Optional Assignment 6

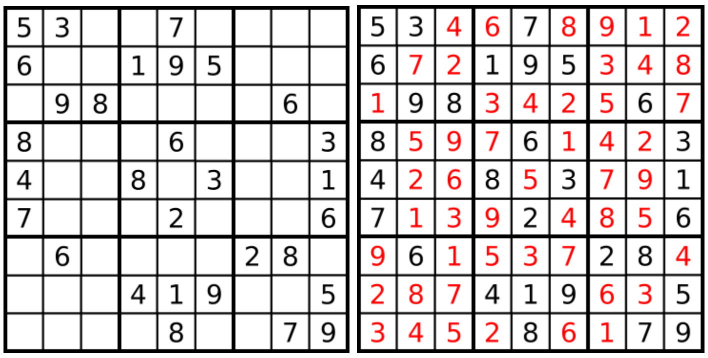
Deadline:	Hand in by ***midnight Tuesday, 3rd January 2015***. Please make an appointment with one of the teaching team members (Suriadi Suriadi, Teo Susnjak, Indu Sofat, or Tong Liu) to set a time for face-to-face marking of the assignment by Friday 6th Feb 2015**.**
Evaluation:	8% of your final grade if your mark scores higher than any one of your previous 5 assignments
Late Submission*:*	*1 mark deduction per day late*
Marking and Penalty	The assignment will be marked face-to-face with a tutor *Failure to attend the face-to-face marking session with your tutor (without prior alternative arrangement) will result in your marks being reduced by 50% (out of the possible maximum of 8 marks).*
Work	This assignment is to be done **individually** – your submission may be checked for plagiarism against other assignments and against Internet repositories. If you adapt material from the Internet acknowledge your source.
Purpose:	To learn to develop non-trivial application by applying basic and advanced OOP concepts, visual programming, event handling, input/output mechanisms, error handling, TDD methodology, and good programming practices.

Problem to solve:

Implement a Sudoku game application.

According to Wikipedia: "*Sudoku is a logic-based, combinatorial number-placement puzzle. The objective is to fill a 9×9 grid with digits so that each column, each row, and each of the nine 3×3 sub-grids that compose the grid (also called "boxes", "blocks", "regions", or "sub-squares") contains all of the digits from 1 to 9. The puzzle setter provides a partially completed grid, which typically has a unique solution. Completed puzzles are always a type of Latin square with an additional constraint on the contents of individual regions. For example, the same single integer may not appear twice in the same 9×9 playing board row or column or in any of the nine 3×3 sub-regions of the 9×9 playing board".*

Image below depicts a 9×9 playing board with a partial and a completed solution:



Images source: www.wikipedia.org

Sudoku comes in many variations including different board sizes such as 4×4 with four 2×2 regions and 6×6 with six 3×2 regions as seen below:

		6		4	
				6	
			5		3
3					
	1				
	5				4

Functional Requirements:

Create a Sudoku game application that allows a player to select to play a 4×4, 6×6 or a 9×9 board-size game. The application allows a user to select a text file and read from it a partially completed grid which the user must then solve. The format for the file will be as follows:

```
4x4
0 2 3
1 2 1
…
```

The first line signifies the size of the board to play. Each subsequent line has 3 integer values. The first signifies the row index in the board, the second the column index and the third the actual value for the cells at the given coordinate. The design decision on what data structures to use for representing the board is up to you.

It is entirely your decision regarding how the application looks and what controls you use, but keep usability in mind. When the user enters a number, the application should inform the user if a completed row, column or region are valid or not. Finally, once the entire board is populated, the application will inform the user if the board has been successfully completed.

Software Development Requirements:

Your software must follow all conventions and coding guidelines covered in the course thus far. Your assignment will be marked on the robustness and the comprehensiveness of your unit test suite built up during TDD (you need not write unit tests for event-handlers but you must write at least one unit test for each of your user-defined methods/properties within your classes).

Your entire software must be architecturally designed in such a way that it is decoupled from the presentation layer and is therefore testable. A deficient architectural design that tightly couples the logic with the UI will incur penalties.

Additionally, you are to demonstrate in your code the usage of the more advanced features of OO programming (material from the last week of lectures) such as interfaces or inheritance together with polymorphism if suitable to your design. Make sure once you finish your assignment, you go back to it and refactor it at least once. **Code that does not compile will receive zero marks.**

Marking Criteria:

Functionality (3%)

- Functional Sudoku game interface, including
    - the initialization of the game (0.5%),
    - the use of appropriate windows form controls and event handlers (1%)
    - the ability to cope with different types of board-size game (0.5%)
- Robust application logic that adhere to the rules of the Sudoku game (1%)

Programming Approach (4%)

- Appropriate decoupling of the presentation layer from the application logics. In particular, the application should be designed using the following architectural approach (1%):
  - the use of a Class Library Project to develop classes that encapsulate the necessary classes, including the necessary application (game) logics, and
  - a Windows Forms Project that creates the UI and uses the class library project containing all the application logic by referencing it.
- The use advanced OOP concepts, including (but not limited to) inheritance, abstract class(es), and interface(s) (1%)
- Appropriate error handling mechanisms (1%)
- Testable application as per the TDD approach. You need not write unit tests for event-handlers but you must write at least one unit test for each of your user-defined methods/properties within your classes (1%).

Good Programming Practice (1%)

- Variables and function names must be named appropriately.
- Looping and conditional statements must follow conventional and best-practice principles.
- The application must be broken down into subroutines/functions that perform one task each.
- All code duplication must be eliminated**.**
- Proper handling of possible errors in the application for all input and operations in the application
- The inclusion of refactoring in the application as necessary.

Hand-in: Submit your **program** via **stream** in a **zip** file (**ONLY** if the submission site is not available email to s.suriadi@massey.ac.nz). Make sure you zip your **entire** solution folder so that the .sln as well as the relevant project files are included.

If you have any questions or concerns about this assignment, please ask the lecturer well in advance.