School of Science

COSC2452 Introduction To Programming (OUA)

Assignment 2 (v.2020.12.21)

	Assessment Type: Individual assignment; no group work. **Do not even let anyone see** any code that will be used for a submission. Submit online via Canvas→Assignments→Assignment 2. Marks awarded for meeting requirements as closely as possible. For consistency, clarifications/updates will only be made via announcements/Assignment 2 discussion forum (no email 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 day late (1.5 mark penalty) applies for up to 1 week and submissions are not accepted after (unless special consideration received). Extensions: For all new extension requests, please follow the central special consideration process directly to avoid delays.
	Weighting: 15 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.

If there are questions, you must ask via the Canvas→Discussions→Discussions→Assignment 2 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.

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).

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 of the same type. May store more than 1 type of record.

F2) Allows the user to add, remove and modify such records.

F3) Uses at least two separate windows (that can exist simultaneously): one for main menu or high-level operations and at least one more for more specialised operations (e.g. entering/editing details).

F4) Have information and operations organised within the windows using tables, text fields, buttons and presentation-related operations from weekly live lectures.

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 (15 marks):

To receive marks for Code+Justification requirements, you must use the following code concepts to make a functionally cohesive program that also meets the functional requirements. You must only use concepts explained and demonstrated in the weekly live lectures held by Gayan (typically held on Monday nights). If you require the use of some additional concepts, please seek clarification through the Canvas→Assignments→Assignment 2 forum. Your work cannot be simply a renamed version of an example shown in class. Code without justification in the required 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 this section.

Code concept	Requirements (15 marks when justified as required) – No partial marks for dot points
CJ1) Single-class object oriented code using provided template with... 1x3=3 marks	• Appropriate name for java file (must not use names like Assignment2.java); Follows conventions shown in IIE solution lectures, other standard class materials and common ones in the Java API. The main method should have only one line to create an object of the main application class (refer to startup code and week 6+ weekly live lecture examples). • Consistent code and comments formatting, with comments start on the line before the documented block/statement (e.g. not on lines with code). Only relevant, reachable code+comments included. • No uses of return in middle of methods, break, continue, System.exit or similar branching (spaghetti code) anywhere in the code.
CJ2) Variables... 0.5x2=1 mark	• Some of which are object member variables (may include arrays). These are explicitly private, non-static and there should be no = signs near declarations. Every reference to an object member variable from a method starts with this. (i.e.

	“this dot”, e.g. this.gt). Descriptive variable names used and does not use vague names (e.g. numRecords) • Used in place of duplicated literals. Demonstrates understanding of primitive data types vs. class types where relevant.
CJ3) Constructor 1x1=1 mark	• The class has only one constructor and all object member variables, arrays, etc. declarations are explicitly initialised in this constructor before any other operations.
CJ4) Methods 1x2=2 marks	• All methods are explicitly public and non-static. Methods are created when absolutely necessary or when it reduces duplication of code. • One or more methods created by student must take parameters and one or more methods created by student must return values. Both could be demonstrated using the same method.
CJ5) Multi window user interface using GTerm... 1x3=3 marks	• Uses GTerm exclusively for inputs. Most, if not all, user inputs must be taken via either text fields or text areas (vs. getInputString). May use password fields and dialogue boxes. • Uses GTerm exclusively for outputs. Must use tables with headings and columns. Minimal use of .show...Dialog methods to display outputs. Uses GTerm’s methods setXY, addImageIcon, setFontSize, etc. to improve presentation. • Uses GTerm buttons. Performs operations on rows selected from tables.
CJ6) Conditional execution and repetition 1x2=2 marks	• Uses if/else/else if appropriately and exclusively for non-repeating conditional execution and at least one reachable else if statement. Conditions do not include tautologies. Every code block in every if/else/else if/while structure is reachable. • 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.
CJ7) Arrays 1x3=3 marks	• Uses multiple arrays of primitive or shown Java API classes to maintain records. Only standard Java arrays used. (e.g. does not use ArrayLists, etc.) • 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. without using other classes.

Justification Requirements

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

Type of code	**Evaluate and justify your choice over other possible alternative...**
Declarations (also applies to method definitions)	Identifier names Data types Localities 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 (e.g. “is glass half full” vs. “is glass half empty”, etc.)

Template/Start-up code

In the most initial form, your code must take the following organisation:

```java
public class Assignment2RenameThisClass {
          private GTerm gtMain;
          private GTerm gtSub;

          public Assignment2RenameThisClass() {
                    this.gtMain = new GTerm(600, 400);
                    this.gtSub = new GTerm(400, 600);

          }

          // The main method must only perform the included operation.
          // Do not add any other code to the main method.
          public static void main(String[] args) {
                    Assignment2RenameThisClass a2obj = new Assignment2RenameThisClass();
          }
}
```

You can rename identifiers to suit. You can create the second GTerm object later but it still must be initialised in the constructor.

4.3) Documentation Requirements (**3 mark penalty if requirements below 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 be written to show a non-programmer how to make use of the functions of the application (based on functional requirements). Must include screenshots of sample inputs and relevant/corresponding outputs. Include instructions on what the user can and cannot do (e.g. what they can and can’t input). Do not refer to code in the written sections or show any code even in the screenshots.

4.4) Bonus Requirements

Important note: The total mark of A1+A2+A3 is capped at 45 (for a full break-down, see Canvas→Assignments). To obtain any bonus marks for this assignment, you need to be able to get full marks for the non-bonus/standard requirements of this assignment.

B1: Submit at least 1 work-in-progress version of your assignment (minimum requirement is CJ1) and then submit your final version of assignment 2 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 directly under Canvas→Modules, 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 2, submit all of the following in one go, every time you submit:

1. One .java file (see section 4.2)
2. All images required by code (see section 4.2. CJ5)
3. One 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.