Unit:
Designing and Developing Object-Oriented Computer Programs

Assignment title:
Sonar Sweeper

December 2015 – Sample Assignment

Important notes

- Please refer to the Assignment Presentation Requirements for advice on how to set out your assignment. These can be found on the NCC Education *Campus*. Click on Policies and Advice in the left-hand menu and look under the Advice section.
- You must read the NCC Education documents 'What is Academic Misconduct? Guidance for Candidates' and 'Avoiding Plagiarism and Collusion: Guidance for Candidates' and ensure that you acknowledge all the sources that you use in your work. These documents are available on *Campus*. Click on Policies and Advice in the left-hand menu and look under the Policies section.
- You **must** complete the **'Statement and Confirmation of Own Work'**. The form is available on *Campus*. Click on Policies and Advice in the left-hand menu and look under the Policies section.
- Please make a note of the recommended word count. You could lose marks if you write 10% more or less than this.
- You must submit a paper copy and digital copy (on disk or similarly acceptable medium). Media containing viruses, or media that cannot be run directly, will result in a fail grade being awarded for this assessment.
- All electronic media will be checked for plagiarism.

Introduction

Minesweeper is one of the most popular video games in the world. For decades, it was installed on every Microsoft Windows system. If you have not played the game, you can try it out here: http://minesweeperonline.com/

The rules of the game are simple – the player's job is to clear a grid of hidden 'mines' without detonating any of them. The game starts with a grid, with each space representing a hidden cell of the game. Upon clicking a cell, the game uncovers each of the cells which are not adjacent to a hidden mine. This is a recursive process, where each empty cell will also reveal adjacent empty cells.

Each of the non-empty cells discovered as part of this recursion display a number showing how many mines to which they are adjacent. Players can choose to 'flag' a cell to indicate the suspected presence of a mine – they get a number of flags equal to the number of mines in the grid. The game is finally completed when all empty squares have been uncovered.

Your task for this assignment is to write the game minesweeper. However, your version of minesweeper will have one key difference from the normal game – you will **NOT** show the number of adjacent mines on a square.

Your version of minesweeper is called *Sonar Sweeper*. It should include this modification. When a player places a mouse over a cell, your game will emit a number of 'sonar' beeps equal to the number of mines to which it is adjacent. When moving your mouse over a cell, a different sound should play. As a result, it should be possible to play a fully functioning game of 'Sonar Sweeper' with your eyes closed.

In *Sonar Sweeper*, you **should** allow the player to choose from one of three sizes of maps:

Length	Width	Number of Mines
9	9	10
16	16	40
24	24	99

The user interface for this game should consist of a 2D grid of buttons. Left clicking should 'explode' a cell and reveal an empty space around it or indicate to the player they have lost if they explode a cell that contains a mine. Right clicking should flag it as having a mine.

It is important that you also include this requirement. The first cell that a player clicks is always empty – the mine locations are generated **after** the player has made their first move.

Tasks are on next page

Task 1 – The Application (50 Marks)

The program should fully meet the requirements of the brief as outlined above. The program algorithms should make effective use of all available tools and should involve functions, loops, selection classes, objects and either array or string manipulation. 10 marks are available for **each** of the following: (1) Appropriate use of objects; (2) Handling user interaction; (3) Minesweeper setup; (4) Uncovering cells; and (5) Encapsulation and Abstraction.

Task 2 – Testing Data (25 Marks)

Testing data should be sufficient to provide suitable coverage of all equivalence classes, and should use black box and white box testing to explore each function. The marks for this task are broken down as follows: (1) 10 marks for developing a test plan; (2) 10 marks for implementing a test plan; and (3) 5 marks for making effective use of exception handling.

Task 3 – Design Documentation (25 Marks)

A fully detailed UML diagram of their classes should be submitted. The marks for this task are broken down as follows: (1) 5 marks for class relationships; and (2) 20 marks for methods and attributes.

Submission requirements

- Your program must be submitted as a zip file of the full project.
    - Whatever IDE you use, it should be possible to open and run the project directly from the extracted archive.
- Your testing data must be accompanied with a short, 100 word discussion of how the data was selected and executed.

Candidate checklist

Please use the following checklist to ensure that your work is ready for submission.

Have you read the NCC Education documents 'What is Academic Misconduct? Guidance for Candidates' and 'Avoiding Plagiarism and Collusion: Guidance for Candidates' and ensured that you have acknowledged all the sources that you have used in your work?	❑
Have you completed the 'Statement and Confirmation of Own Work' form and attached it to your assignment? **You must do this.**	❑
Have you ensured that your work has not gone over or under the recommended word count by more than 10%?	❑
Have you ensured that your work does not contain viruses and can be run directly?	❑