FULL SAIL UNIVERSITY

COP2334

PG2 – LAB: CARD WARS

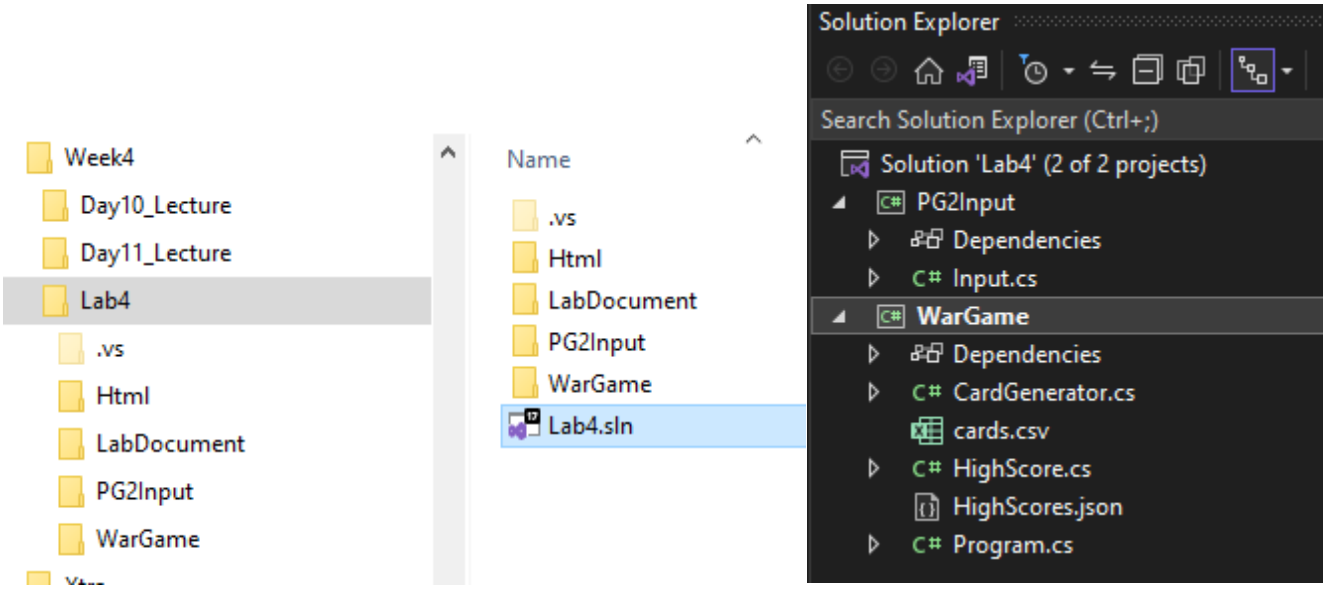
CONTENTS

FULL SAIL UNIVERSITY

COP2334

SETUP

A **C# .NET Core console application** has been provided for you in your **GitHub repo**. **Use the provided solution.**



Lab Video

Here's a video showing what the lab could look like when completed:

https://web.microsoftstream.com/video/504e89cc-ee22-43e2-ab1c-34d5a25483c6

Lecture videos for lab 4

FILE I/O LECTURE:

https://fullsailedu-my.sharepoint.com/:v:/g/personal/ggirod_fullsail_com/EdYpySsTlvBPsXQjhUHJj58Bat0hq6vY-lrXswXBjKh5tA?e=HeXea9

SPLITTING STRINGS LECTURE:

https://fullsailedu-my.sharepoint.com/:v:/g/personal/ggirod_fullsail_com/EQdvRQGWin1It3KDZxkBxf0BeMXHNSl5wurc6zSKbGeq0g?e=yxd0Ls

FULL SAIL UNIVERSITY

COP2334

PART A HIGHSCORE CLASS

Part A-1: HighScore class

Add a **HighScore** class with a Name property and a Score property.

Part A-2: LoadHighScores

Add a **LoadHighScores** method to the **HighScore** class. It should have a string parameter for the file path. In the method, it should deserialize the file into a List<HighScore>. Return the list.

NAME	RETURNS	PARAMETERS	COMMENTS
LoadHighScores	List<HighScore>	string	Deserializes the file into a list and returns the list.

In Main, before the while loop, call LoadHighScores passing the highScoreFile variable and store the list it returns into a variable to be used later.

Part A-3: SaveHighScores

Add a **SaveHighScores** method to the **HighScore** class. It should have a string parameter for the file path and a List<HighScore> parameter. In the method, it should serialize the list of high scores to the file. Call this method in the game when a player gets a new high score.

NAME	RETURNS	PARAMETERS	COMMENTS
SaveHighScores	nothing	string List<HighScore>	Serializes the list into the file.

Part A-4: Show High Scores

Add a **ShowHighScores** method to the **HighScore** class. It should have a List<HighScore> parameter. It should print a “High Scores” title then loop over the high scores list and print each item. Format the output so that the scores are right-aligned and have a color different than the name. See example screenshot.

NAME	RETURNS	PARAMETERS	COMMENTS
ShowHighScores	Nothing	List<HighScore>	Prints the list of high scores.

In case 1 of the menu switch in Main in Program.cs, call the ShowHighScores method and pass the list of highscores.

```
1. Show High Scores
2. Play Card Wars
3. Exit
Choice? 1
----HIGH SCORES----
Batman       52
Bruce Wayne  52
Robin        40
Joker        39
GDawg!       36
GMan         32
Garrett      32
Alfred       31
Flash        30
Superman     29
```

PART B CARDWARS CLASS

Part B-1: LoadCards

Add a **LoadCards** method to the **CardWars** class. It should have a string parameter for the file path. In the method, it should read the csv file, **split** the data into a List<string>. Return the list.

NAME	RETURNS	PARAMETERS	COMMENTS
LoadCards	List<string>	string	Read the csv file, split the data, and return a list of strings.

In Main, before the while loop, call LoadCards passing the cardsFile variable and store the list it returns into a variable to be used later.

Part B-2: Play War Game

Add a **PlayGame** method to the **CardWars** class. It should have 3 parameters: List<string> for the cards, List<HighScore> for the high scores, and a string for the name of the high score file.

NAME	RETURNS	PARAMETERS	COMMENTS
PlayGame	nothing	List<string> List<HighScore> string	See below for the description of what the method should do.

Here is the game logic to put in **PlayGame**:

1. Call shuffle passing in the list of cards.
2. Take the shuffled list and split it into 2 equal lists: **playerCards** and **npcCards**.
3. Create 3 lists: **playerPile**, **npcPile**, **unclaimedPile**.
4. Loop while the **playerCards** list is not empty

a. Print out the first card from playerCards and npcCards (see example below on how to print)
b. Add the first card from playerCards and npcCards to the unclaimed pile.
c. Call `CompareCards` and pass the first card from the playerCards and npcCards.
    i. NOTE: `CompareCards` will return -1 if the card1 < card2, 0 if card1 = card2, 1 if card1 > card2
d. If `CompareCards` returns -1, add the unclaimed pile to the npcPile. Clear the unclaimed pile. Print NPC wins.
e. If `CompareCards` returns 1, add the unclaimed pile to the playerPile. Clear the unclaimed pile. Print player wins.
f. Remove the first card from the playerCards and npcCards.

5. After the loop, check who won. Print the counts from the playerPile and npcPile lists.
    a. If the npcPile has more cards, print that the npc won the round.
    b. If the npcPile has the same number of cards as the playerPile, print that it was a tie.
    c. Else, the playerPile has more cards. Print out that the player won and check if it's a new high score.
        i. NOTE: the last score in the high score list is the smallest high score. Therefore, if the playerPile count is greater than the last score in the high score list, the player has a new high score.
        ii. If a new high score,
            1. Get the user's name using Input.GetString
            2. loop from the beginning of the high score list
            3. If the score is >= the high score, then
                a. insert the high score into the list at that index
                b. remove the last score in the list
                c. call SaveHighScores (see part A-3)

In case 2 of the menu switch in Main, call PlayGame to play a game of war!

```
 2♣ vs 10♣  player wins
 A♦ vs  J♦  player wins
 J♠ vs  J♣
 8♣ vs  Q♠  player wins
 3♣ vs  2♦  NPC wins
 6♦ vs  4♥  NPC wins
 7♦ vs  9♦  player wins
 3♠ vs  9♣  player wins
 3♦ vs  6♣  player wins
 K♦ vs  K♠
10♥ vs  6♥  NPC wins
 9♠ vs  K♣  player wins
 5♠ vs  A♥  NPC wins
 Q♥ vs 10♠  NPC wins
 K♥ vs  5♣  NPC wins
 8♦ vs  7♥  NPC wins
 5♦ vs  2♥  NPC wins
 7♠ vs  J♥  player wins
 4♦ vs  6♠  player wins
 Q♣ vs  8♠  NPC wins
 7♣ vs  3♥  NPC wins
10♦ vs  2♠  NPC wins
 A♣ vs  4♠  player wins
 Q♦ vs  9♥  NPC wins
 A♠ vs  5♥  player wins
 8♥ vs  4♣  NPC wins
NPC wins! 28 vs 24
```

FULL SAIL UNIVERSITY

COP2334

RUBRIC

Part A

FEATURE	POINTS
Part A-1: HighScore class	10
Part A-2: LoadHighScores	20
Part A-3: SaveHighScores	15
Part A-4: Show High Scores	15
TOTAL	60

Part B

FEATURE	POINTS
Part B-1: LoadCards	15
Part B-2: Play War Game	25
TOTAL	40