Unit:
Designing and Developing Object-Oriented Computer Programs

Assignment title:
Battleslugs

March 2016

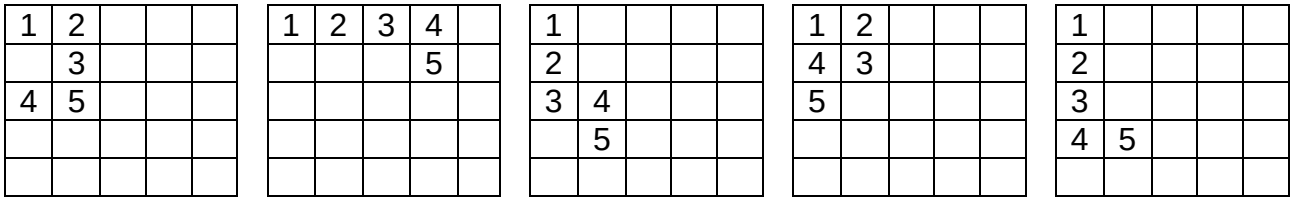
Important notes

- Please refer to the *Assignment Presentation Requirements* for advice on how to set out your assignment. These can be found on the NCC Education website. Click on 'Policies & Advice' on the main menu and then click on 'Student Support'.
- 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 the NCC Education website. Click on 'Policies & Advice' on the main menu and then click on 'Student Support'.
- You **must** complete the *Statement and Confirmation of Own Work*. The form is available on the NCC Education website. Click on 'Policies & Advice' on the main menu and then click on 'Student Support'.
- 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

Slugs Incorporated is a game developer specialising in mollusc themed games for multiple platforms. Most of their games are twists on existing classic titles, and they have put together a proposal for a game they have called *Battleslugs*. You have been to develop a prototype of the game *Battleslugs*.

Battleslugs is very similar to the popular game Battleships. But, there is one significant difference between the two games. In *Battleslugs*, there are no fixed shapes for each of the two players. Instead, each player gets five 'slugs' to place on a 12x12 board. Each of these slugs consists of five segments, but the arrangement of the segments is randomised for each game. The only rule is that each segment of the slug must be within an eight cell neighbourhood of the previous segment. The following examples are all valid slug configurations:



The game progresses as follows:

1. Player one gets a set of five randomly generated slugs.
2. Player one places each in turn on the 12x12 grid.
    a. The game should not let the player place a slug if it will overlap another.
3. Player two gets a set of five randomly generated slugs.
4. Player two places each in turn on the 12x12 grid.
    a. The game should not let the player place a slug if it will overlap another.
5. The game begins, with each player taking a turn to fire salt at their enemy.
    a. The player selects the grid co-ordinate of a square on the other player's grid.
    b. If they select a square which contains a slug, that segment of the slug is killed.
6. The game continues until all the slug segments of one player have been killed.

On each player's turn they should see a display of where they have previously targeted on the opponent's grid, and whether it was a hit or a miss. It should also provide a button that allows them to see their own grid, along with the state of each of their own slugs.

When the game is over, a win screen should be presented that shows: (1) the player who won; (2) how many shots each player took; (3) how many of each player's shots were hits; and (4) the hit percentage of each player.

Your program will need to perform the following operations:

- Set up the GUI
- Set up a program loop to each of the player turns.
- Randomly generate slug shapes for each player.
- Allow the player to place a slug, making sure it doesn't overlap any other.
- Allow each player to select a square on their opponent's grid to attack.

- Keep track of which slug segments have been killed on each grid.
- Identify the win state and inform the players when it occurs.

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 the available tools and should involve functions, loops, selection classes, objects and either array or string manipulation. 10 marks are available for the following: (1) Appropriate use of objects; (2) Handling user interaction; (3) Slugs setup; (4) Game Flow; 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 – 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? ❑