RMIT UNIVERSITY

Introduction To Programming
COSC2452
Assignment 1

Assessment Type	Individual assignment (no group work). Marks are awarded for meeting requirements as closely as possible. Clarifications/updates will only be made via official announcements and Canvas→Discussions→Assignment 1. Must follow submission instructions in this document.
Due	Please always refer to the up-to-date deadlines given under Canvas→Assignments. Deadlines will not be advanced but they may be extended. **Late submissions:** The late submission period lasts for 5 working days from the deadline. A 10% late penalty will apply for each working day late after the deadline for up to 5 working days. After this period, submissions will not be accepted (unless special consideration has been granted.) **Extensions:** For new extensions, please apply directly via the RMIT Special Consideration page.
Marks	10 + bonus marks

1. Expectations

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.

In this assignment, you are awarded marks for demonstrating your ability to meet the requirements given in this document using no more than the concepts and approaches taught in Introduction To Programming compulsory weekly live lectures in a justifiable way to code and debug a working Java program.

Consider the simplest solution that meets the requirements as the best solution (Occam's razor principle). There are functional requirements, code justification requirements and documentation requirements that must be met to receive the full mark for this assignment.

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.

2. 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.

RMIT UNIVERSITY

3. Assessment details

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

Assume that a children's software developer has hired you for your coding skills to work on a project where you must create an interactive story, using GTerm (all inputs and outputs must be via GTerm). Please do/watch the week 1 exercises on getting started on this with Eclipse and GTerm.

You must meet Functional Requirements (3.1), Code+Justification Requirements (3.2) and Documentation Requirements (3.3) to obtain the full mark for this assignment.

3.1) Functional Requirements:

Important: The functional requirements below must be implemented and justified by following the 3.2 Code+Justification requirements.

F1: When your program is run, the program asks the user to enter various inputs (e.g. names of characters, actions to take, numbers of things, etc.) that are relevant to the chosen story and then, displays the story in an interactive manner.
F2: At least one of the values entered must be a numerical value and at least one of the values must be text.
F3: The story must have different pathways depending on the inputs.

Tip: As you are not given marks for creativity, you can come up with your own story or feel free to use one found online (provide references when relevant). As this is for hypothetical children, please keep the non-changing parts of the story "tame".

3.2) Code+Justification Requirements (7.7 marks):

To receive marks for Code+Justification requirements, you must use the following code concepts as demonstrated during weekly live lectures and add code justification comments when meeting the 3.1 Functional Requirements. 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 this section. If the code produces run-time errors a 50% penalty will be applied to this section.

Code concept	Requirements (0.35 x 22 = 7.7 marks when justified as required)
CJ1) Java class	• Has only 1 class in 1 .java file and it is appropriately named to suit the application (must not use names such as Assignment1.java); Follows conventions shown in tutorial solution and lectures, other standard class materials and common ones in the Java API. • All code must be within the main method. • 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 return, break, continue, System.exit or similar branching (spaghetti code) anywhere in the code.
CJ2) Variables and inputs	• Must be able to store at least one relevant String value. • Must be able to store at least one relevant numerical value. • The user must be allowed to enter these values at run-time and the inputs must come via a suitable GTerm method. These values must be embedded in the story (refer CJ3). • Demonstrates understanding of primitive data types vs. class types where relevant.
CJ3: User interface design using GTerm version '2020.08.25' or newer.	• Story must be shown within the GTerm window (e.g. not via "show...Dialog" windows) • Use GTerm exclusively for inputs • Use GTerm exclusively for outputs • Use GTerm methods setXY, setFontSize, setFontColor, setBackgroundColor, clear • Uses GTerm method addImageIcon to enhance the presentation of the program by loading existing .gif, .png or .jpg images from the default/project folder (do not include folder names) without prompting the user to choose the file.
CJ4) If-statements	• Must exclusively use if-statements for non-repeating conditional execution (e.g. no switch). • There must be at least one else-if statement. • There must be an example of nested if-statements. • There must be a conditional check performed on the text input. *Continues on next page...*

	• There must be a conditional check performed on the numerical input. • Conditions do not include tautologies. Pathways are not redundant. Every code block in every if/else/else if/while structure is reachable. • The story must branch and have pathways that vary depending on user entered values. • Each branch/pathway must also include either inputs and/or outputs using GTerm (refer to CJ2 and CJ3)

Justification Requirements

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

Declarations, code blocks and conditions must be justified by following the rationales given in the 'weekly live lectures'.

Type of code	Compare and justify your choice over other possible alternative...
Declarations (also applies to method definitions)	Identifier names Data types
Contents of code blocks	Formulations (i.e. is there a simpler way to meet requirements without creating this code block?) Inclusions (i.e. what you have added and why? Can these be done before/after this code block?) Sequences (i.e. why are the statements in this order?) Exclusions (i.e. what you haven't added and why)
Conditions	Formulations of conditions (e.g. "is glass half empty" vs. "is glass half full")

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 1' when in doubt.

3.3) Documentation Requirements (2.3 marks):

Important note: Documentation must match with testable, functional and justified code to attract marks.

D1. Create an illustrated PDF user guide (one file). Shows screenshots of sample inputs. Shows screenshots of corresponding sample outputs. Has brief instructions on what the user can and cannot do (e.g. what they can and can't input). Does not show code and does not have any references to code; the guide is intended for a user who is not a programmer.

3.4) Bonus marks:

Important note: To become eligible for any bonus marks for this assignment, your non-bonus component of this assignment must receive full marks. The total mark of Assignments 1+2+3 is capped at 50.

B1. Make your final submission 1 day before for +0.1 bonus marks or 2 days before for +0.2 bonus marks, etc.

4. Submission

All files required for the operation of your code must be submitted in one go (do not submit one after another; the 'submit' button must only be selected once) via Canvas→Assignments→Assignment 1 (emailed files will not be accepted). Only the last set of files submitted is considered as your official submission by the university for marking purposes. Therefore you should aim to submit multiple times to minimise risks. You must also verify your submission to ensure that the correct files are submitted. It is OK if Canvas automatically renames your files upon resubmission.

Required files:

1. The .java file (only this file should contain code justification comments). No .java file or an invalid Java file means no mark even if other files are submitted.

2. All required .png, .jpg or .gif images used by addImageIcon.

3. UserGuide.PDF containing the user guide documentation (must not contain any references to code)

Assessment declaration:
When you submit work electronically, you agree to the assessment declaration:
https://www.rmit.edu.au/students/student-essentials/assessment-and-exams/assessment/assessment-declaration

5. 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 https://www.rmit.edu.au/students/student-essentials/rights-and-responsibilities/academic-integrity