School of Science

COSC2542 Introduction To Programming

Assignment 3 (v.2020.11.02)

	Assessment Type: Individual assignment; no group work. Do not share your code. **Do not let others see your code.** Submit online via Canvas→Assignments→Assignment 3. Marks awarded for meeting requirements as closely as possible. For consistency, clarifications/updates will only be made via announcements/Canvas→Discussions→Assignment 3 discussion forum (no email clarifications and tutors will not make clarifications).
	Due date: Deadlines will not be advanced but they may be extended. Please check Canvas→Assignments for the most up to date information as this PDF may not be updated if changes are made. As this is a major assignment in which you demonstrate your understanding, a university standard late penalty of 10% per each working day (3 mark penalty) applies for up to 5 working days late (submission cutoff), unless special consideration has been granted. Extensions: For all new extension requests, please directly apply for special consideration online. Emailing your instructor first may lead to delays as the special consideration process has requirements on when to apply. If you have an Equitable Learning Plan (ELP) that permits extensions, you do not need to email for extension of up to 5 working days.
	Weighting: 30 marks + 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.

A programmer is also a developer who can plan, minimise risks, iteratively develop, test and deliver programs to a "client". As a part of this, a programmer should be able to show snapshots of the various stages of the development. The snapshot should be a runnable Java program that does not need to have all of the features that will be there in the final version of the program.

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.

2. You are able to analyse and evaluate your implementations against possible alternatives in your code documentation.

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.

You can convert your Assignment 2 program to suit the requirements of Assignment 3.

If there are questions, you must ask via the Canvas→Discussions→Assignment 3 forum in a general manner (replicate your problem in a different context in isolation before posting). **Do not show your assignment code to anyone** as this could violate Academic Integrity requirements set by the University.

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.

You must meet Functional Requirements (4.1), Code+Justification Requirements (4.2) and Documentation Requirements (4.3) to obtain the full mark for this assignment. You can also attempt the Bonus Requirements (4.4). You may receive 0 (zero) if you do not meet the submission requirements (see section 6).

4.1) Functional Requirements:

Important: The functional requirements below must be implemented and justified by following the 4.2 Code+Justification requirements.

F1) Allows the user to store an arbitrary number of records. May store more than 1 type of record.
F2) Allows the user to add, remove, modify such records.
F3) Allows the data for the records to be loaded from a file and saved later in to a file.
F4) Has no restrictions on how many user interfaces can be used simultaneously in the system.

Tip: As you are not given marks for creativity or the usefulness of the program, do not spend too much time thinking of what is a "good" program. What is good depends on how well the code is written, justified and documented (refer to sections 4.2 and 4.3).

4.2) Code+Justification Requirements (25 marks):

To receive marks for Code+Justification requirements, you must use the following Application.java and code concepts to make a functionally cohesive program that also meets the functional requirements.

```java
public class Application {
	private BackEnd backEnd;
	private FrontEndGTerm uiGT;
	private FrontEndConsole uiConsole;

	public Application() {
		this.backEnd = new BackEnd();
		this.uiGT = new FrontEndGTerm(this.backEnd);
		this.uiConsole = new FrontEndConsole(this.backEnd);
	}

	public static void main(String[] args) {
		Application app = new Application();
	}
}
```

The above file must not be changed by it should be possible to create multiple FrontEndGTerm objects to use the same BackEnd.

Concepts must be used exclusively as demonstrated by Gayan's weekly live lectures in this course offering. Code without justification in the given format would attract no more than 50% of the mark allocated for that component. Comments without code will not attract any marks.

An important note on Java code validity: A program with even one red dot (compilation error) cannot be tested and therefore will attract 0 marks for the entire submission. Test your code thoroughly.

Code concept	Code requirements below must also be justified as required
CJ1) Muti class Object Oriented Code with Application+BackEnd +FrontEndGTerm+FrontEndConsole and additional justifiable class(es) in the BackEnd 8x1=**8 marks**	• Application class and FrontEnd classes must not be split further in to additional classes. The Application class must not be submitted (the version provided in this document will be added by marker at time of marking). Only one class per .java file. • Must create at least one more class to reduce code/logic duplication and have an array of this type in the BackEnd. • Only the FrontEndGTerm must have any mentions of GTerm. (If written correctly, it should be possible to use multiple FrontEndGTerm objects simultaneously on the same BackEnd object.) • Only the FrontEndConsole must have any mentions of Scanner, System.out, etc. • The FrontEnd... classes and BackEnd classes must communicate exclusively using strings/primitive data types/wrapper classes of primitive types (arrays permitted). FrontEnd classes only communicate with the BackEnd class directly (does not know other classes used by BackEnd). • Code supplied by student does not use *static* anywhere. • Each class has object member variables that are explicitly private and non-static. Whenever a method refers to an object member variable, it uses this. (i.e. "this dot", e.g. this.name). • Each class has one constructor and all object member variables, arrays, etc. declarations are explicitly initialised in this constructor before any other operations (e.g. there are no equal signs where member variables are declared).
CJ2) Other general 4x0.5=**2 marks**	• All identifiers and names are descriptive and appropriate for their purpose. Follows conventions shown in lectures, other standard class materials and common ones in the Java API. • Formatting is consistent. Justification comments start on the line before the documented block/statement (e.g. not in-line comments). • Only relevant, reachable code+comments included. • Does not use break, continue, System.exit or similar branching anywhere in the code does not return from the middle of methods.
CJ3) Variables 2x1=**2 marks**	• Does not use literals when justifiable. • Demonstrates understanding of primitive data types vs. class types where relevant.
CJ4) Methods 4x0.5=**2 marks**	• Methods are created when absolutely necessary or when it reduces duplication of code. • One or more classes have methods that return values (e.g. accessor/get methods) • One or more classes have methods that take parameters (e.g. mutator/set methods). • All methods are explicitly public and non-static.
CJ5) User interface (FrontEnd...) class requirements 7x0.5=**3.5 marks**	• Both FrontEnd... classes must offer an identical set of functionalities to the end-user. • FrontEndGTerm uses GTerm exclusively for inputs and takes most, if not all, inputs via text fields/text areas/password fields (vs. getInputString). • FrontEndConsole uses Scanner's .nextLine() exclusively for inputs (e.g. must not use other .next... methods). • FrontEndGTerm must produce all outputs exclusively using GTerm. Whenever possible, outputs must be in the main window (vs. via showMessageDialog, etc.). • FrontEndConsole must produce all outputs exclusively using System.out or System.err. • FrontEndGTerm uses textfields/textareas, button(s) and table(s) as shown in weekly live lectures. May use multiple GTerm windows within one FrontEndGTerm object. If using addImageIcon, images must be loaded from the default/project folder (do not include folder names) without prompting the user to choose the image file. • FrontEndConsole shows text-based menus and formats outputs in to tables, etc. as shown in weekly live lectures.
CJ6) Conditional execution and repetition 3x0.5=**1.5 marks**	• Uses if/else/else if appropriately and exclusively for non-repeating conditional execution and at least one reachable else if statement. • Uses while-loops appropriately and exclusively for repetition. Loop condition describes all situations under which the loop will repeat and condition fails eventually. • Conditions do not include tautologies. Pathways are not redundant. Every code block in every if/else/else if/while structure is reachable.
CJ7) Arrays 4x0.5=**2 marks**	• Only standard Java arrays are used (e.g. does not use ArrayLists, etc.) • The size of this array mentioned in CJ1.2 should be able to vary at run-time. • Some array lengths are determined at run-time (e.g. based on how many records the user wants to store). • All array manipulation performed by student using while-loops, if-statements, etc. i.e. Must not use classes such as Arrays, etc. for array manipulation.
CJ8) File input+output from BackEnd 4x1=**4 marks**	• Only the BackEnd class must have file input/output code. • Uses only BufferedReader+FileReader when reading. • Uses only BufferedWriter+FileWriter when writing. • Uses only .txt or .csv files and these are placed in the default/current folder (paths do not contain folder/directory names). May allow the user to choose the data files to open.

Justification Requirements
Note: You will not receive full marks for the CJ requirements unless each occurrence is justified as required below.

Type of code	Compare and justify your choice over other possible alternative...
Declarations (also applies to class and method definitions)	Identifier names Data types Locality of declaration (why object-level vs. parameter-level vs. method-level vs. block-level, etc.).
Contents of code blocks	Formulations (is there a simpler way to meet requirements without creating this code block?) Inclusions (what you have added and why?) Sequences (why are these in this order?) Exclusions (what you haven't added and why)
Conditions	Formations of the logic

4.3) Documentation Requirements (5 mark penalty if not met):
Important note: Documentation must match with testable, functional and justified code to attract marks.

D1. Create an illustrated PDF user guide (one file). ***Must have side-by-side instructions*** *on how to perform the same functionalities on both interfaces. Shows screenshots of sample inputs. Shows screenshots of corresponding sample outputs.* ***Must include examples of using files to load and save.*** Include *instructions on what the user can and cannot do (e.g. what they can and can't* input*). Does not include any references to code as the guide is intended for a user who is not a programmer. This document* needs to be professionally presented, more structured, starting with a table of contents, page numbers, clearly marked sections, etc.

4.4) Bonus Requirements

Important note: The total mark of A1+A2+A3 is capped at 50 (for a full break-down, see Canvas→Assignments). To obtain any bonus marks, you need to be able to get full marks for the non-bonus/standard requirements of this assignment.

B1: Submit your final version of Assignment 3 one day before the deadline for +0.1 marks or 2 days before the deadline for +0.2 marks, etc.

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 from the lectures and tutorials, you must give acknowledge the sources and give references using IEEE 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

Via Canvas→Assignments→Assignment 3, submit all of the following ***in one go****, each time you submit:*

1. All .java files except Application.java (see section 4.2)
2. All required images, data files, etc. (see section 4.2)
3. PDF user guide (see section 4.3)

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. It is not an issue if Canvas renames your submission.

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.