COMP285: Computer Aided Software Development
Assignment 2 2016/2017

This is the second of two assessments for COMP285 which contributes 50% of the final module mark.

OBJECTIVE

This coursework involves the development and testing of a game in Java™ using the software tools, Eclipse and JUnit.

Assessment Information

Assignment number	2 of 2
Weighting	50%
Assignment Circulated date provided to class	**2/3/2017**
Deadline Day & Date & Time	Friday, May 5, 16:00 2017
Submission Mode	Electronic
Learning outcome assessed	1. Perform software development tasks using the techniques of Automated Testing, Continuous Integration and Test Driven Programming 2. Use Ant, JUnit and Eclipse both individually and jointly as tools for Automated Testing, Continuous Integration and Test Driven Programming
Submission necessary in order to satisfy Module requirements	No
Purpose of assessment	To assess the students ability to effectively use software development and testing tools
Marking criteria	See end of document

This is the second of two assessments for COMP285 where each contributes 50% of the final module mark.

The general purpose is assessment of the following learning outcomes:

1. Perform software development tasks using the techniques of Automated Testing, Continuous Integration and Test Driven Programming
2. Use Ant, JUnit and Eclipse both individually and jointly as tools for Automated Testing, Continuous Integration and Test Driven Programming with the *main stress* on JUnit testing and test driven programming with Eclipse.

The goal of this assignment is working on an initial (partial) implementation of a game *"Hangman"* in the style of *Test Driven Programming* in Java by using Unit Testing with JUnit4 in the framework of Eclipse. Two compatible Java files are given to you: a source file Hangman.java and JUnit4 test case HangmanTest.java, both implemented partially and Hangman.java also containing some bugs.

You will need to finish/correct this implementation and testing framework. All tests should succeed and the game should be functioning well. The game *"Hangman"* is described as follows:

*The computer chooses a word (the "word-to-be-guessed"), and displays a star sign * for each letter in the word (the *-form of the word).*
*The user inputs a letter; if that letter occurs in the word, then every occurrence of that letter in the word is shown (i.e., the letter is written instead of the *).*

If the letter does not occur in the word or is chosen repeatedly, the user loses a life. Play continues in this way until either
- all the letters in the answer have been input and the user has won, or
- the user has lost all their lives (10) and the computer has won.
After each round, the user has the option of quitting or playing another round. If quitting, the number of rounds played is shown and the round numbers are displayed which the user has won.

You can download the (correctly implemented compiled version of) Java class MyHangman.class into a directory H:\dir and play it by running from the command line as follows:

H:\dir>java MyHangman

It has a slightly different name. You will work with Hangman.java, HangmanTest.java and their compiled classes.

Presentation of work

In Eclipse, create a new project with the project name (directory) of the form the student's
surname_name_ID

Similarly, at the beginning of the two .java files given you should present your personal data, plus personal e-mail address where the feedback will sent.

For simplicity, *when creating this project in Eclipse*, choose the option *"Use project folder as root for source and class files"*. The source files and compiled files will be under the same project directory named as described above. Also for simplicity, this time *no package declaration* should be used in .java files (although Eclipse discourages this choice).

All the actions such as creating the project class source file, JUnit test case, etc.) should be done by Eclipse wizards.

After creating Hangman.java, it should be populated by the code above.

After finishing the these steps, the JUnit 4 test case HangmanTest.java should be created similarly by using an appropriate the Test Case wizard of Eclipse. While using this wizard appropriate setUp and tearDown methods should be chosen, then after clicking NEXT and checking all (11) methods of the source file HangmanTest.java, *except* the main method, 11 stub test methods will be generated automatically. The wizard will also suggest adding JUnit 4 library to the build path. You should include this otherwise it will be impossible to run JUnit test case from Eclipse. After the stub class HangmanTest.java is generated, it should be populated by the code given above, again with your personal data.

Two MSWord files Hangman.doc and HangmanTest.doc containing the codes of Hangman.java and HangmanTest.java should be created (directly from the physical workspace). These files will be used by the lecturer for coloured/highlighted comments and marking.

The main feature of this program (a game) is user input from the console. The problem is how to test it. To this end, imitation of the sequence of input strings by the user is done by a Vector <String> INPUT and a *mock* method input_next() imitating the standard one *input*.next() using
Scanner *input* = new Scanner(System.*in*);
This should be clearly understood before doing anything. See examples of how it works in the current versions of both files Hangman.java and HangmanTest.java.

The general advice is to work first on something simple. Then there will be fewer problems since you can then depend on what you have already resolved and well-tested.
You should *comment* appropriately all the code you add to the sample code. This way you can demonstrate your understanding of what they do. This will also be taken into account when marking the work.

All Java files presented by students should be *compilable* and appropriately *formatted.*

To submit this work, the following should be done:
Go to your physical workspace by using Windows Explorer.
Right-click your project name surname_name_ID, choose send to Compressed (zipped) folder (with the same name). The resulting zip file surname_name_ID.zip will automatically contain your project directory and all it contents.

All the auxiliary files and one auxiliary directory created by Eclipse should be present (since the submitted project surname_name_ID and its files will be imported by Eclipse to run in Eclipse and mark them). The compiled files Hangman.class and HangmanTest.class should be up to date.

This zip file should be submitted electronically by the deadline mentioned above via *Coursework Submission System:*

http://www.csc.liv.ac.uk/cgi-bin/submit.pl
http://www.csc.liv.ac.uk/teaching/modules/year2s2/comp285.html

by going to link Assignment Submission.

Late submissions policy and Plagiarism/collusion guidelines are described in Student's Handbook, http://www.csc.liv.ac.uk/student/ugpgfhandbook.pdf, Sections, 5.8-5.10.

Marking Schema
This assignment contributes 50% of the final mark for this module, and will be marked according to the marking scheme which can be found in both java files Hangman.java and HangmanTest.java with wording like "Costs 7 marks". The maximum total mark is 100. Any failures in fulfilling the above requirements which cannot be taken into account by this marking scheme (such as working not in Eclipse, etc.) can be penalised by up to 10 marks.

COMP285 CW2