have tonsillitis or not.

To make this work well , you will need to have a GUI to take the new input (patient systems) that you want to test whether likely leading to tonsillitis or not.

- Have the application read in the training set and do the training dynamically (i.e. don't hard code the rules for the classifier).
- If extra data rows, termed "instances" are added to the dataset, your classifier should still work (in fast, improved as a result of new data).
- Allow for model to train on some of the dataset (e.g. 70%) and test itself on the remainder of the dataset(30%) so that it knows how "good" it is (i.e. be able to evaluate itself).

Previous cases (i.e. "training" data):
DataSet:

Temperature	**Aches**	**Sore throat**	**Tonsillitis**
hot	Yes	No	Yes

hot	Yes	Yes	No
hot	No	No	No
normal	No	Yes	Yes
cool	Yes	Yes	no
cool	Yes	No	no
cool	No	No	yes
normal	No	Yes	yes
cool	No	No	no
normal	Yes	Yes	yes
normal	Yes	Yes	yes
normal	No	Yes	yes
hot	Yes	No	no
normal	Yes	No	no
hot	Yes	Yes	no
normal	No	Yes	yes
cool	Yes	No	no
normal	Yes	Yes	yes