




COSC2452 Introduction To Programming (OUA)

Assignment 2 (v.2019.09.16)

	Assessment Type: Individual assignment; no group or collaborative work. Submit online via Canvas→Assignments→Assignment 2. Marks awarded for meeting requirements as closely as possible. Clarifications/updates may be made via announcements /relevant discussion forums .
	Due date: 6:30pm, 14/Oct/2019; Deadlines will not be advanced but they may be extended. Please check Canvas→Assignments→Assignment 2 for the most up to date information. As this is a major assignment in which you demonstrate your understanding, a university standard late penalty of 10% per each working day applies for up to 5 working days late (submissions will not be accepted afterwards), unless special consideration has been granted. All extensions must be applied for directly via RMIT Special Consideration .
	Weighting: 8 marks (excluding bonus marks)

1. Overview

There is no book containing the music to every song that will be written. There is no book containing the answers to every mathematical calculation that we will need to perform. Similarly, there is no book, set of lecture slides, video, etc. that will give a programmer (you) the solutions to every programming problem. A programmer is able to take fundamental programming concepts and, with the experience they have gained from analysis, evaluation and problem solving, put them together to solve new problems.

For this assignment, assume that you are a freelance programmer creating a small tool or a utility program of your own choosing to add to your portfolio of simple Java applications. With this project you aim to demonstrate to potential employers or clients how you can:

1. Create a small tool or utility program using (exclusively) a limited set of fundamental code concepts (detailed in section 4.1 below) and,
2. You are able to analyse and evaluate in your code documentation (see requirements in section 4.2 below).

Note: You must not just “throw in the concepts” to your program just because they need to be there; it should be clear from the code why a certain concept should be there and you must further explain these through your comments. You will also need debug your code on your own and document any issues, etc. You are given marks on your ability to fulfill all requirements of this document.

There are code requirements (6 marks) and documentation requirements (2 marks) for a total of 8 marks.

Develop this assignment in an iterative fashion (as opposed to completing it in one sitting). You can and should get started now as there are concepts from the week 1 lessons that you can incorporate from now itself.

If there are questions, you must ask via the relevant Canvas discussion forums in a general manner (replicate your problem in a different context in isolation before posting).

2. Assessment Criteria

This assessment will determine your ability to:

1. Follow coding, convention and behavioral requirements provided in this document and in the lessons.
2. Independently solve a problem by using programming concepts taught over the first several weeks of the course.
3. Write and debug Java code independently.
4. Document code.
5. Ability to provide references where due.
6. Meeting deadlines.
7. Seeking clarification from your “supervisor” (instructor) when needed via discussion forums.
8. Create a program by recalling concepts taught in class, understanding and applying concepts relevant to solution, analysing components of the problem, evaluating different approaches.

3. Learning Outcomes

This assessment is relevant to the following Learning Outcomes:

1. Demonstrate knowledge of basic concepts, syntax and control structures in programming
2. Devise solutions to simple computing problems under specific requirements
3. Encode the devised solutions into computer programs and test the programs on a computer
4. Demonstrate understanding of standard coding conventions and ethical considerations in programming.

4. Assessment details

Note: Please ensure that you have read sections 1-3 of this document before going further.

Your code must meet the following code requirements (section 4.1) and documentation requirements (section 4.2).

In places where this specification may not tell you how exactly you should implement a certain feature, the programmer (you) need to use your judgment to choose and apply the most appropriate concepts from class materials. Follow answers given by your “client” or “supervisor” (your coordinating instructor) under Canvas→Discussions→‘Assignment 2’ when in doubt.

4.1) The following code requirements/concepts must be applied to demonstrate your knowledge of lesson materials and approaches; must refer to corresponding rows in the rubric (section 9). [1+1+1+1+1+1=6 marks]

C1. Code presentation and format – The program must be entirely in one Java class (one .java file). The name of the class (file) must be chosen to match the application (default names such as AssignmentX.java) will not be accepted. Code must be formatted consistently (Tip: use Eclipse→Source menu→Format before every submission). Must not include any unused/irrelevant code (even inside comments); what is submitted must be considered the final product.

C2. Data types and user inputs - Must use Scanner or JOptionPane. The program must take at least one numerical input and store in a variable of a numerical data type. The program must also take at least one String input and store in a suitable variable.

C3. else if statements – Must not have redundant conditions.

C4. while loops to repeat; no for, do..while, recursion, or any other form of repetition. while-loop condition must fail eventually. Must not use break, continue, return, System.exit or similar branching for terminating loops.

C5. Accessing and modifying values in (standard) arrays using while loops. No ArrayList or similar data structures. Students must demonstrate their ability to manipulate (standard) Java arrays on their own without using external classes/libraries.

C6. Create methods in addition to the ‘main’ method. Must not use static anywhere in the program except in the main method.

4.2) Documentation requirements. Must refer to corresponding rows in the rubric (section 9). [1.5+0.5=2 marks]

D1. The code comments in this assignment must focus more on the analysis of your approaches and their evaluation instead of simply translating the Java code to English (see rubric in section 9 of this document; examples will also be given in lessons). Write comments before code blocks (e.g. before methods, loops, ifs, etc. where you can have { }) and important variable declarations.

D2. Explain any code requirements that you have not met and all bugs (situations that might cause the program to crash or behave abnormally) in the approximate locations they originate within your code. Bugs imposed by limitations in the lesson topics such as input type mismatches need not be corrected in code but they still must be documented, if they exist. If you do not have bugs, you must explicitly say that there are none. Tip: A good programmer knows the limits of their program. If doing B2, explain any standard code requirements that you may have violated due to the bonus requirements.

4.3) Bonus requirements

You can attempt either B1, B2 or both but to obtain any bonus marks, you must meet all requirements of the non-bonus/standard requirements. The total mark for assignments+weekly work (the non-exam component) is capped at 50 marks; the exam will be 50 marks. For a full break-down, see Canvas→Home→Syllabus.

B1. Submit your final version of the assignment, 1 week before the deadline for +0.5 bonus marks or 2 weeks before the deadline for +1 bonus mark or 3 weeks before the deadline for +1.5 bonus marks.

B2. Incorporate file reading and/or file writing to using `BufferedReader+FileReader` and/or `BufferedWriter+FileWriter` in to your program as shown in Canvas→Modules→Week 10 (you will need to read ahead independently). If meeting bonus requirements mean that you have to violate a non-bonus requirement, explain this in your code documentation (marks will not be deducted as long as these are explained). If you choose to incorporate this bonus functionality, your bonus version is the only version that you must submit. Please do not include multiple versions of your code in the final submission. (+1 bonus mark).

5. Referencing guidelines

What: This is an individual assignment and all submitted contents must be your own. If you have used sources of information other than the contents directly under Canvas→Modules, you must give acknowledge the sources and give references using IEEE referencing style.

Where: Add a code comment near the work to be referenced and include the reference in the IEEE style.

How: To generate a valid IEEE style reference, please use the [citethisforme tool](#) if unfamiliar with this style. Add the detailed reference before any relevant code (within code comments).

6. Submission format

Submit **one .java file** of your program and one screenshot of it running via [Canvas→Assignments→Assignment 2](#).

If doing bonus B2, please also submit at least one sample data file (.csv or .txt only).

It is the responsibility of the student to correctly submit their files. Please verify that your submission is correctly submitted by downloading what you have submitted to see if the files include the correct contents.

7. Academic integrity and plagiarism (standard warning)

Academic integrity is about honest presentation of your academic work. It means acknowledging the work of others while developing your own insights, knowledge and ideas. You should take extreme care that you have:

- Acknowledged words, data, diagrams, models, frameworks and/or ideas of others you have quoted (i.e. directly copied), summarised, paraphrased, discussed or mentioned in your assessment through the appropriate referencing methods,
- Provided a reference list of the publication details so your reader can locate the source if necessary. This includes material taken from Internet sites.

If you do not acknowledge the sources of your material, you may be accused of plagiarism because you have passed off the work and ideas of another person without appropriate referencing, as if they were your own.

RMIT University treats plagiarism as a very serious offence constituting misconduct. Plagiarism covers a variety of inappropriate behaviours, including:

- Failure to properly document a source
- Copyright material from the internet or databases
- Collusion between students

For further information on our policies and procedures, please refer to the [University website](#).

8. Assessment declaration

When you submit work electronically, you agree to the [assessment declaration](#).

9. Rubric/assessment criteria for marking

Code must be valid, runnable Java to be given a mark (non-compilable code such as JavaScript, pseudocode, incomplete Java code cannot be marked). Run-time errors will incur up to a 50% penalty (run-time errors due to data type mismatches in inputs are acceptable).

	Inadequate	Partial	Complete (Uses only the concepts covered in class materials for meeting stated criteria)
C1	More than one one of the criteria in the 'complete' level missing/incorrect.	Any one of the criteria in the 'complete' level missing/incorrect.Any one of the criteria in the 'complete' level missing/incorrect.	One class file submitted with all required code. Only method declarations are at class-level (e.g. no member variables). Class name suits program and is not an arbitrary or default name. Identifier names are descriptive. Avoids hard-coding when appropriate. Formatting is consistent. Only relevant code and comments included. All code reachable. Does not use break, continue, System.exit anywhere in the code and does not return from the middle of methods.
C2	Missing/incorrect use of input taking OR some criteria in the 'complete' level missing/incorrect.	Any one of the criteria in the 'complete' level missing/incorrect but demonstrated as a part of meeting overall functional aims of program.	Appropriate prompts, input taking and storing demonstrated. Demonstrates understanding of primitive data types vs. class types where relevant. Above demonstrated as a part of meeting overall functional aims of program.
C3	Missing/incorrect use of else if OR some criteria in the 'complete' level missing/incorrect.	Any one of the criteria in the 'complete' level missing/incorrect but contains at least one reachable else if statement demonstrated as a part of meeting overall functional aims of program.	Uses if/else/else if appropriately and exclusively for non-repeating conditional execution. Contains at least one reachable else if statement. Conditions do not include tautologies and pathways are not redundant. Every code block in every if/else/else if structure is reachable. Does not use break, continue or return. Above demonstrated as a part of meeting overall functional aims of program.
C4	Missing/incorrect use of while loops OR some criteria in the 'complete' level missing/incorrect.	Meets all criteria of 'complete' level except while loops not used exclusively for repetition (e.g. additionally uses do..while, for, recursion, etc.)	Uses while loops appropriately and exclusively for repetition. Loop condition must describe all situations under which the loop will repeat and condition must fail eventually. Conditions do not include tautologies. Does not use break, continue or return. Above demonstrated as a part of meeting overall functional aims of program.
C5	Missing/incorrect use of arrays OR some criteria in the 'complete' level missing/incorrect.	One of the 'complete' level criteria not met but uses standard Java arrays as a part of meeting overall functional aims of program.	Uses standard Java arrays exclusively where data structures are needed (e.g. no ArrayList). Loops are used for traversing arrays and all array manipulation is performed without using other classes/code (e.g. does not use the Arrays class). At least one such array has its size determined at run-time.
C6	Missing/incorrect use of methods OR some criteria in the 'complete' level missing/incorrect.	At least one method in addition to the 'main' method is present as a part of meeting overall functional aims of the program but one of the other criteria in the 'complete' level missing.	Has at least one method in addition to the 'main' method. Methods used in ways to reduce code repetition. Methods are explicitly public and not static. Every created method is used in the program. Control/execution within each method always reaches the very last statement of that method (no spaghetti code; avoids return statements in the middle of the method). Methods may contain parameters and/or return values. Above demonstrated as a part of meeting overall functional aims of program.
D1	Lacks one or more of the 'partial' level criteria OR documentation not within .java file.	Identifies concepts required and application of each concept is discussed with respect to the overall aims of the program. Has an even spread of the above near code blocks and important variables in plain English where possible.	Further to 'partial' level criteria... Analyses the breaking down of the programming task to smaller parts (e.g. methods and other blocks) and organisation to achieve the overall goals of the program. Evaluates strategies used to meet requirements and compares the quality of one approach to another approach. Has an even spread of the above near code blocks and important variables in plain English possible.
D2	Does not discuss potential issues in the code OR documentation not within .java file.	Identification (without discussion) of the criteria stated under 'complete' or discussions exclude some of the criteria under 'complete'.	Explains any code requirements that have not been met, situations that might cause the program to crash or behave abnormally. Able to identify the approximate locations of any issues within the code and comments are given in those locations. Bugs imposed by limitations in the lesson topics discussed.