# Assignment 4 – Classes and Collections

|  |  |  |
| --- | --- | --- |
| 1. | Create a class Player that can keep track of a player’s name, wins, and losses. Use private variables and public properties. (Ch. 4, starting page 73) | 15% |
|  |  |  |
| 2. | Create an enum Actions with the values ‘Rock,’ ‘Paper’, and ‘Scissors’ (or anything to that effect if you want to get creative…). (Ch. 7, page 175-177) | 15% |
|  |  |  |
| 3. | Create a method in the Player class to allow a player to play against another player by having each randomly choose an Action from the enum Actions. If actions are numbered 0, 1, …, n, assume that action 0 defeats action n; action x defeats action x – 1; otherwise, the players play again. Appropriately update each players’ wins and losses. The function should check to make sure a player is not playing against themselves. | 15% |
|  |  |  |
| 4. | Create a function that accepts *any number* of player names, and returns a List of players with those names. Instantiate a list of players using this function. (Ch. 8, page 234; Ch. 9, page 250-253) | 15% |
|  |  |  |
| 5. | Using a loop, have each player play another random player (not themselves), and repeat this for 5 rounds (so for 5 players, there should be a total of 25 games). (Optional: For fun, try using namespace System.Threading, and explore the function Sleep()). | 20% |
|  |  |  |
| 6. | After the games, use LINQ to display a leaderboard of the players sorted by wins in descending order. (Ch. 9, page 245-255) | 20% |
|  |  |  |
| 7. | Using LINQ, display any players with an equal number of wins and losses. | 10% Extra |

Submit the files to the assignment on MUOnline in a compressed (.zip) format. Include all project files for your programming solution. The submission should follow the naming convention:

CIT265\_*LastName\_FirstInitial\_*A4.zip