Welcome to Casino Royale

Blackjack (also called "21" or "twenty-one") is the most popular Casino cards game. There are more than 100 variations of Blackjack in different Casino houses.  
We will use the **simple double-exposure variation** in order to make the software modeling readable.

Given are the following **Classes** along with some brief descriptions which you can use.  
**Note**: Feel free to write any additional Variables/Methods to help your code

The game should run and display the final result when the user executes the **play()** method in class **BlackJackGame.**

class Card:

* **Data Members**
  + string suit
  + int rank
* **Function Members**
  + getSuit() - Returns the suit of the current card
  + getRank() - Returns the rank of the current card
  + printCard() - Returns the current card in String format. Eg if Suit is 'Diamonds' and Rank is '3'. Then this method should return "3 of Diamonds

class Deck

* **Data Members**
  + Card[] theDeck
  + String[] suits = {"Spades", "Hearts", "Clubs", "Diamonds"}
  + int currCard (Controls the index of the next card to be dealt)
* **Function Members**
  + shuffle() - Shuffles the deck in random order.
  + deal() - Returns a card for dealing.
  + closeToEmpty() - Returns true if we're close to running out of cards (less than 10).

class Hand

* **Data Members**
  + int MAX\_CARDS = 11 (At no point should any player's hand exceed 11 Cards. This value never changes universally)
  + Card[] cards - This should represent the cards of the hand
  + int numCards - The number of cards currently in the present Hand
* **Function Members**
  + getCards() - Return a multi-line string representation of the current cards in the hand
  + clear() - Clear the hand of its current cards
  + addCard(Card card) - Add a card to the hand.
  + total() - Add up all the values of the cards based on rank.
    - Add up all the values of the cards based on rank.
    - Find the highest value of hand depending on Ace = 1 or 11 points, whichever is to your benefit without going over 21!
    - Also calculate the total based on all face cards (Jack, Queen, King) being 10 points.
    - Other cards in the deck are based on their face value.

class Player

* **Data Memebrs**
  + Hand hand - Every player has a hand. This info is known only to the player, and not anyone else.
  + int score - Score of the player's hand. This info is known only to the player, and not anyone else.
* **Function Members**
  + getScore() - Returns the score of the player (The number of rounds the player has won)
  + hit(Card card) - Add a card from Deck to the players Hand
  + total() - Returns the value of current hand.
  + getHand() - Returns the string representation of the current hand
  + addPoints(int points) - Add given points to the players current score.
  + handClear() - Discard/Clear all the cards in the current player's hand so it's empty.

class Dealer → Dealer is exactly like a player, with the additional responsibility of managing the Game (Think Inheritance here!)

* **Data Members**
* **Function Members**
  + autoPlay(Deck d) - Continually adds cards to their hand until their hand point value is 16 or greater.

class BlackJackGame

* **Data Members**
  + Deck theDeck
  + Dealer theDealer
  + Player thePlayer
* **Function Members**
  + createInitialHand(Player p, Deck d) - Initialize a BlackJack hand with 2 cards
  + resetHand(Player p) - Clear the given player's hand of any cards.
  + play()
    - Play the game repeatedly dealing hands until the deck is almost used up.
    - First deal the player and the dealer with two cards each.
    - Keep asking for input until we get 'Hit' or 'Stay'
    - Add a Card to the Player's Hand until he inputs HIT
    - If the  Player's total value crosses 21, this round is over, and Dealer wins this round
    - If the Player inputs STAY, The dealer will then add cards to his hand and try to beat the Player(Hint: Delaer.Autoplay()
    - Final Winner - if Human player is higher than dealer hand or the dealer hand has gone bust (over 21)
    - Reset hands for next round.