




School of Science

# Assignment 1

## COSC1284 Programming Techniques

July 2020

---

	<b>Assessment Type:</b> Individual assignment.  Submit online via Canvas → Assignments → Programming Assignment #1.  Marks awarded for meeting requirements as closely as possible. Clarifications/updates may be made via announcements/relevant discussion forums.
	<b>Due date:</b> Week 5, Friday 21st August 2020, 11:59pm.  Late Submissions/Extensions: A penalty of 10% per day is applied to late submissions up to 5 days, after which you will lose ALL the assignment marks. Extensions will be given only in exceptional cases; please refer to the special consideration process.
	<b>Weighting:</b> 25%

## 1 Overview

A programmer can 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, you will take on the role of a freelance programmer who has been given a small program to write.

The program will present a menu to the user with two functions:

- Unit Converter
- Lotto Checker

The requirements for the menu and the two functions are reported in detail below.

### Menu

The menu presents the user with a choice of running either the Unit Converter or the Lotto Checker. The user inputs required by the user are presented as part of the menu display.

The implementation of this method must abide by these constraints:

- M1. The menu will run as a once-off, and it should not be re-displayed again as the program will terminate after any function has been completed.
- M2. An incorrect input will cause the program to prompt the user for another input.
- M3. After three unsuccessful attempts, an incorrect input will cause the program to display an error message to the user (e.g., "The input was incorrect!") and then terminate.
- M4. A correct input will cause the program:
  - a. To execute the chosen function;
  - b. To require the user input of their three parameters from the console;
  - c. To display the output of the chosen function;
  - d. To finally terminate.

## Unit Converter

This method will be declared as such: `private double convert(double number1, int operation)`

The method performs one of six predetermined unit conversion operations: convert Celsius to Fahrenheit, convert Kelvin to Fahrenheit, convert Celsius to Kelvin, convert Fahrenheit to Kelvin, convert Fahrenheit to Celsius, and convert Kelvin to Celsius.

Each of these operations are represented by an integer: Celsius to Fahrenheit (0), Kelvin to Fahrenheit (1), Celsius to Kelvin (2), Fahrenheit to Kelvin (3), Fahrenheit to Celsius (4), and Kelvin to Celsius (5).

The implementation of this method must abide by these constraints:

- U1. Two parameters must be taken as input
- U2. The first parameter should be the unit to be converted.
- U3. The second parameter takes values in the range 0--5, and it indicates the type of conversion to be performed.
- U4. The method returns only the result of the conversion.
- U5. The method returns -1.0, if the operation supplied is invalid.

## Lotto Checker

This method will be declared as such: `private String checkLotto(int chosen1, int chosen2, int chosen3)`

The implementation of this method must abide by these constraints:

- L1. Lotto numbers are drawn in the range between 1 and 48.
- L2. A number may be drawn one or more times in any single draw.
- L3. This method should receive 3 number picks from a user that represent the numbers chosen by a player.
- L4. The numbers chosen by the player must fall within the range of the field of numbers being drawn.
- L5. Each time the method is called it will generate eight random numbers within the predefined range (1--48).
- L6. The method should return a string that informs the user of the results of the draw including the drawn numbers, the numbers chosen by the player, which numbers were correct guesses by the player and how many correct guesses occurred in the draw.
- L7. The method should also notify of any error e.g., if the number picked by the user fall outside the prescribed range.

**This program will demonstrate the following key skills:**

1. Creating a small program to demonstrate what you have learned as a developer.
2. Analysing a problem and developing an algorithm to solve the problem.
3. Converting an algorithm to computer code.
4. Writing code using best practices (within the limitations placed on you by this specification)
5. Documenting your program.

## Read the requirements thoroughly!

There are specific constraints that have been placed on you for this assignment to force you to work within defined parameters. The ability to work within a pre-defined set of parameters is a fundamental skill required by any software developer.

You will also need to debug code on your own.

You are given marks on your ability to fulfil all requirements of this document.

Develop this assignment in an iterative fashion, as opposed to completing it in one sitting.

You should get started now as there are concepts even from the week 1 lessons that you can incorporate into your program.

Any questions regarding this assignment must be asked via the relevant Canvas discussion forums only (no emails to teaching staff, please) in a general manner.

## 2 Learning Outcomes

This assessment relates to the following course learning outcomes:

- CLO1. Demonstrate (through small programming exercises) knowledge and skills with concepts of program design and acceptable coding standards.
- CLO2. Use Java programming language as a vehicle to demonstrate good software development practices.
- CLO3. Use arrays and control structures to demonstrate skills of basic algorithms and data structures.
- CLO5. Analyse requirements for a small-scale programming project.
- CLO6. Design and implement small-scale software systems.
- CLO8. Demonstrate skills for self-directed learning.

## 3 Assessment Details

Your code must meet the following code requirements (section 3.1) and documentation requirements (section 3.2).

### 3.1 Coding Requirements (15 marks)

The following code requirements must be applied similarly to what has been shown in lesson materials. Also, refer to corresponding rows in the rubric (see Section 6).

C1. Code presentation and format:

The program must be written in exactly two files. One file should contain your 'main method' and should only be responsible for starting the program. The name of the class (file) with the 'main' method must be called 'Driver.java'. The second file should contain the program logic. The name of the second class (file) must be called A1.java. Code is formatted consistently and follows the standard Java coding conventions. You must not include any unused/irrelevant code (even inside comments); the code you submit must be considered the final product. The code should be submitted as a Visual Studio Code Java project.

#### C2. Data types, user inputs and program outputs:

You must use the Scanner class for any user input related functions. Variables and constants should reflect the most appropriate data type for the data they are representing. Variables and constants should not be defined at the class level i.e. only defined locally with a method. Constants should be used for any fixed values even if only used once in the program to increase readability of your code. You should avoid the use of 'hard coded' (magic numbers) values when used they must be defined as a constant. Formatting of output to the console must be done using the printf method and not via escaped tab sequences or manual spacing. You are not permitted to use ArrayLists or any other data structures. All your variables and constants must be a primitive type or a String data type.

#### C3. Conditional Logic:

You should use if/else if/else (or switch/case) appropriately and exclusively for non-repeating conditional execution. Contains at least one reachable else if statement. Every code block in every if/else - if/else (or switch/case) structure must be reachable. You must not have any redundant conditions.

#### C4. Repetition

You are not permitted to use any type of loop in this program. Any code repetition should be reduced by writing cohesive methods.

#### C5. Algorithm development and method implementations

You must not use static anywhere in the program except in the main method. Methods should be cohesive and used to represent each step in an algorithm in order to keep each method as small as possible. There is at least one method in addition to the 'main' method. Methods are used to reduce code repetition. Every created method is used in the program. Methods are appropriately private or public. Control/execution within each method always reaches the very last statement of that method (no spaghetti code e.g., please, avoid return statements in the middle of the method). Methods may contain parameters and/or return values.

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 instructors under Canvas → Discussions → Programming Assignment #1 when in doubt.

## 3.2 Documentation requirements (10 marks)

D1. The files (classes) in your program must be FULLY documented. This means that you should have class level comments, method level comments and inline comments. The class level comments should specify the name of the class, your name, your student number and a short description of the purpose of the class.

Method level comments should include a short description of the method and the steps of the algorithm that the method is implementing. You must also include a description of any parameters that the method takes and the return value of the method.

Your variable, constant and method names should also be named appropriately to represent a 'self-documenting' style of programming.

Every method must have a method level comment regardless of its size/complexity. You may use inline comments sparingly where it helps the readability of your program.

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.

## 4 Submission

Submit a zipped archive of a Visual Studio Code project that contains two .java files via Canvas → Assignments → Programming 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 content.

### 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

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>


## 6 Marking Guidelines

Code must be valid, runnable Java to be given a mark (code that cannot be compiled 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 of the criteria in the 'complete' level missing/incorrect.	Any one of the criteria in the 'complete' level missing/incorrect.	<p>The program must be written in exactly two files.</p> <p>One file should contain your 'main method' and should only be responsible for starting the program.</p> <p>The name of the class (file) with the 'main' method must be called 'Driver.java'.</p> <p>The second file should contain the program logic.</p> <p>The name of the second class (file) must be called A1.java.</p> <p>Code is formatted consistently and follows the standard Java coding conventions.</p> <p>You must not include any unused/irrelevant code (even inside comments);</p> <p>The code you submit must be considered the final product.</p> <p>The code should be submitted as a Visual Studio Code project.</p>
C2	More than one of the criteria in the 'complete' level missing/incorrect.	Any one of the criteria in the 'complete' level missing/incorrect.	<p>You must use the Scanner class for any user input related functions.</p> <p>Variables and constants should reflect the most appropriate data type for the data they are representing.</p> <p>Variables and constants should not be defined at the class level i.e. only defined locally with a method.</p> <p>Constants should be used for any fixed values even if only used once in the program to increase readability of your code.</p> <p>You should avoid the use of 'hard coded' (magic numbers) values when used they must be defined as a constant.</p> <p>Formatting of output to the console must be done through the use of the printf method and not via the use of escaped tab sequences or manual spacing.</p> <p>You are not permitted to use ArrayLists or any other data structures. All your variables and constants must be a primitive type or a String data type.</p>
C3	More than one of the criteria in the 'complete' level missing/incorrect.	Any one of the criteria in the 'complete' level missing/incorrect.	<p>Uses if/else - if/else (or switch/case) appropriately and exclusively for non-repeating conditional execution.</p> <p>Contains at least one reachable else if statement.</p> <p>Every code block in every if/else - if/else (or switch/case) structure is reachable.</p> <p>Must not have any redundant conditions.</p>
C4	More than one of the criteria in the 'complete' level missing/incorrect.	Any one of the criteria in the 'complete' level missing/incorrect.	<p>You are not permitted to use any type of loop in this program.</p> <p>Any code repetition should be reduced by writing cohesive methods.</p>



C5	More than one of the criteria in the 'complete' level missing/incorrect.	Any one of the criteria in the 'complete' level missing/incorrect.	<p>Must not use <code>static</code> anywhere in the program except in the main method.</p> <p>Methods should be cohesive and used to represent each step in an algorithm in order to keep each method as small as practical.</p> <p>Has at least one method in addition to the 'main' method.</p> <p>Methods used in ways to reduce code repetition.</p> <p>Every created method is used in the program.</p> <p>Methods are explicitly appropriately private or public.</p> <p>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).</p> <p>Methods may contain parameters and/or return values.</p>
D1	More than one of the criteria in the 'complete' level missing/incorrect.	Any one of the criteria in the 'complete' level missing/incorrect.	<p>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 are named in plain English when possible.</p> <p>The files (classes) in your program must be FULLY documented. This means that you should have class level comments, method level comments and inline comments. The class level comments should specify the name of the class, your name, your student number and a short description of the purpose of the class.</p> <p>Analyses each method with method level comments that include a short description of the method and the steps of the algorithm that the method is implementing. You must also include a description of any parameters that the method takes and the return value of the method.</p> <p>Your variable, constant and method names should also be named appropriately to represent a 'self-documenting' style of programming.</p> <p>Every method must have a method level comment regardless of its size/complexity.</p> <p>You may use inline comments sparingly where it helps the readability of your program.</p>
D2	More than one of the criteria in the 'complete' level missing/incorrect.	Any one of the criteria in the 'complete' level missing/incorrect.	<p>Includes a readme file that contains the following:</p> <p>Evaluate and explain any code requirements that have not been met including situations that might cause the program to crash or behave abnormally.</p> <p>Identifies and explains the parts of the code that could be improved using repetitive algorithms (loops) and/or data structures that are able to hold more than one value.</p>