[Please read all 5 pages of this document first]

COSC1519 Introduction To Programming

Assignment 1: Chatterbot

[spec 2018.03.09]

Submission and due date: See Canvas→Assignments→Assignment 1

Note: This is an individual assignment. All submitted work must be your own and you must not give any part of your work to others (see RMIT Academic Integrity information). When in doubt, ask Daryl or Gayan. **You are awarded marks based on your ability to follow and fulfill the given requirements** of this assignment. **Every word in this document matters.** If it's not written, it's not a requirement but you must use the necessary concepts as shown in class materials as the list of things you must not do is too long to list. If there are changes to any requirements or additional clarifications, announcements will be made via *Canvas→Announcements*.

Tip: Books and lecture notes do not contain solutions to programming tasks. Instead, programmers learn concepts and apply them to solve unforeseen programming tasks. In this assignment, you will be demonstrating your understanding of the concepts covered in weeks 1-4 of Intro To Programming (see *Canvas→Syllabus)*. There is no need to wait to get started as you can attempt some parts already. Students are expected to start on assignments upon their release and are expected to manage time and minimise risks that can prevent you in completing the work. As this is tertiary education, you are also expected to independently investigate additional examples and perform R&D that is required to complete this assignment.

Abstract

In this assignment, you will create a simplified *chatbot*/*chatterbot* (tip: look at examples of chatter bots like ELIZA) using Java. In the simplest sense, a chatterbot is a program that can imitate a conversation with a human by checking the human/user input to see if there are any known keywords/topics. Like with most programming tasks, there are many ways to solve a problem but you need to ensure that you are fulfilling the requirements of this document.

Depending on how many marks you want to go for, you can either go for the 'limited functionality' level (up to 5 marks maximum) or the 'normal functionality' level (up to 10 marks maximum).

You will need to demonstrate your code during demonstrations in practical classes and submit a report.

Hints and tips will be given during the classes and the class work will help you complete this assignment. You are expected to do the class work and the assignment simultaneously.

Full details in the rest of this document.

Development Levels

You can choose and go for either level below but only the 'normal functionality' will make you eligible for the full 10 marks of this assignment. For either level, you will need fulfill the code requirements and the report requirements for your submission to be valid. Your assignment will be invalid if you do not complete any of these steps (e.g. submitting just the code will not result in any marks). If you wish to organise special consideration, please contact gayan.wijesinghe@rmit.edu.au at least 2 working days before any due dates/deadlines.

1. Limited Functionality Level (5 marks max)

There are three components (code, report and demo) to this functionality level. You must do them all. Details below.

1.1. Limited functionality code requirements (2.5 marks when completed with rest of the steps):

Your program must follow the algorithm below:

a. Create separate variables for at least 10 built-in keywords/topics (hard-coded in your program). E.g. a keyword can be "work". Arrays are not needed for this level (code will be lengthy – aim to do normal functionality instead!)
b. Ask the user to say something about a known topic (must show them the built-in keywords/topics in the input prompt).
c. If a known keyword/topic is in the entered user input, the computer must ask the user about that topic. E.g. if the user entered "i want to talk about work and stuff" and if "work" is the first matching keyword, the computer must ask them "Tell me more about work...". (Note: recognised keyword/topic must be in the prompt when asking). Hint: Avoid deep conversations.
d. If the (most recently entered) user input contains a question mark anywhere, you must tell them that you (the computer) will be asking the questions. (Hint: This must be done even if a keyword is not recognised).
e. If the (most recently entered) input contains no known keyword/topic, you must repeat from *b.*
f. Repeat from *c.* until user cancels.

Restrictions: while-loops must be used for repetition. Must not use *return*, *break* or *continue* statements.

Hint: Before writing your code, for ease of understanding, draw a state diagram, where some of the steps a..f in the algorithm above are the states (written inside circles) and user inputs/selections (written on the lines) will cause transitions to other states.

To submit the code, follow the instructions under *Canvas→Assignments→Assignment 1.*

1.2 Limited functionality report requirements (2.5 marks when completed with rest of the steps):

The goal of the report is to explain how to make your chatterbot program to a new Intro To Programming student. I.e. assume the reader only knows how to code and run a hello-world program in Java. The **report must not contain your program** (marks will be deducted if it does). Use a word processor (e.g. OpenOffice Writer, Microsoft Word, etc.) to write a professional, technical report but the final submission format of the report must be plain-text and should be presentable if opened using Windows Notepad.

Your report must be titled 'How to create a chatterbot that uses a fixed set of keywords' and it should have the following major sections/headings in which you **explain things in your own words** (no part of the report requires you to quote any other work):

Section 1: Concept of a chatterbot

Explain your understanding of a chatterbot, what it is and how it is different to programs that have similar features.

Section 2: Programming concepts required and why

Explain how you have used each of the following concepts in your program when meeting the code requirements (create subheadings under section 2 for each): Variables, user inputs, if-statements (and input vs. topic matching), while-loops.

Section 3: Issues/suggested improvements

Explain any existing issues (when meeting requirements) and improvements.

Marks distribution of report: section 1 (0.5 marks), section 2 (3 marks), section 3 (0.5 marks). Show your work-in-progress report in demo 1 to find out if you're on the right track with format/style, level of detail, etc.

To submit the report, follow the instructions under *Canvas→Assignments→Assignment 1.*

1.3 Limited functionality demo requirements (must do both demos to get marks for other steps):

The demos are conducted during the practical sessions and the mark of the practical session comes from the demo (see Canvas→Syllabus for details). Demo requirements are as follows:

1. Before every demo, you must make a submission of your code and report.
2. You are required to demonstrate and answer questions on the submitted version of your work to your instructor when they come over to you during the practical session to which you are allocated. (If you are attending a class to which you are not allocated, priority will be given to students who are allocated to that class. Demos will not be taken outside of class times and demo 1 must be done in week 4 and demo 2 must be done in week 6. If you do not have pre-approved special consideration and if you miss the demo, your submission will be invalid and will not be marked.
3. Demo 1/'early feedback demo' requirements: Code must compile under Eclipse without red dots/errors (yellow dots/Eclipse warnings ok). Must show use of some variables, some user inputs, and minimal use/experimentation with if-statements. It is good idea to show work-in-progress report to get feedback on format/style and level of detail.
4. Demo 2/'final demo requirements: Every code requirement for the 'limited functionality' level must be implemented. Code must compile under Eclipse without red dots/errors or warnings (yellow dots).

Code must be formatted consistently, descriptive variable names must be used and complex parts of the code must be explained using comments.

2. Normal Functionality Level (10 marks max)

There are three components (code, report and demo) to this functionality level. You must do them all. Details below.

2.1 Normal functionality code requirements (5 marks when completed with rest of the steps):

Implement the 'limited functionality' code requirements first and extend it so that the keywords are stored in an array. When run, the program must ask the user how many keywords they want and then take inputs for each, one at a time and store them in the array. Your code must work with 1 or more keywords. User inputs must be validated then and there in a justifiable way and user must be asked to re-enter until they get the inputs correct (e.g. you can assume that a user will enter an integer for the number of keywords, don't restrict the maximum number of keywords but there must be a lower limit).

To submit the code, follow the instructions under *Canvas→Assignments→Assignment 1.*

2.2 Normal functionality report requirements (5 marks when completed with rest of the steps):

The goal of the report is to explain how to make your chatterbot program to a new Intro To Programming student. I.e. assume the reader only knows how to code and run a hello-world program in Java. The **report must not contain your program** (marks will be deducted if it does). Use a word processor (e.g. OpenOffice Writer, Microsoft Word, etc.) to write a professional, technical report but the final submission format of the report must be plain-text and should be presentable if opened using Windows Notepad.

Your report must be titled 'How to create a chatterbot that uses an array for keywords' and it should have the following major sections/headings in which you **explain things in your own words** (no part of the report requires you to quote any other work):

Section 1: Concept of a chatterbot

Explain your understanding of a chatterbot, what it is and how it is different to programs that have similar features.

Section 2: Programming concepts required and why

Explain how you have used each of the following concepts in your program when meeting the code requirements (create subheadings under section 2 for each): Variables, user inputs (and validation), if- statements (and input vs. topic matching), while-loops, arrays.

Section 3: Issues/suggested improvements

Explain any existing issues (when meeting requirements) and improvements.

Marks distribution of report: section 1 (0.5 marks), section 2 (7 marks), section 3 (0.5 marks). Show your work-in-progress report in demo 1 to find out if you're on the right track with format/style, level of detail, etc.

To submit the report, follow the instructions under *Canvas→Assignments→Assignment 1.*

2.3 Normal functionality demo requirements (must do both demos to get marks for other steps):

The demos are conducted during the practical sessions and the mark of the practical session comes from the demo (see Canvas→Syllabus for details). However, without completing the demos, your assignment attempt will not be valid. Demo requirements are as follows:

1. Before every demo, you must make a submission of your code and report.
2. You are required to demonstrate and answer questions based on the submitted version of your work to your instructor when they come over to you during the practical session to which you are allocated. (If you are attending a class to which you are not allocated, priority will be given to students who are allocated to that class. Demos will not be taken outside of class times and demo 1 must be done in week 4 and demo 2 must be done in week 6. If you do not have pre-approved special consideration and if you miss the demo, your submission will be invalid and will not be marked.
3. Demo 1/'early feedback demo' requirements: Code must compile under Eclipse without red dots/errors (yellow dots/Eclipse warnings ok). Must show use of some variables, some user inputs, and minimal use/experimentation with if-statements. It is a good idea to show work-in-progress report to get feedback on format/style and level of detail.
4. Demo 2/'final demo requirements: Every code requirement for the 'normal functionality' level must be implemented. Code must compile under Eclipse without red dots/errors or warnings (yellow dots). Code must be formatted consistently, descriptive variable names must be used and complex parts of the code must be explained using comments.

End of Document