**Te Hoe Rorohiko**

**Department of Enterprise and Digital Innovation**

**Bachelor of Information and Communication Technologies**

**Best Programming Practices in .NET**

**BCPR283**

**Assignment 1 Part A Practical Project**

Semester 2 2019

Due date: Wednesday 18 September

Time: 19:00

Instructions:

Create and provide supporting documentation for a <<MODEL>> for running Sudoku as described in this document.

**TOTAL MARKS: 100**

Student Name/ID

*Ara Institute of Canterbury and its division members reserve the right to use electronic means to detect and help prevent plagiarism. Students agree that when submitting this assignment, it may be subject to submission for textual similarity review to Turnitin.com.*

Submissions received late will be subject to a penalty of 10% of the student’s mark per working day.

This assignment is worth 15% of the total marks for this course.

This paper has four(4) pages including the cover sheet.

**OVERVIEW**

You are to create the first part of a program that will enable a user to solve a Sudoku puzzle. This will be written in C#. Naturally you will be expected to use good design, programming and testing practices. For an overview of Sudoku, try the puzzles at [www.websudoku.com](http://www.websudoku.com) , or use Google.

**OUTSIDE THE SCOPE**

* Creating puzzles programmatically
* Solving the puzzles programmatically
* Checking if a given puzzle state can be solved

**PROVIDED FILES**

You have been provided with a set of Interface files as below:

* IGame for initialization purposes
* IGet which retrieves a cell value from a game board in various ways
* ISet which sets a cell value into a game board in various ways
* ISerialize which reads/writes values for a whole game board and implies knowledge of the way the data is stored, wither in CSV or in an appropriate class.

You MUST use these.

**REQUIRED STRUCTURES**

* Game will be serialized into a CSV format. You can decide (and document) this format
* The game board must be stored as a 1-d array
* The row, column and square numbering is 0-based.

**DELIVERABLES PART A**

You need to design, write and test a <<MODEL>> that will be able to be used later in association with a <<VIEW>>.

This means you need to:

1. Develop appropriate UML diagrams (including at least a class diagram)
2. Decide on the features that your program will have
3. Develop appropriate tests
4. Code and test

You may work in groups for creating your code, but must make individual presentations. Clearly comment the source of all code that is not your own. You may use code provided by others in the class but will only be marked on code you have written.

**NOT DELIVERABLE**

* Views and Controllers, although you may wish to create some to assist with your testing
* Serialization to disk. In particular, you do not need to load a file from disk, but you are expected to implement as if you had a file loaded into memory as a string.

**HAND IN**

* A 10-minute MAXIMUM (optionally narrated) PowerPoint explaining how much of the <<MODEL>> you have implemented.
* A digital copy of the project code including unit testing code
* A digital copy of all the analysis and design documentation used

The presentation should cover three issues:

1. Analysis and design of the <<MODEL>> classes you have implemented. This includes at least:

* UML Class diagram(s)
* Diagrams appropriate to the type of program you are creating.

1. A list of features of the <<MODEL>> prioritized using MoSCoW (Must have, Should have, Could have, and Won’t have). Examples of a feature would be:

* A list of all the possible values for a particular cell
* The ability to convert a partially completed board to CSV format for storage
* The ability to reset the board to it’s initial state
* A count of the number of cells that do not have a valid value in them if the rest of the board is valid

1. How many of the unit tests which prove the correctness of the <<Model>> your code pass? In your presentation make clear how fully the tests you use actually test the functionality of the <<MODEL>>.

MARKING RUBRIC

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Element | 0 | 1 | 2 | 3 | 4 | 5 | Mark |
| Class diagram | Not attempted | Analysis level diagram | Design level diagram of data structures | Design level diagram of structures and attributes | Design level diagram of structures, attributes and methods | Design level diagram of structures, attributes, methods, and interfaces |  |
| Dynamic UML diagrams, Structure diagrams | Not attempted | 1 diagram | 2 diagrams | 3 diagrams | 4 diagrams | >4 diagrams Correct UML |  |
| Documentation / Define scope of first iteration MoSCoW | Not attempted | 5 scoped features | 6-10 scoped features | 10-15 scoped features | 15-20 scoped features | >20 scoped MUST features | NOTE: %IMPLEMENTED will be reduced for incomplete **feature** coverage |
| % IMPLEMENTED requirements | Not attempted | All features implemented, plus correct corresponding diagrams and documents | All features implemented, plus correct corresponding diagrams and documents | All features implemented, plus correct corresponding diagrams and documents | All features implemented, plus correct corresponding diagrams and documents | All features implemented, plus correct corresponding diagrams and documents | \* 2 |
| Unit tests | Not attempted | <10 tests partial test coverage of functionality | 10-19 tests partial test coverage of functionality | 20-29 tests partial test coverage of functionality | 30-40 tests partial test coverage of functionality | >40 tests full test coverage of functionality | NOTE: %TESTS will be reduced for incomplete test coverage |
| % TESTS passed | No tests pass | All tests pass. | All tests pass. | All tests pass. | All tests pass. | All tests pass. | \* 1 |

**Final marks = 2 \* min{ marks( Class diagram ),  
marks( Dynamic UML diagrams & wireframe & storyboard & Structure diagrams ),  
marks( MoSCoW features ),  
marks( implemented requirements ) }  
+ min{ marks( Number of unit tests ),  
marks( Number of tests passed ) }**