FIT1002

Assignment 2 – Part 1

This document specifies the requirements of part 1 of assignment 2. Part 2 (to be posted soon) requires the use of arrays which will be covered in week 10 lectures.

The assignment is marked out of 100. Part 1 is worth 75 marks and part 2 is worth the remaining 25 marks.

Assignment 2 (part 1 and part 2) contributes 15% of your final mark for FIT1002

Submission Instructions:

Will be included in the specification of requirements for part 2.

Assignment Objectives:

The purpose of this assignment is to test the following:

- Use of Java API classes (Scanner, String, Math, etc.)
- Use of control structures (selection and repetition, nested and unested) to create non-trivial logic
- Design of object oriented solutions to simple problems using multiple user-defined classes
- Testing of programming solutions
- Use of arrays (Part 2)

Assessment Criteria:

Your assignment will be assessed using the following broad criteria:

- Are all functional requirements implemented correctly i.e. does the program actually work?
- What is the quality of the solution? Includes logical clarity and code maintainability.
- Are good coding standards followed?
- Is the code sufficiently and appropriately documented?
- Has a methodical and thorough test process been followed?

Assessment will be based on evaluation of submitted code and any other required submitted documents. In addition students will be interviewed to test their understanding of their own code.

Assignment Overview:

For this assignment you will be creating a grid-based game in which a hunter attempts to shoot a snark. The snark is a vicious beast that lives in the dismal swamp beyond the limits of the city.

As a game begins the snark hides at a random grid point. At each step of the hunt the hunter takes a shot by indicating a grid point expressed as an x and y coordinate. The origin (0, 0) is in the bottom left hand corner of the grid. For a grid of size 10 the top right hand corner is (9, 9). If the grid point of the hunter's shot is the same as the snark's hiding place the snark is dead and the game is over and the hunter wins. The hunter has 5 shots to kill the snark. If the hunter fails to shoot the snark in 5 shots or gives up before all 5 shots have been used the game ends and the snark wins. After each unsuccessful shot the hunter is told where the snark is relative to the shot using compass directions as follows:

North	Above	Higher y coordinate
South	Below	Lower y coordinate
East	right	Higher x coordinate
West	left	Lower x coordinate

Details :

There is a console dump of a short but typical run of the application in the file called dump.txt. In this dump two games are played. In the first game the snark is shot dead which is a score for the hunter. In the second game the hunter gives up before 5 tries which is a score for the snark. In the dump the position of the snark is indicated by an "S". This is just for testing and to make the dump clear. Obviously the final version of the application would suppress this indicator of the snark's position. The position of the shot is indicated with an "*" and a successful shot (i.e. a hit) by an "X" (see first game). Although your application is not required to reproduce the dump format exactly it should produce all the features of the dump.

The following points give more details:

1. At the start of the application (before the first game) the application's title is displayed followed by some brief instructions on how to play a game.
2. At the start of each game an empty grid is displayed before the coordinates of the first shot of the game are prompted for.
3. The Snark is randomly located somewhere within the swamp at the start of each game.
4. In each game the hunter has five shots. The number of shots remaining at any time is displayed.
5. A successful hit should produce an appropriate message and end the current game and increment the hunter's score.

6. An unsuccessful shot should produce hints as to the relative compass position of the shot and the snark.
7. An unsuccessful fifth shot should end the current game and increment the snark's score.
8. A user response of -1 before or at the fifth shot should end the current game and increment the snark's score.
9. If the player chooses to play another game at any stage all relevant game variables should be reset and a new game played. There is no need to print the application title and brief game instructions for games other than the first game.
10. If the player chooses not to play another game at any stage the scores of both the hunter and the snark accumulated over the current run of the application should be displayed along with a goodbye message.

Assignment Tasks:

Part 1: Program Design [10 marks]

As a first task read the whole assignment specification and draw a class diagram for each of the required classes. This helps develop an application's design before plunging into the complexities of coding.

Part 2: Writing the Hunter and snark classes Class [10 marks]

You can write individual classes for the Snark and the Hunter classes or write a single class called Player that is used to instantiate a snark Player instance and a hunter Player instance. The Hunter class abstracts the hunter's shot. You can call it Shot if you prefer.

Snark and Hunter instances (or Player instances) have a name and an x and y coordinate.

You should include in these classes appropriate attributes, constructors, accessors and mutators. The constructors should set any instance variables they do not initialise with their parameters to sensible default values. You should also include a toString() method.

You must write a test driver to test your Snark class. Test its attributes and methods as appropriate. The driver class is part of the assignment's required deliverables.

Part 3: Writing the Swamp Class [20 marks]

The Swamp class has a size (if size = n the swamp is represented by n X n grid of locations), a hunter and a snark.

Write a Swamp class including appropriate attributes, constructor, accessors and mutators. You should also include a toString() method or equivalent.

The constructor should take as its single parameter the Swamp's grid size and it should also instantiate its hunter and snark.

A Swamp instance also performs several important services for the Game driver class including:

- Randomly positioning its snark at the start of each game
- Setting the location of its hunter's shot at each step of a game
- Determining if its snark has been hit
- Reporting on the relative position of its hunter's shot and its snark
- Displaying its grid

You must write a test driver to test your Swamp class. Test its attributes and methods as appropriate. The driver class is part of the assignment's required deliverables.

Part 4: Writing the Game Class [35 marks]

Write the Game class to implement all the basic program logic in its main method. Where possible simplify this code by calling helper methods to gets main's task performed.

continued …

Student Guide to Grading Assignment 2:

Note: Only a grade will be released for assignment 2 along with specific comments that indicate where you need to improve.

The following guide is to be used in conjunction with the marking scheme indicated for each part of the assignment previously described in this document.

To obtain a **Pass**, you must demonstrate the following abilities:

- Draw class diagrams of user-defined classes
- Code and instantiate user-defined classes
- Implement simple conditions and control structures
- Implement user input using the Scanner class

To obtain a **Credit**, you must achieve all that was required for a Pass and additionally demonstrate the following abilities:

- Nest control structures to implement required logic
- Use simple methods inside a class
- Use driver classes to test the user-defined classes

To obtain a **Distinction**, you must achieve all that was required for a Credit and additionally demonstrate the following abilities:

- Use of methods with parameters
- Use of methods with return values
- Use of methods to modularise code
- Implement advanced program features (see part 2 of assignment 2 when it is posted)

To obtain a **High Distinction**, you must achieve all that was required for a Distinction and additionally demonstrate the following abilities:

- Implement advanced program features (see part 2 of assignment 2 when it is posted)