Black Jack

Introduction:

Motivation:

I have always enjoyed card games such as blackjack and poker. I see this project as an opportunity to program a game of blackjack. I also see an opportunity to reuse some code that I previously wrote for this class which represents a deck of cards.

Objective:

The objective of this project is to design a project, from scratch, using all the topics that are covered in EEEE 346: Advanced Programming. These are enumerated below as well as an initial plan to use all of the desired c++ functionality.

Using the Program:

In order to use the blackjack program, run the executable file. The window that is initially displayed presents a welcome message which is read from a .txt file, a short instructions section that details facets of blackjack, and a question of how many players are playing. Once a number of players is entered, the program asks for the names of all the players. Once the players are named, the program deals two cards (a hand) to each player as well as the computer player, the only difference between the computer player's (house's) and the human player's initial hand is that the house's first card is kept face down. After an initial hand is dealt, the program asks player one if they want to hit, if they do then the next card in their hand is dealt and then player one either busts or is asked again until player one either busts or does not want to receive another card. The program then cycles through all of the players until all players have been asked. Once all of the human players have been asked and their final scores have been calculated, the house calculates its value and takes additional cards until the value of the house's hand is above 16 or the house busts. After one round is played, the last player is recorded in the .txt file LastPlayer. The last thing that the program does is asks the player if they would like to play again, if the answer is yes then the same players are used for another round of blackjack, if the answer is no then the program is closed.

Class and Function Descriptions:

Class Card:

Class card represents one playing card. The rank and suit of the card is enumerated and uses an overloaded << operator to assign values to different cards.

Function GetValue:

Function get value retrieves the value of the card

Function Flip:

Function Flip flips a card either face up or face down.

Class Hand:

Class hand represents multiple cards.

Function Add:

Function add uses push_back to increase the container size of the hand by one.

Function Clear:

Function clear iterates through the vector and clears each element to clear the hand.

Function GetTotal:

Function get total adds the value of all face up cards and returns the numerical value. The function also treats the value of an ace as an 11 unless it will bust a player.

Class GenericPlayer:

Class GenericPlayer is an Abstract Class and is used as the base class for the polymorphic classes of Player and House.

Function IsHitting:

Function IsHitting is a pure virtual function indicating the need to override this function in every class derived from GenericPlayer.

Function IsBusted:

Function is busted signals to another portion of the program that the value of the player's hand is above 21 and that means that the player is busted

Function Bust:

Function bust outputs that the player busts**.**

Class Player:

Class Player is a derived class from class GenericPlayer and represents a human player.

Function IsHitting:

Function IsHitting overrides IsHitting from GenericPlayer and takes a character response of either 'y' or 'n' which indicates whether the human player wants another card.

Function Win:

Function Win outputs that the player has won the hand of blackjack.

Function Lose:

Function Lose outputs that the player has lost the hand of blackjack.

Function Push:

Function Push outputs that the player has tied the dealer in the hand of blackjack.

Class House:

Class House is a derived class from class GenericPlayer and represents the computer (house) player.

Function IsHitting:

Function IsHitting overrides IsHitting from GenericPlayer and continues to give new cards to the house until the total is above 16 or the house busts.

Function FlipFirstCard:

Function FlipFirstCard flips the house's first card and shows the players.

Class Deck:

Function Populate:

Function Populate creates the deck using the enumerated list in class card.

Function Shuffle:

Function Shuffle uses an algorithm random_shuffle to shuffle the cards in the deck.

Function Deal:

Function Deal deals the cards by popping them from the stack of cards.

Function AdditionalCards:

Function AdditionalCards allows either the human or computer player to keep hitting (taking more cards) until they are either busted or indicate they do not want another card.

Class Game:

Function Play:

Function Play contains the general gameplay environment and facilitates the game being played which includes calling functions from various classes.

Function Instructions:

Function instructions reads a welcome message from a file and then displays brief instructions and facets of blackjack.

The main function facilitates the class game, gets the number of players, and writes the last player to a file.

Diagram of Class Hierarchy:

Class Player is-a GenericPlayer.

Class House is-a GenericPlayer.

Class Deck is-a Hand.

Table of Required Topics:

Subject	Yes/No	Where in File
Pointers	**Yes**	**Everywhere**
Classes	**Yes**	**7 Classes**
Inheritance	**Yes**	**Class Deck, Player,House**
Polymorphism	**Yes**	**Class Player, House**
Operator Overloading	**Yes**	**Class Card,GenericPlayer**
File Processing	**Yes**	**Function Instructions, Main**
Standard Library	**Yes**	**Class Card, Function Shuffle, Add, Clear**
Data Structures	**Yes**	**Class Card**

Classes: I used 7 classes to implement a blackjack game.

Card: This class will represent one playing card

Hand: A collection of Cards

Deck: Derived from hand, a collection of cards that represents the blackjack deck. Include shuffling and dealing

GenericPlayer: Base player. This will be the base of both human and house

Human: Human blackjack player

House: Automated blackjack player

Game: Game where other classes are called and interact with each other.

Overloading: I used operator overloading in order to assign cards to a hand.

Inheritance: I will use public inheritance to for is-a relationship such as Player is-a GenericPlayer and Deck is-a Hand (collection of cards)

Polymorphism: GenericPlayer is an abstract class because it is only used as a base class to Player, and house.

File Processing: I used the standard library fstream to read information from a file in order to welcome the user to the blackjack program and will write the name of the last player to play to a file.

Standard Library: I used a standard library when I made the class hand a container. I then used algorithms to shuffle the cards and an iterator to clear the memory of class hand to prepare for the next hand.

Data Structures: I used data structures by grouping different cards located in a hand under one name.