RMIT UNIVERSITY

School of Science

COSC2531 Programming Fundamentals

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: **end of Week 6**; Deadlines will not be advanced but they may be extended. Please check Canvas → Assignments → Assignment 1 for the most up to date information. As this is a major assignment, a university standard **late penalty** of 10% per each working day applies for up to 5 working days late, unless special consideration has been granted.
	Weighting: **10 marks** out of 100

1. Overview

The objective of this assignment is to develop your programming and problem solving skills in a step-by-step manner. The different stages of this assignment are designed to gradually introduce different concepts, such as loops, arrays, and methods. Students may be awarded partial marks for explaining a valid strategy, even if the program is not working.

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.

If there are questions, you must ask via the relevant Canvas 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.
Your code must meet the following code and documentation requirements.

MyBlock is a simple tool which can help one design a block of land. Your task is to write a Java program that implements the design functionality, checks if all the rules are followed, displays the block. The basic structure of the code is provided in the Canvas shell. You need to complete the following tasks. ***Please use the skeleton code provided to complete your code.***

Your program should consist of a `MyBlock` class. The class has a two dimensional array of integers called `block`. The class also has a Boolean variable `vacant` that stores the information of whether the block of land is vacant or not. The methods of this class are: `displayBlock()`, `addHouse()`, and `clearBlock()`.

Task A
The constructor of the `MyBlock` class takes 2 parameters: the number of rows of the block, the number of columns of the block. In the constructor of this class, write code to initialize the block with the size of row and column. Initialize each value of the array `block` with the value `0`, which means that block is unused. Initialize the value of `vacant` to `true`.

Task B
Write a method of the `MyBlock` class called `displayBlock()` that prints the block as a two dimensional array. Give a space between each element during printing, and use a line for each row. An example of the output of `displayBlock()` is shown below, for a vacant block of 4 x 3.

```
0 0 0 0
0 0 0 0
0 0 0 0
```

Task C
Write a method of the `MyBlock` class called `clearBlock()` that sets the value of each element of the array '`block`' to zero (`0`). Set the value of `vacant` to `true`.

Task D
From the main method, the user can enter the row and column of the block. The number of rows and the number of columns should be an integer **greater than 2** and **less than or equal to 10**. If any input is incorrect, show an error message and ask for that input again. If all inputs are correct, create an object of `MyBlock` class from main method. The row and column values are passed as the parameter of its constructor.

Task E
From the main method, show a menu to the user with the following options. If the input is neither 1,2,3,4, show an error message and ask the user for input again.

1. Add a house
2. Display the block
3. Clear the block
4. Quit

Option 1 – Add a house
It prompts the user of the position of the house (x, y) and the number of rows and columns of the house and then call the `addHouse()` method of the `MyBlock` class with those values as parameters (in the order of x, y, row, column). Implementing the `addHouse()` method is a separate task detailed in Task F. If a house cannot be added, show error then back to the menu.

Option 2 – Display the block
Call the `displayBlock()` method.

Option 3 – Clear the block
Call the `clearBlock()`method.

Option 4 - Quit
If the input is 4, terminate the program.

Task F
Implements the `addHouse()` method that takes four input parameters: the row position, the column position, the number of rows, and the number of columns of the house. Parameters row position and column position are the house's top left corner position. For example (2, 1) means the house starts from 2 rows down from the top and 1 column from the left edge of the block. This method must observe the follow rules:

Rule 1: If the block is empty, a house can be anywhere in the block, but not touching the edges. That means there needs to be at least one row and one column gap between the house and the four sides of the block. For example, the largest possible house in a 5 x 7 block is of size 3 x 5, with a top-left corner position (1, 1) (see below).

```
0  0  0  0  0  0  0                0  0  0  0  0  0  0
0  1  1  1  1  1  0                0  1  1  1  1  1  1
0  1  1  1  1  1  0                0  1  1  1  1  1  1
0  1  1  1  1  1  0                0  1  1  1  1  1  1
0  0  0  0  0  0  0   (VALID)      0  0  0  0  0  0  0   (INVALID)
```

Rule 2: If there is already a house or houses in the block, the new house must be one row or one column away, yet still observing Rule 1 (see below).

```
0  0  0  0  0  0  0                0  0  0  0  0  0  0
0  1  1  0  0  0  0                0  1  1  1  2  2  0
0  1  1  0  2  2  0                0  1  1  1  2  2  0
0  0  0  0  2  2  0                0  0  0  0  2  2  0
0  0  0  0  0  0  0   (VALID)      0  0  0  0  0  0  0   (INVALID)
```

Rule 3: No part of a house can go outside of the block.

Rule 4: The smallest house size is 1 x 1.

You may represent all houses with ‘1’ (No mark deduction). You can also take a challenge: representing the first house with ‘1’, the second with ‘2’, the third with ‘3’ and so on.

If any of the rules is violated, the `addHouse` method should return with no house added. Consider where and how an error message should be showed to user. You may discuss that in comments.

If no rule is violated, the `addHouse` method should change the values in the `block` array for the house(s). The value of `vacant` should be correct. After building the house, call `displayBlock()` from the `addHouse` method to show the updated block.

The following code requirements must be applied in a fashion similar to what has been shown in lesson materials.

The program must be entirely in one Java class/file `ProgFunAssignment1.java.` Other names will not be accepted.

Code formatted consistently. Tip: you would use Eclipse→Source menu→Format before every submission. Must not include any unused/irrelevant code (even inside comments); what is submitted must be considered the final product.

Use appropriate data types and handle user inputs properly e.g. using `Scanner.`

Must not have redundant conditions/parts in *else if* statements

No ArrayList or similar data structures in Assignment 1. Students must demonstrate their ability to manipulate (standard) Java arrays on their own without using external classes/libraries.

In places where this specification may not tell you how exactly you should implement a certain feature, you need to use your judgment to choose and apply the most appropriate concepts from our course materials. Follow answers given by your “client” or “supervisor” (the teaching team) under ***Canvas→Discussions→’Assignment 1’*** when in doubt.

Documentation requirements

Write comments in the same java file, before code blocks (e.g. before methods, loops, ifs, etc. where you can have { }) and important variable declarations. DO NOT write a separate file.

The comments in this assignment should serve the following purposes:

Explain your Java code in a precise but succinct manner. It should include a brief analysis of your approaches and evaluation instead of simply translating the Java code to English. For example why you choose while loop instead of other loops, how rules of adding houses are observed in your method, why you introduce a particular variable or method not specified in the assignment specification.

Document any problems of your code and requirements that you have not met. For example, situations that might cause the program to crash or behave abnormally, or ideas/attempts for completing a certain functionality. Write these in the approximate locations within your code.

No need to handle or address errors that are not covered yet in lectures, such as input type mismatches.

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 **one file** ProgFunAssignment1.java 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.

Code must be compiled under command line with no error,

```
> javac ProgFunAssignment1.java
```

and runnable under command line

```
> java ProgFunAssignemnt1
```

Submission failed to compile would NOT receive ZERO mark.

Rubric

Assessment Task	**Marks**
Task A	0.5 marks
Task B	0.5 marks
Task C	0.5 marks
Task D	0.5 marks
Task E • Menu inputs • Parameter inputs	1+1 marks
Task F • Validation on new house(s) • Updating the block	2.5 + 0.5 marks
Others 1 .Code quality and style 2 .Modularity / Use of methods & arguments 3 .Comments / Analysis/ Reflection	1 + 0.5 + 1.5 marks