Unit:
Designing and Developing Object-Oriented Computer Programs

Assignment title:
Sonar Sweeper

December 2015 – Sample Assignment

Marking Scheme

Markers are advised that, unless a task specifies that an answer be provided in a particular form, then an answer that is correct (factually or in practical terms) **must** be given the available marks. If there is doubt as to the correctness of an answer, the relevant NCC Education materials should be the first authority.

This marking scheme has been prepared as a **guide only** to markers and there will frequently be many alternative responses which will provide a valid answer.

Each candidate's script must be fully annotated with the marker's comments (where applicable) and the marks allocated for each part of the tasks.

Throughout the marking, please credit any valid alternative point.

Where markers award half marks in any part of a task, they should ensure that the total mark recorded for the task is rounded up to a whole mark.

Task	Guide	Maximum Marks
1	**The Application** The program algorithms should make effective use of all the tools students have available - at a bare minimum, it should involve functions, loops, selections classes, objects, and either array or string manipulation. The 50 marks for this section are broken down as follows: ***Appropriate use of objects*** There should be at least the following classes: Grid, GUI, Player, and Upgrade (3 marks each up to a maximum of 9 marks). 1 additional mark is available for any other sensible classes used to improve architecture. ***Handling User Interaction*** The program should set up a GUI made up of a 2D array of buttons (2 marks). Buttons that have been recursively identified as being empty should be marked as disabled (2 marks). Clicking a button should recursively identify empty cells (2 marks). Moving a mouse over an empty cell should play a sound (2 marks). Moving a mouse over a cell adjacent to a mine should play a number of 'sonar' beeps equal to the number of mines underneath (2 marks) ***Minesweeper Setup*** The game should allow for the player to choose the size of map (2 marks) and then set up the underlying 2D array (2 marks) to have the correct size (2 marks) and correct number of mines (2 marks) after the player has chosen the first location (2 marks). ***Uncovering Cells*** Upon detonating a cell, the game should identify all neighbouring cells (2 marks) and recursively (2 marks) disable empty cells (2 marks), terminating when a cell is not empty (2 marks) and updating the interface accordingly (2 marks) ***Encapsulation and Abstraction*** The implementation should have classes appropriately encapsulated, with internal fields set as private (2 marks), and accessor methods provided for each (2 marks). Classes should come with appropriate constructors (2 marks). The system as a whole should show an appropriate amount of coupling (2 marks) and cohesion (2 marks).	**10** **10** **10** **10** **10** **50**
2	**Testing Data** 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 25 marks allocated to this section of the coursework is broken down as follows: ***Develop a test plan*** The task plan should incorporate both white box (2 marks) and black box (2 marks) testing. It should incorporate equivalence cases (2 marks) and boundary checking (2 marks)**.** It should also incorporate a short report discussing the testing data and any regression testing that was required as a result of errors encountered (2 marks).	**10**

Task	Guide	Maximum Marks
	Implement test Plan The report should include a full log of the result of user testing (2 marks)**,** including tables of test data versus actual and expected results (2 marks)**.** Units should be tested in isolation (4 marks) and then integrated (2 marks)**.** ***Making effective use of exception handling*** When an exception can be thrown during the program's operation, it should be caught and handled appropriately. Testing data should identify where there are potential exceptions to be thrown (2 marks) and the code should provide the appropriate structures for dealing with it (3 marks).	**10** **5** **25**
3	**Design Documentation** Students should submit a fully detailed UML diagram of their classes. These should include relationships between classes as well as the attributes and methods that each class exposes. ***Class relationships*** Class relationships should be documented in terms of the nature of their relationship (3 marks) and their cardinality (2 marks) ***Methods and Attributes*** Each attribute of each class must be given (5 marks) along with their visibility and type (5 marks)**.** Similarly, each method must be given (5 marks) along with visibility, return type and parameters (5 marks)	 **5** **20** **25**
		Total: 100 Marks

Learning Outcomes matrix

Task	Learning Outcomes assessed	Marker can differentiate between varying levels of achievement
1	1, 2, 3	Yes
2	1, 4	Yes
3	2, 5	Yes

Grade descriptors

Learning Outcome	Pass	Merit	Distinction
Design object-oriented programmes to address loosely-defined problems	Provide adequate design to address the specification	Provide detailed and appropriate design to address the specification	Provide wholly appropriate and innovative design that meets the specification
Implement object-oriented programmes from well-defined specifications	Provide adequate design to address the specification	Provide detailed and appropriate design to address the specification	Provide wholly appropriate and innovative design that meets the specification
Develop object-oriented programmes that reflect established programming and software engineering practice	Show adequate development	Show sound and appropriate development	Show innovative and highly appropriate development
Develop test strategies and apply these to object-oriented programmes	Show adequate development and application of testing strategies	Show sound and appropriate development and application of testing strategies	Show innovative and highly appropriate development and application of testing strategies
Develop design documentation for use in program maintenance and end-user documentation	Show adequate development of materials	Show sound and appropriate development of materials	Show innovative and highly appropriate development of materials