War – Card Game

Learning Goals:

1. Learn to make and work with a c++ project with multiple files.
2. Practice creating data types using classes in c++.
3. Practice adding data members to classes in c++.
4. Practice adding methods to classes in c++.
5. Practice using enums in c++.

Classes to start with:

1. PlayingCard – represents a single card. Has a face value and suite.
2. Deck – Has a list of 52 cards. They are initialized to the 52 possible cards in a real deck.

Overarching Project Goal:

> Using the classes provided, add other classes and logic that will result in a simplified version of the War card game.

What to start with: Use the PlayingCards code posted for you notes. www.harding.edu/dsteil/170/class resources/projects/war.zip

Project Tasks:

1. Modify the Deck class:
   a. Add a **`shuffle`** method. Call it from the Deck's constructor after the cards are initialized. Be sure the shuffling is random. (5 points)
   b. Add private **`cardCount`** integer that is used to keep track of the number of cards in the **`cards`** list. After the deck is initialized the **`cardCount`** should be 52. (1 point)
   c. A public **`getCardCount`** method that simply returns the **`cardCount`**. (1 point)
   d. A **`popLastCard`** method that returns the last card in the player's hand then decrements **`cardCount`**. The card to return is always at the **`cardCount`** index after it is decremented. (3 point)
2. Add a Player class that has:
   a. A private array of PlayingCard named **`hand`** that is large enough to hold 52 cards. (2 points)
   b. A private **`cardCount`** integer that is used to keep track of the number of cards in the player's hand. (1 points)
   c. A public **`wins`** integer that is used to keep track of the number of wins the player has. (1 points)
   d. A constructor that initializes the **`cardCount`** to 0 and the **`wins`** to 0. (1 points)
   e. A public **`getCardCount`** method that simply returns the **`cardCount`**. (1 points)
   f. An **`addCard`** method that takes a PlayingCard as an argument. It should set the next available position in the player's **`hand`** to the card that is passed in, then increment **`cardCount`**. (2 points)

g. A **`popLastCard`** method that returns the last card in the player's hand then decrements **`cardCount`**. The card to return is always at the **`cardCount`** index after it is decremented. (2 points)

3. Modify main to:
   a. Have one Deck and two Player objects. (1 point)
   b. Add a loop to deal the cards to the players. The loop should continue while there are cards remaining in the deck. Every other card should be distributed to the 2 players. To do this simply pass the result of deck's popLastCard method into the player's addCard method. (3 points)
   c. Add another loop that will run the War game. (4 points)
      i. The loop should continue while the players still have cards. This will only be 26 times for this simplified version of War.
      ii. Pop the last card from each player and compare the cards.
         1. Display each card that is popped.
         2. Display who wins the round or if it is a tie
         3. Increment the **`wins`** on the player that wins each round.
         4. In the case of a tie, no one wins.
   d. Display who the winner is (Player1 or Player 2) by checking the **`wins`** of each. Also display their win counts. (2 points)
   e. Run the program multiple times to be sure that the results are random.

4. What to submit.
   a. Your Visual Studio project folder zipped. Easel will accept the file as War.zip. If you do not use Visual Studio simply submit a zipped file containing all of your source files (both .h and .cpp).