Game description: At the beginning of a a game a player has a set amount of money. For each round, the player decides how much money they would like to bet using a text box. Once an amount is set and it's a valid amount, the player is "dealt two cards" returning them a value of their total from the cards. The dealer is also "dealt two cards". The player is given the value of one of those cards. The player can then decide to "Hit" or "Stand". If they hit, they recieve another card. They can continue to "Hit" so long as they're total does not exceed 21. If it does, they lose the money they bet. They can also "stand" at anypoint after a "hit". Standing reveals the value that the dealer has. If the player has a higher value than the dealer, the dealer will continue to draw new cards until they have a value higher than the player or over 21. If the dealer is over 21 the player wins the amount that they bet, otherwise they lose the value of their bet. The cards are "reshuffled" and a new round then begins.

Notes: cards have a value between 2 and 10. There is also an A card valued at either 1 or 11.
If either the player or the dealer have a value of 21, the round instantly ends.
We should allow card shuffling to be turned on or off to allow for card counting.

Division of labor:

UML diagram(s):

Prototype:

(inserting screenshot)

Algorithm:

1. Create a Card class
    a. Declare a static array of integers called cardsInPlay to prevent repeat cards
    b. For each card: generate a random number between 1 and 52. Check if that value is in the cardsInPlay. If it is get a new number, otherwise add that number to cardsInPlay and then divide that number by 4 and round down. Cards between 2 and 10 are assigned their respective values. Cards above 10 are assigned a value of 10. Cards with a value of 1 are assigned a value of 100 which will be used later to identify aces and determine if they are 1 or 11.
    c. Create a static reshuffle function that gets rid of all values in the cardsInPlay
2. Create UI
    a. Buttons for hitting and standing
    b. Quit button
    c. Betting input text field
    d. etc.
3. Create a dealer class
    a. Create instances of card class

    b. Assign cards by taking the first and third cards generated and giving them to the adding their values to the player and taking the other two and adding their values to the dealer.
    c. Create a stand method which adds cards to the dealer until it's total is either over 21 or over the player's value and either add or subtract the amount from the amount of money
4. Create a player class
    a. Player class will contain to total amount of money and the amount being bet.
    b. Create a hit method which generates a new card and double checks the total to see if it's over 21