**COSC2531 Programming Fundamentals**
**Semester 1 2018**
**Assignment 1**

## Introduction

This is a group assignment and worth 20% of your final grade. The grading is based on both demonstration and final code submission. Each group may consist of 2-3 students. You must demonstrate your assignment in your timetabled practical class prior to your final submission.

## Objective

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 will be awarded partial marks for explaining a valid strategy, even if the program is not working.

## Assessment

**Demonstration**: The demonstration is worth 10% of the final grade (half of the weight of Assignment 1). The whole assignment group must be present to demonstrate the assignment in a timetabled practical class (tute-lab) prior to the final submission. If members come from different classes, please choose just one of the classes to make the demonstration. Subject to on-time final submission and plagiarism detection, you will receive the preliminary marks for your demonstration. The code should be clean and have proper comments. You can choose to demonstrate either during week 6 or week 7 tute-lab, but you can do it just once.

**If you demonstrate in week 6 tute-lab, you will receive a bonus 1 mark. If you do not demonstrate by the end of week 7 you will receive marks only on the final submission code.**

**Final submission**: The final submission is worth 10% of the final grade (half of the weight of Assignment 1), due on week 7 (22 April, 2018, 23:59 pm). **Only one group member needs to make the final submission.**

Your final submission through Canvas should be a zip file that includes the following files –

1. A single java file named *LandVille.java*. Please do not submit the .class file.
2. A .txt file that contains the name and student ID of each member of your group.
3. (Optional) If you are unable to complete the code, please also submit a word document file with the brief description of problems encountered and lessons learnt. If you have not received full marks during the demonstration and then made any change in your code afterwards, submit a word document file with the brief description of changes as well.

## Assessment criteria

| Assessment Task | Marks |
|---|---|
| Task A | 1 marks |
| Task B | 1 marks |
| Task C | 1 marks |

| | |
|---|---|
| Task D | 1 marks |
| Task E<br>• Menu inputs<br>• House inputs | 2+2 marks |
| Task F<br>• House input check<br>• buildHouse method | 2+5 marks |
| **Others**<br>1. Correctness / Test cases & results<br>2. Code quality / comments<br>3. Modularity / Use of methods & arguments<br>4. Clear Logic / Strategy explained<br>5. Reflection / Lessons learnt | 5 marks |

**General Requirements**

1. Your final code submission should be clean, neat, and well-formatted (e.g., consistent indentations).
2. Identifiers should be named properly.
3. You must include adequate meaningful code-level comments in your program.
4. For each input from the user, display appropriate prompt message.
5. For each invalid input from the user, display appropriate error message.

**Assignment overview**

LandVille is a one player game, where the player can build his/her dream house in a land, but needs to follow the building rules and regulations. Your task is to write a Java code that checks if all the rules are followed, then display what the house will look like to the player. 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.

**Assessment tasks**

Your program should consist of a LandVille class that contains the main method. The class has a two dimensional array of integers called 'land'. The 'land' variable should not be accessible outside the class. The class also has a Boolean variable 'hasHouse' that stores the information of whether is an existing house in the land. The methods of this class are: displayLand(),buildHouse(), and clearLand().

Task A

The constructor of the LandVille class takes 2 parameters: the number of rows of the land, the number of columns of the land. In the constructor of this class, write code to initialize land with the size of row and column. Initialize each value of the array 'land' with the value 0, which means that plot is empty. Initialize the value of 'hasHouse' to false.

Task B

Write a method of the LandVille class called ‘displayLand()’ that prints the land 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 displayLand() is in the following, where the number of rows is 3 and columns is 4.

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

Task C

Write a method of the LandVille class called ‘clearLand()’ that sets the value of each element of the array ‘land’ to zero (0). Set the value of ‘hasHouse’ to false.

Task D

From the main method, ask the player for the row and column of the land. The number of rows and the number of columns should be greater than 0 and less than or equal to 10. If any input is not correct, show an error message and ask for that input again.

If all inputs are correct, create an object of LandVille 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:

1. Build a house
2. Display land
3. Clear the land
4. Quit

Take the input option as an integer. As long as the input is not between 1 to 4, show the menu message and ask for input again. If the input is 4, terminate the program. If the input is 2, you need to call the displayLand() method of LandVille class (displayLand method is described in Task B). If the input is 3, call the clearLand() of LandVille class.

If the input is 1, you need to check first if there is already a house in the land. If there is a house, show an error message and show the menu again. Otherwise, call the buildHouse() method of LandVille class. Method buildHouse() of the LandVille class takes two input parameters: number of rows of the house to build, and the number of columns of the house to build. From the main method, ask for these parameter values as input, then call the buildHouse method with these inputs.

After the buildHouse is executed (details in task E), the program should keep showing the menu and ask for input from the menu.

Task F

Write a method buildHouse() of the LandVille class that takes two input parameters: number of rows of the house to build, and the number of columns of the house to build.

You need to build the house at the top-left corner of the land (i.e., from row 0 and column 0), with borders of width 1 on each side. The position of house is represented with the integer '8' and the border is represented with '1'.

To make sure that the house with the border on each side will fit in the land, the difference between the row of the house and the row of that land must be greater than or equal to 2. Similarly, the difference between the column of the house and the column of that land must also be greater than or equal to 2. If any of the input parameters do not satisfy the condition, show an error message and then return from this method. As an example, let the row number of a land is 7 and column number is 6. The row number of the house is 3 and column number is 5. In this case, do not make any change to the 'land' array, show an error message, and return from the buildHouse method.

If all the input parameters satisfy the condition, you need to set the values of appropriate elements of the 'land' array with '1' for border and '8' for the house. As an example, let the row number of a land is 7 and column number is 6. The row number of the house is 3 and column number is 2. The land array will look like the following –

```
1 1 1 1 0 0
1 8 8 1 0 0
1 8 8 1 0 0
1 8 8 1 0 0
1 1 1 1 0 0
0 0 0 0 0 0
0 0 0 0 0 0
```

As another example, let the row number of a land is 5 and column number is 4. The row number of the house is 1 and column number is 2. The land array will look like the following

```
1 1 1 1
1 8 8 1
1 1 1 1
0 0 0 0
0 0 0 0
```

Write code of buildHouse method to change the value of the array 'land' for the house and border. Set the value of 'hasHouse' to True. After building the house, call displayLand() from the buildHouse method.