Software Design and Development Coursework

Version: 20201008

Part 1: Module coupling and cohesion

For this part of the coursework, you are to analyse a program in terms of module coupling and cohesion. The output of this part is a short report containing extracts of source code along with descriptive writing.

Instructions

Select one of the example programs you have seen in the degree so far. It can be from any of the modules you are taking now or have taken before. Carry out the following tasks:

* Describe the purpose of the program and which technologies it uses.

* Examine the code and identify two different types of module coupling that are happening in the code. In your report, put in extracts of the code which illustrate the coupling and explain why you think that type of coupling is happening there.

* Examine the code and identify two different types of module cohesion that are happening in the code. In your report, put in extracts of the code which illustrate the cohesion and explain why you think that type of coupling is happening there.

Here is a checklist for part 1:

	Done?	Marks
Typed report in PDF format, approximately 500 words. Word count does not include source code extracts		1
Description of the purpose of the program and the technology it uses		1
Module coupling example 1: code extract and explanation		2
Module coupling example 2: code extract and explanation		2
Module cohesion example 1: code extract and explanation		2
Module cohesion example 2: code extract and explanation		2
Total		10

Part 2: Unit testing activity

In this part of the coursework, you are to develop a set of unit tests for one of the example programs you have seen in the course. The output of this part is some source code and a testing report.

Instructions

Select **ONE** of the unit test example programs you have seen in the course: snakestats, terriblefall or norestforthewiccad. We recommend that you select the one you feel most confident working on.

* Write a total of 9 tests, organised into three sets.
* Each test set should target one functional element of the program (e.g. an API path, a statistical function or a physics simulation function).
* Write the code that allows the program to pass your tests.
* Explain your approach in each test set. What was your strategy here?
* Capture screenshots or console logs of the code before and after it passed the tests, showing the output of the unit testing library. Shorten the output to the most important part, where it lists the number of tests passed and failed.
* Write everything up in a report which you should save out to PDF format

Here is a checklist you can use to ensure you have completed all the tasks:

	Done?	Marks
Select example program to test		0
Test set 1 target function name (e.g. /spells)		0
Test set 1 written explanation of strategy		1
Test set 1 test 1 coded, explained and run		0.5
Test set 1 test 2 coded, explained and run		0.5
Test set 1 test 3 coded, explained and run		0.5
Test set 2 target function name		0
Test set 2 written explanation of strategy		1
Test set 2 test 1 coded, explained and run		0.5
Test set 2 test 2 coded, explained and run		0.5
Test set 2 test 3 coded, explained and run		0.5
Test set 3 target function name		0
Test set 3 written explanation of strategy		1
Test set 3 test 1 coded, explained and run		0.5
Test set 3 test 2 coded, explained and run		0.5

Test set 3 test 3 coded, explained and run		0.5
Annotated output of running 9 tests before and after passing		0.5
Source code of all tests in a zip file (not a rar/7z, etc.)		1
Report in PDF format containing description of test strategy in each test set and annotated unit test output before and after passing, as described above. Aim for around 500 words.		1
Total		10

Part 3: Secure programming

For this part, you are to develop a strategy to improve the security of a piece of software using secure programming techniques. The output of this part is a report in PDF format.

Instructions

Select one of the example programs you have seen in the degree so far. It can be from any of the modules you are taking now or have taken before.

Referring to the strategies in ‘ Secure Programming HOWTO,2015 David A. Wheeler v3.72’ https://web.archive.org/web/20200816000912/https://dwheeler.com/secure-programs/, describe **THREE** things you can do to the code in that program to improve its security. Report your secure programming plan. Note that you do not need to actually implement the secure code, just report on what you plan to do.

Here is a checklist for part 3:

	Done?	Marks
Secure programming recommendation 1: what is the problem? What should change and how?		3
Secure programming recommendation 2: what is the problem? What should change and how?		3
Secure programming recommendation 3: what is the problem? What should change and how?		3
Report in PDF format containing secure programming recommendations, as described above. Aim for 500 words.		1
Total		10

What to submit

You should submit the following:

Part 1: PDF report, approx 500 words.

Part 2: Source code for unit tests in a zip file and PDF report, approx 500 words.

Part 3: PDF report, approx 500 words.