




COSC2452 Introduction To Programming (OUA)

Assignment 1

	Assessment Type: Individual assignment; no group work. Submit online via Canvas→Assignments→Assignment 1. Marks awarded for meeting requirements as closely as possible. Clarifications/updates may be made via announcements/relevant discussion forums.
	Due date: 6:30pm, 16- 23/September/2019; Deadlines will not be advanced but they may be extended. Please check Canvas→Assignments 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 (submission cutoff), unless special consideration has been granted.
	Weighting: 7 marks + 1 bonus mark

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.

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 lecture lessons and “modules” to write and debug a working Java program. Consider the simplest solution that meets the requirements as the best solution ([Occam's razor principle](#)). There are code requirements (5 marks) and documentation requirements (2 marks) for a total of 7 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.

2. Assessment Criteria

This assessment will determine your ability to:

1. Exclusively follow coding, convention and behavioral requirements provided in this document and in the compulsory live 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.

Assume that a children's software developer has hired you for your coding skills to work on a project involving personalised/dynamic stories/poetry named 'StoryMaker'. At first, the program asks the user to enter several fields such as names of characters/places, numbers of things, etc. that are important to the chosen story and then finally displays that story with the entered fields embedded (similar to the Myself program but with inputs). You must not use the given example story. You must meet the code (section 4.1) requirements and documentation (section 4.2) requirements detailed further below.

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".

4.1) Code requirements (1+1+1+2=5 marks):

C1. Code presentation and format – The program must be entirely inside the main method of a Java class/file named StoryMaker.java and formatted consistently (Tip: use Eclipse→Source menu→Format before every submission). Must not include any unused/irrelevant code (even inside comments).

C2. The story/poem must be able to store at least two String fields and one number field based on the chosen story.

C3. Values for the fields must come via user inputs when the program is run. All inputs must be taken before any part of the story is shown.

C4. The final composition of the story must be in one variable and it must be displayed using one output statement. Refer to the behaviour demonstrated in the Myself and TypeDemo examples. Consider these as requirement presentations by the "client".

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" (you instructor) under Canvas→Discussions→'Assignment 1' when in doubt.

4.2) Documentation requirements (1+1=2 marks):

D1. Write comments next to defined code blocks (e.g. near public static void main, ifs, etc. where you can have { }) and declared variables explain their purpose in the program and what alternatives would have been possible. Do not assume that the reader can figure these out.

E.g. with important variables, discuss which data type you have used and what other data types might have been possible/impossible and why.

E.g. with if statements (if you have them), discuss how else one might perform the same conditional check or if there is no other way, state why it would be impossible.

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) The following is only an example. You must choose your own story and relevant fields.

"Enter name of big animal" (You enter "lion")

"Enter name of small animal" (You enter "mouse")

"Enter story duration in days" (You enter 2)

[Your code should then display in one go (similar to the Myself program shown in week 1 tutorial lesson)...]

Once upon a time there lived a lion in a forest. One day after a heavy meal it was sleeping under a tree. After a while, there came a mouse and it started to play on the lion. Suddenly the lion got up with anger and looked for those who disturbed its nice sleep. Then it saw a small mouse standing trembling with fear. The lion jumped on it and tried to kill it. The mouse requested the lion to forgive it. The lion felt pity and left it. The mouse ran away. 2 days later, the lion was caught in a net by a hunter. The mouse came there and cut the net. Thus it escaped. There after, the mouse and the lion became friends. They lived happily in the forest afterwards.

4.4) Bonus requirements (bonus marks for this assignment are capped at 1)

You can do either or both of the requirements below but to obtain any bonus marks, you must meet all requirements of the non-bonus/standard part.

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 marks.

B2. By incorporating if-statements to perform conditional checks, make your story change based on the inputs. You can make it like an interactive adventure where the story is displayed progressively. 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 marks).

Note: 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.

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 **two files**: only the final version of your **StoryMaker.java** file and a **screenshot** showing the final output of your program via Canvas→Assignments→Assignment 1. 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

For full marks, you must use only the concepts taught in the compulsory Live Lecture (and relevant Canvas→Modules); programs that work do not necessarily receive full marks if required approaches are not demonstrated. Code must also be valid, runnable Java to be given a mark (non-compileable code such as JavaScript, pseudocode, incomplete/broken 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)
C1	Formatting inconsistent or irrelevant code included.	Program name not an exact match or formatting somewhat inconsistent or irrelevant code included or has unreachable code.	Correct program name, formatted consistently, only relevant code and comments included.
C2	Does not demonstrate knowledge of data types and declarations.	Required number of fields in required types not present or identifier names may not be descriptive but shows understanding of the concepts of storing values in relevant data types.	Able to store at least two String fields and one number field. Demonstrates understanding of primitive data types vs. class types where relevant. Uses descriptive identifier names.
C3	Relevant input-taking not present.	Does not show appropriate prompts or only some fields are taken as inputs or not all inputs are taken at once.	Appropriate prompts, input taking and storing demonstrated. All inputs taken before any part of the story is shown.
C4	No output or story hard-coded to output statements.	Multiple output statements or entire story not stored in one variable.	Final composition of the story must be in one variable and displayed using one output statement.
D1	Purposes of most variables and code blocks not explained.	Explains purpose of most declared variables and defined code blocks but lacks detail given under 'complete'.	Explains purpose of each declared variable in plain English and analyses possible alternative declarations. Explains purpose of each written code block in plain English and analyses possible alternative implementations or orderings.
D2	No comments aside from D1 'complete' level. Does not discuss issues in the code.	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. If doing B2, discusses if any B2 requirements violate non-bonus requirements.

