THE TURTLE CHALLENGE

## Introduction

I have treated this exercise as one of my production level application. Therefore, you will find **unit tests**, **Interface Segregation** and **Single Responsibility principles** and other **SOLID design principles** along with **Dependency Injection** patterns in the solution. You may argue I have over engineered it but if you keep **SOLID design principles** in mind while reviewing it you may agree with my approach. I am a big fan of **SOLID design principles** because it allows us to design / write the code which is maintainable and it goes hand in hand with **scrum** methodology.

It would be very much appreciated if you could kindly provide a feedback (negative and positive) on this project once it has been code reviewed. If you decide to not to carry on with my job application, it will help me to learn and get a better preparation for next job application.

## Technologies

C#

.Net framework v4.6

Log4net (Logging)

Unity (IoC container)

nUnit (unit testing)

RhinoMocks (mocking framework)

Visual Studio 2017

Newtonsoft.Json

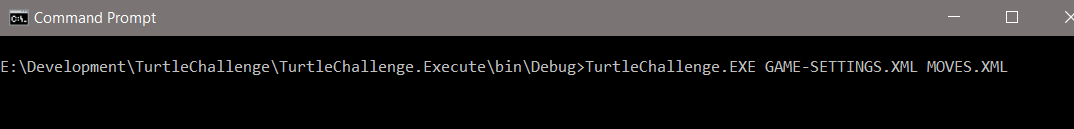
## Hosting

The application is designed in a way that it can be hosted as any executable (windows service, IIS web service etc) however, I have chosen a simple console application (*TurtleChallenge.Execute*) to host.

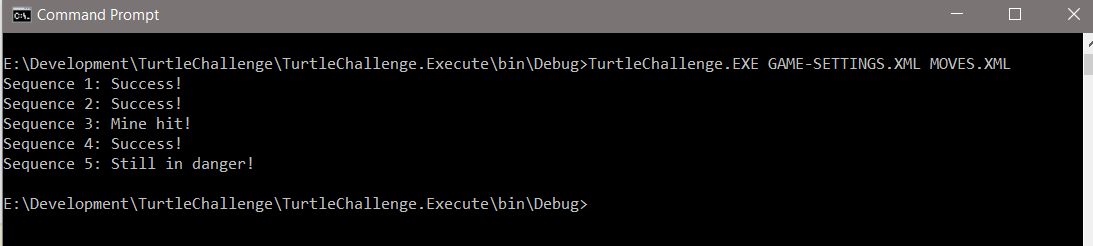
## How to run

Once the source code is downloaded / bootstrapped from the **GitHub**, open the solution file **TurtleChallenge.sln** in visual studio 2017. Build the solution using visual studio. Open the command prompt and navigate to the TurtleChallenge\TurtleChallenge.Execute\bin\Debug folder.

Type the command: turtlechallenge.exe game-settings.xml moves.xml to run the program.



With the game-settings.xml and moves.xml files you should get the following output unless these are changed.



## Supported File

Application supports 2 formats of game-settings and moves files:

1. XML
2. JSON

There is an interface IDataProvider which returns xml or json data provider depending upon the file extension. Its implementation knows how to retrieve the data from the file.

## A Valid Input File

As stated above, you can use xml or json file for game-settings and moves or any combination of the two.

## Assumptions made

**Game-settings**

1. Rows and columns of the board must be greater than zero.
2. Starting and end points cannot be the same.
3. Starting and end points and all mines must be within the game board.
4. Starting and end points cannot be on any mine.
5. The point x = 0, y = 0 is valid and represents the first row and first column tile on the game board.
6. If the turtle hits the border, the program will ignore it and it will keep the last valid location and will continue if rotated in other direction.
7. For each new sequence, the turtle is replaced back to its starting position and the direction defined in the game-settings file.

## Unit Tests

Most of the methods are developed using TDD approach, however the simple methods and properties in the domain model and console runner are not unit tested simply because they are trivial.

## Logging

You will find the info and the error log files under the bin/debug folder with names TurtleChallenge.Infolog.txt and TurtleChallenge.ErrorLog.txt respectively. Should you wish to change the location, please update the log4net.config file under config folder.

End of the document