# CSCI 428 - Object Oriented Programming – Assignment 2

The goal of assignment 2 is to write a C# console application that plays a single-player game of Hangman. Classically, the game is played on paper:



Game Requirements

* At startup, reads a list of possible puzzle phrases from a text file you create
* Randomly selects one of the puzzles
* Asks the user their name
* Prints on the display:
  + A greeting that includes the user’s name
  + An underbar for each letter in the phrase, with spaces as appropriate
  + The letters incorrectlyPlafo guessed so far
  + A count of incorrect guesses
  + When/if the user is on his/her last guess, a special warning:
    - Danger: You only have one guess left!
  + A prompt telling the user to make their next guess
* Collect a character from the user and:
  + Validate it to ensure it’s a letter, printing an error if not
  + Determine if it’s a part of the phrase
  + If so, reveal the letter(s) in the puzzle
    - If the puzzle is complete, print a “You won!” message and exit the program
  + If not:
    - Add the letter to the list of those guessed
    - Increment the wrong guess count
    - On the 6th wrong guess, the game is over, so print a “Game over – you lost” message and exit the program
* On each win/loss, update a log book in XML format containing:
  + The results of each game played:
    - Player name
    - Date/time
    - Puzzle phrase
    - Won or lost
  + Statistics for each player
    - Games won
    - Games lost
    - Winning percentage

Since you’ll be working with a text display, obviously the graphical element of the game is out of scope for the assignment. One suggestion is that you simple re-draw the entire game board at startup and each time player makes a guess. Keep it simple, make it fun!

As for the log book, keep it in the same directory as your application. Take a look at this post by Scott Hanselman for help:

<https://www.hanselman.com/blog/how-do-i-find-which-directory-my-net-core-console-application-was-started-in-or-is-running-from>

## Implementation Requirements

The following design elements must be implemented to receive full credit on the assignment:

* Implement a Game class that controls the flow of the game.
* Implement a Player class that manages the player attributes
* Implement a Puzzle class that manages the puzzle attributes

Beyond that, it’s up to you to make good design choices! There are a lot of possible ways for these classes to be implemented to work together.

## Log Book Specifications

<LogBook>

<Games>

<Game>

<PlayerName>string</PlayerName>

<DateTimePlayed>datetime m/d/yyyy h:i:s</DateTimePlayed>

<Phrase>string</Phrase>

<WonFlag>bool</WonFlag >

</Game>

</Games>

<Stats>

<Player>

<Name>string</Name>

<GamesWon>int</GamesWon>

<GamesLost>int</GamesLost>

<WinningPercentage>decimal 999.99</ WinningPercentage >

</Player>

</Stats>

</LogBook>

## Unit Testing

This assignment contains several elements that are quite unit-testable, including before-after state of the game following a guess, player win/loss percentage, and several others.

Find a way to implement at least a couple of unit tests for each of the major classes in your application.

Tests should follow the Arrange, Act, Assert pattern and be:

* Repeatable
* Deterministic
* Without environmental constraints; e.g., location of data files, existence of specific directories, etc.

## Extra Credit

Earn up to 10 extra credit points on Assignment 2 by providing a written narrative discussing ways my implementation of Assignment 1 could be improved.

No code is required, but be clear and precise with your recommendations and explain why your suggestions have merit.

The code to review is in the class bit bucket, here:

<https://bitbucket.org/tamuc-csci-428-public/assignment1-spring2023/src/main/>