COSC2531 Programming Fundamentals

Assignment 1

Introduction

This is a group or individual assignment and worth 20% of your final grade. The grading is based on both a demo and final code submission. Students can work on their own, or in groups from 2-3 students. You must demonstrate your assignment in your timetabled practical class prior to your final submission in week 6.

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 demo is worth 5 marks out of a total of 20 marks and will be done in week 6 tutorial classes before the mid-semester break. The demo will require you to show your code for tasks A, B, C and D and your tutor will provide you feedback so that you can finish the assignment over the holidays.

If you are working as a group and your 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.

Marking Priority

Students will only be marked in the tute-lab they have enrolled in. If a group delays their marking to the latest day possible then they run the real risk of the tutor running out of time and their assignment being marked late. Students are expected to have their work completely ready for assessment when the tutor gets to them – if you are not ready you will be placed back at the end of the marking queue.

Final submission: The final submission is worth 20 marks or 20%, 5% for demo and 15% for submission. The assignment is now due on the **7th of September at 11:59pm**. ***Only one group member needs to make the final submission.***

Penalties will apply for late submission of assignment after this date.

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

1. *LandVille.java and LandVilleMain.java*. Please do not submit the .class file.

2. A members.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 or pdf 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 or pdf document file with the brief description of changes as well.

Assessment criteria

Assessment Task	**Marks**
Task A, B, C, D	Maximum 5 marks for demo (week 6)
Task A, B, C, D, E, F	Maximum 15 marks for submission (due Sept 7th)
Others	You will be marked on 1.Correctness/Results 2.Code quality/comments 3.Modularity/Use of methods & arguments 4.Clear Logic/Efficient code 5.Lessons learnt (either shown in the demo or word doc if you were unable to complete the assignment)

General Requirements

1. Your final code submission should be clean, neat, and well-formatted (e.g., consistent indentations) and abide by the formatting guidelines.
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 program 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 class also has a Boolean variable `hasHouse` that stores the information of whether a house exists 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 shown below, 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 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.

Option 1 – Build a house

Check if there is already a house in the land. If there is, show an error.

Otherwise, prompt the user of the number of rows and columns of the house to build and then call the `buildHouse()` method of the `LandVille` class with those values as parameters (in that order). For Task E this method can be empty – that is it doesn't need to contain any code inside the main `{}`.

Option 2 – Display Land

Call the `displayLand()` method.

Option 3 – Clear Land

Call the `clearLand()` method.

Option 4 - Quit

If the input is 4, terminate the program.

For all inputs except 4 you should loop back and display the menu again.

Task F

Write a method `buildHouse`() that takes two input parameters: number of rows and 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
```

The `buildHouse` method should change the values in the `land` array for the house and border. Set the value of `hasHouse` to True. After building the house, call `displayLand`() from the `buildHouse` method.