The solution provided here goes through 3 different classes and the one class with the main method which is the blackJackGameSimulator. The first class that I started with was the cards class which produces two sets of arrays one for name and one for suits. The cardName array produces a value for each of the cards in the array through the method getValue. Anything in the array that is above the position 10 would equal to the value of 10. While at the position 1 of the array which is the ace it will be given a value of 11 to begin with. In order to get the values correctly I inputted a space in the beginning of the array to allow for the position to equal to value of the card so at position two of the array it would be the card two. This allowed for less of a hassle for the code. Decks class depicts how many cards are in one set of deck. The decks class starts with a for loop that is for the deck of cards and the s for the suits and n for the cardName array going from 1 to 13. Inside decks also has the shuffle deck method that would use a random number generator that would allow for different cards to be pulled out each time it goes through a play. Last method in there will deal the next card by going through a for loop for the num of cards in the deck. Each time it pulls a card in the deck it subtracts the number of cards from that. The last class is the DandPhands class and this will allow for the hand to be emptied for each play and to also add cards to the players hand. It will also produce the total value for both the dealer and the player. At this point when the players hand is greater than 21 with an ace in hand the value will turn to 1 but if there hand is under or at 21 the value will stay at 11.The display cards in hand method uses a for loop and if/else statement which will allow for the dealer to have one card hidden. So when the showFirstCard is set to false the Not operator at the beginning will check and end up producing the output of hidden.

The main class that produces the output is blackjackgamesimulator. It starts with the main method of welcoming the user to the game and asking them to enter their name through a scanner sequence. Then we use the new key word to create a brand-new keyword to create an instance of the class with the variable playerName there to overwrite any previous value the variable held. Next, we set the number decks to 4 and the Boolean operator to true inside the class decks. Then we set the Boolean operator gameEnd is set to false to allow the game to keep going were the user will then type how much money they insert. Once money is inserted it will ask the user if he would like to deal or end and if user selects deal goes to method playTheGame and if not, the game just ends. Playthegame method asks for user bet and starts by emptying user hand and dealer hand and then deals 2 cards to player and 2 to dealer with one hidden. It will then check for if player end is false through a if else statement which will then go to the method players choice were it will ask the user if they would like to hit or stay. Once user choice is done or user has busted it will then check if it is dealers turn were it will then hit if it is below 17 and stay when it is equal to or less than or equal to 21.

An alternative choice to this design would be instead of having array for the card names would be to go through an if/else statement that would produce the values for each card output. I didn't choose this path because an array makes this a lot more simple and faster than outputting each card name in that way. Another design approach that was considered was making a class for player choice and how the winner was decided. This would have allowed for less code in the main method but 2 more additional classes. I didn't do this because I felt like it wasn't necessary due to them being not that long of a method. At first I had the player hands within the main class while having the dealers hand separate but was causing too much confusion for myself so I decided to combine the dealer and player hand into one class which allowed for a more concise code.

```
/**
 * the DAndPHands class provides the information about both the
 * dealers and the players hand.
 * @author Thomas Muamar
 * @version 1.0
 */
package blackjacksim;

public class DAndPHands
```

```
{
   // intializing the variables
   private String playerName;
   private Cards[] playerHand = new Cards[10];
   private int cardsInHand;

   /**
    *
    * @param name Provides the players name
    */
   public DAndPHands(String name)
   {
      this.playerName = name;
   }
   /**
    * empty Hand method checks if player has a empty hand so
    * the dealer can then deal the two cards for the next
    * play.
    */
   public void emptyHand()
   {
      for(int dp=0; dp<10;dp++)
      {
         this.playerHand[dp] = null;

      }
      this.cardsInHand = 0;
   }
   /**
    *
    * @param card this allows for the use of the Cards class
    * @return when the players hand is less than 21 use is able to
    * can add a card to there hand
    */
   public boolean addCardToPlayersHand(Cards card)
   {
      this.playerHand[this.cardsInHand] = card;
      this.cardsInHand++;
      return (this.getpdHandTotal() <=21);

   }
   /**
    *
    * @return this returns the hand total and allows to produce the
    * value of the ace so if users hand is greater than 21 with
    * an ace it will return the value of 1 and if the users hand is
    * less than or equal to 21 it will return a 11.
    *
    */
   public int getpdHandTotal()
   {
      int handTotal = 0;
      int cardValue;
      int valueOfAces = 0;
      for(int c =0; c<this.cardsInHand;c++)
      {
         cardValue = this.playerHand[c].getName ();
         if(cardValue == 1)
         {
            valueOfAces++;
            handTotal += 11;
```

```
         }
         else if(cardValue >= 10)
         {

            handTotal += 10;

         }
         else
         {
            handTotal += cardValue;

         }

      }while(handTotal > 21 && valueOfAces > 0)
      {
         handTotal -= 10;
         valueOfAces--;

      }
      return handTotal;
      }
   /**
    *
    * @param showFirstCard allows for the first card of the dealer
    * to show and to also hide the second card of the dealer
    * Also display the card for the current player.
    *
    */

   public void displayCardsInHand(boolean showFirstCard)
   {
      System.out.println("cards in hand for " +this.playerName);

      for(int c=0; c<this.cardsInHand;c++)
      {
         if(!showFirstCard && c==0)
         {
            System.out.println("\t[hidden]");

         }else{
            System.out.println("\t" +playerHand[c]);

         }

      }

   }

}// end


/**
 * This class computes the values of each of the cards in an array and also allows
 * for the suit of the card to be displayed.
 * @author Thomas Muamar
 * @version 1.0
 */
package blackjacksim;

public class Cards
```

```
{
   // intializing the variables
   private int name;
   private int suit;
   private int value;
   private String[] cardName = {" ","Ace", "Two", "Three", "Four", "Five", "Six", "Seven",
"Eight", "Nine", "Ten", "Jack", "Queen", "King"};
   private String[] cardSuits = {"Clubs", "Diamonds", "Hearts", "Spades"};

   /**
    * @param suit Suit of card
    * @param value basically the score of the card
    * the within the array.
    * provided above with a bunch of setters and getters.
    */
   public Cards(int suit, int value)
   {
      this.name = value;
      this.suit = suit;
   }
   /**
    * getter for name
    * @return name
    */
   public int getName()
   {
      return name;
   }
   /**
    * getter for suit
    * @return suit
    */
   public int getSuit()
   {
      return suit;
   }
   /**
    * method gets the value of the card in the array cardName
    * @return provides the values for the cards in the
    * array provided through if/else statements.
    */
   public int getValue()
   {
       if(name > 10)
       {
           value=10;
       }
       else if(name == 1 )
       {
           value=11;
       }
       else
       {
         name = value;
       }
       return value;
   }
   /**
    * @return provides a string for the card that was
    * selected in the array and the suit from the array
    */
```

```
   public String toString()
   {
      return cardName[name]+" of "+cardSuits[suit];
   }
}// end

/**
 * this class provides how many cards are in a deck. when it computes
 * it will have a for loop for the number of suits and number of cards for the array
 * in the Cards class.
 * Then we have a method to shuffle the deck, deal the next card.
 * @author Thomas Muamar
 * @version 1.0
 */

package blackjacksim;

import java.util.Random;

public class Decks
{
   // intializing the variables
   private Cards[] cardsInDeck;
   private int deckOfCards = 52;
   /**
    * method produces the deck of cards
    * @param shuffle whether the deck needs to be shuffled or not
    */
   public Decks(boolean shuffle)
   {
      this.cardsInDeck = new Cards[this.deckOfCards];
      int c = 0;
         for(int s=0; s<4;s++)
         {
            for(int n=1;n<=13;n++)
            {
               this.cardsInDeck[c] = new Cards(s, n);
               c++;

            }

         }
      if(shuffle)
      {
         this.shuffleDeck();

      }

   }
   /**
    * Shuffle deck method just shuffles the deck to allow
    * for a different card to be displayed each time a user plays
    * the game.
    */
   public void shuffleDeck()
   {
      Random ds = new Random();
      Cards dec;

      for(int d = 0; d<deckOfCards; d++)
      {
```

```
            int i = ds.nextInt(this.deckOfCards);
            dec = this.cardsInDeck[d];
            this.cardsInDeck[d] = this.cardsInDeck[i];
            this.cardsInDeck[i] = dec;

        }

    }
    /**
     * This method deals the next card when the dealer is dealing
     * the cards and when the user takes a hit
     * @return Returns the next card in the deck
     */
    public Cards dealNextCard()
    {
        Cards nextCard = this.cardsInDeck[0];

        for(int f=1; f<this.deckOfCards;f++)
        {
            this.cardsInDeck[f-1] = this.cardsInDeck[f];

        }
        this.cardsInDeck[this.deckOfCards - 1] = null;
        this.deckOfCards--;
        return nextCard;

    }

}// end


/**
 * Main class that contents the main method that will go through
 * a few scanner sequences that asks for user name, amount of money
 * they want to input and the bets they would like to place.
 * @author Thomas Muamar
 * @version 1.0
 */
package blackjacksim;

import java.util.Scanner;

public class BlackJackGameSimulator
{
    // intializing the variables
    protected Decks newDeck;
    public String playerName;
    public float balance;
    public float bet;
    protected boolean playerEnd;
    protected boolean dealerEnd;
    protected DAndPHands dealer;
    private Scanner bc = new Scanner(System.in);
    protected DAndPHands player;
    private static Scanner sc;

    public static void main(String[] args)
    {
        sc = new Scanner(System.in);
        String playerName;
        System.out.println("Welcome to blackjack");
```

```
      System.out.println("Enter Your Name:\n");
      playerName = sc.nextLine();
      new BlackJackGameSimulator(playerName);
   }
   /**
    * This produces the balance and asks player if they
    * want to start or end game.
    * @param playerName users name
    *
    */
   BlackJackGameSimulator(String playerName)
   {
      this.newDeck = new Decks(true);
      boolean gameEnd = false;
      this.balance = 0;
      this.playerName = playerName;
      System.out.println("Enter amount of money you would like to insert: " +balance);
      this.balance = bc.nextFloat();

      player = new DAndPHands(this.playerName);
      dealer = new DAndPHands("Dealer");
      while(this.balance > 0 && !gameEnd)
      {
         System.out.println(this.playerName+", Do you want to Deal or End the game [Deal or
End]?");
         String gameOn = bc.next();

         if(gameOn.compareToIgnoreCase("Deal") == 0)
         {
            this.playTheGame();

         }
         else
         {
            gameEnd = true;

         }

      }

      System.out.println( this.playerName+ ",Game is Over");
      System.out.println("\n"+this.playerName+", would you like to Play again ??[Y or N]");
      String Y = bc.next();
      if(Y.compareToIgnoreCase("Y") == 0)
      {
         new BlackJackGameSimulator(this.playerName);

      }
      bc.close();

   }
   /**
    * The play game method will ask the user how much he
    * would like to bet and then it will deal cards to both
    * dealer and player.
    *
    */
   public void playTheGame()
   {
      boolean blackJack = false;
      this.bet = 0;
```

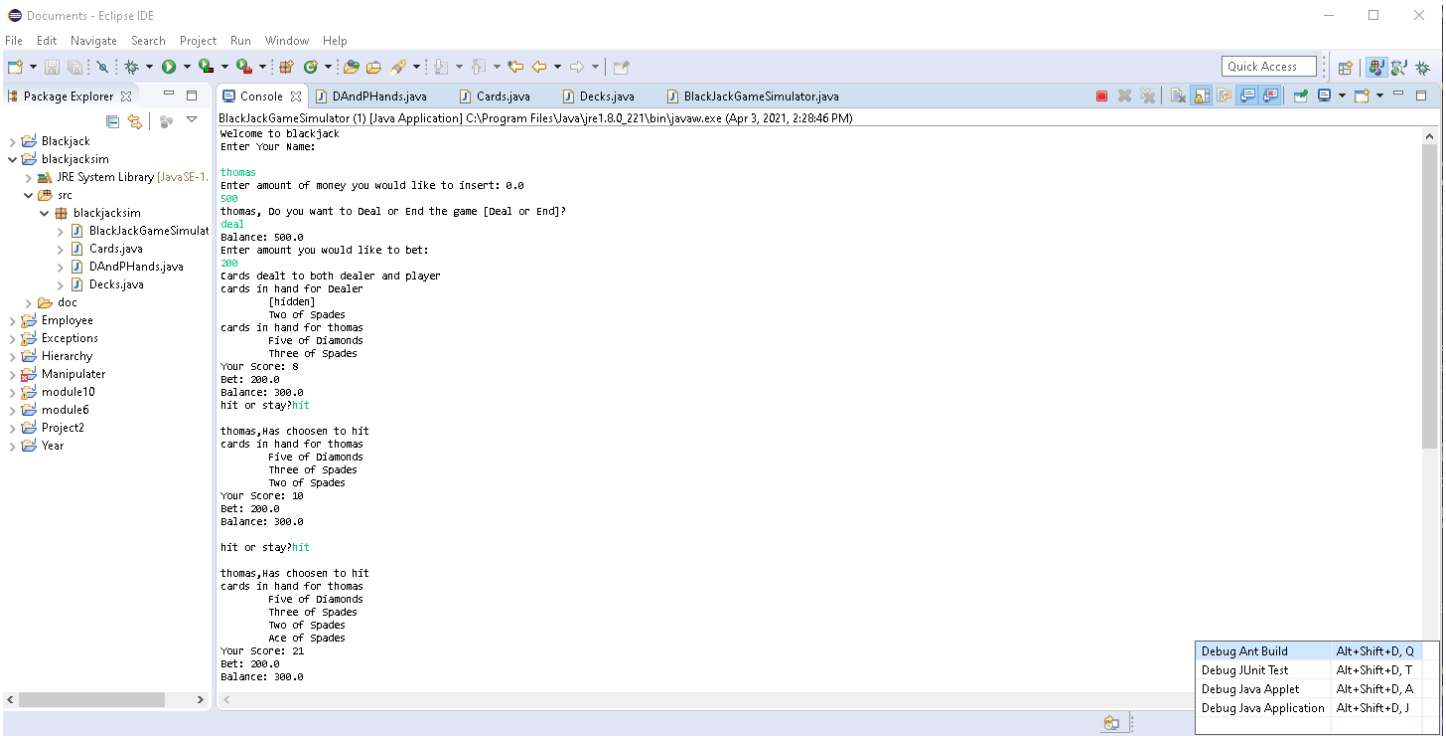
```
   System.out.println("Balance: " +this.balance);
   System.out.println("Enter amount you would like to bet: ");
   this.bet = bc.nextFloat();
   if((this.bet >= 1) && (this.bet%1 == 0) && (this.balance-this.bet>=0))
   {
      this.balance = this.balance - this.bet;
      // Empties the players to allow for the cards to be added.
      player.emptyHand();
      dealer.emptyHand();
      this.playerEnd = false;
      this.dealerEnd = false;
      // cards two both dealer and player
      player.addCardToPlayersHand(newDeck.dealNextCard());
      dealer.addCardToPlayersHand(newDeck.dealNextCard());
      player.addCardToPlayersHand(newDeck.dealNextCard());
      dealer.addCardToPlayersHand(newDeck.dealNextCard());
      //displays the cards drawn and what the users score is
      System.out.println("Cards dealt to both dealer and player");
      dealer.displayCardsInHand(false);
      player.displayCardsInHand(true);
      System.out.println("Your Score: " +player.getpdHandTotal());
      System.out.println("Bet: " +this.bet);
      System.out.println("Balance: " +this.balance);

      blackJack = checkBlackJack();

      while(!this.playerEnd || !this.dealerEnd)
      {
         if(!this.playerEnd)
         {
            playersChoice();

         }
         else if(!this.dealerEnd)
         {
            dealersTurn();

         }

         System.out.println();

      }
      if(!blackJack)
      {
         this.winLosePush();

      }

   }
   // if user puts bet that is over the balance
   else if (bet>balance)

   {
      System.out.println("Wrong bet amount, should not be more than balance");
      System.out.println("Your Balance: " +this.balance);

   }

}
/**
 * If dealer has blackjack game ends and if player has blackjack
```

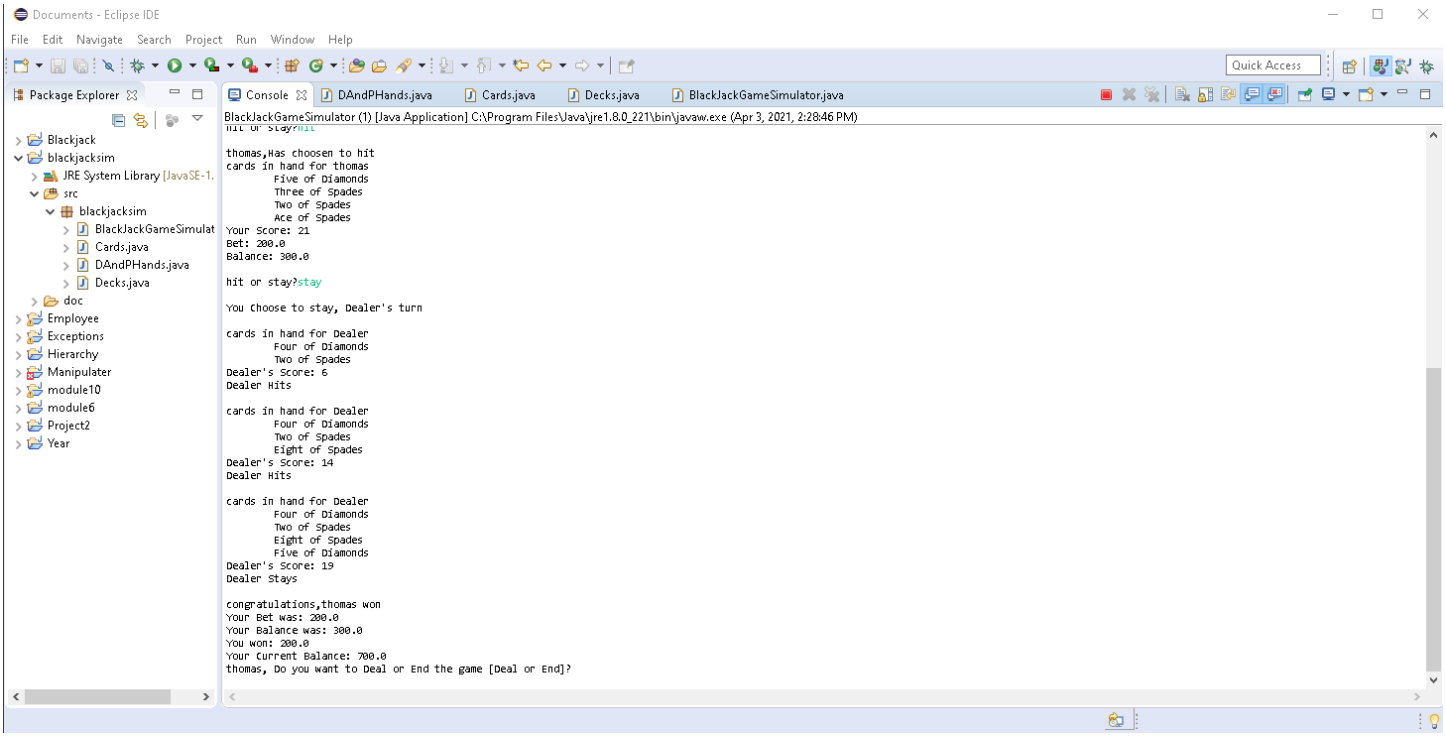
```
 * then player wins.
 * @return whether a person produces a blackjack or not.
 */
public boolean checkBlackJack()
{
   boolean blackJack = false;

   if(player.getpdHandTotal() == 21)
   {
      this.playerEnd = true;
      this.dealerEnd = true;

      if(player.getpdHandTotal() > dealer.getpdHandTotal() || dealer.getpdHandTotal() > 21)
      {
         System.out.println("Congurates"+playerName+ " you won by getting blackJack");
         dealer.displayCardsInHand(true);
         System.out.println("Dealer's Score: " +dealer.getpdHandTotal());
         System.out.println("Your Bet was: " +this.bet);
         System.out.println("Your Balance was: " +this.balance);
         System.out.println("You win: " +this.bet);
         this.balance = this.balance + this.bet + this.bet;
         System.out.println("Your Current Balance: " +this.balance);
         blackJack = true;

      }

   else if(dealer.getpdHandTotal() == 21)
   {
   dealer.displayCardsInHand(true);
   System.out.println("Dealer's Score: " +dealer.getpdHandTotal());
   System.out.println("Dealer has BlackJack."+playerName+ "Lost.");
   this.dealerEnd = true;
   this.playerEnd = true;
   blackJack = false;
   }
   }
   return blackJack;
}
/**
 * method will ask if the player would like to hit or stay
 */
public void playersChoice()
{
   String input;
   System.out.print("hit or stay?");
   input = bc.next();
   System.out.println();

   if(input.compareToIgnoreCase("Hit") == 0)
   {
      System.out.println( playerName+ ",Has choosen to hit");
      playerEnd = !player.addCardToPlayersHand(newDeck.dealNextCard());
      player.displayCardsInHand(true);
      System.out.println("Your Score: " +player.getpdHandTotal());
      System.out.println("Bet: " +this.bet);
      System.out.println("Balance: " +this.balance);

      if(player.getpdHandTotal()>21)
      {
         System.out.println("you have busted. Hand is over 21");
         dealer.displayCardsInHand(true);
```

```
            System.out.println("Dealer's Score: " +dealer.getpdHandTotal());
            playerEnd = true;
            dealerEnd = true;

         }

      }
      else if (input.compareToIgnoreCase("Stay") == 0)
      {
         System.out.println("You Choose to stay, Dealer's turn");
         playerEnd = true;
      }
   }
   /**
    * Once players turn is over the dealer will then hit if it is under 17
    * and if greater than 21 it will bust. Once over or at 17 it will stay.
    */
   private void dealersTurn()
   {

      if(dealer.getpdHandTotal() < 17)
      {
         dealer.displayCardsInHand(true);
         System.out.println("Dealer's Score: " +dealer.getpdHandTotal());
         System.out.println("Dealer Hits");
         dealerEnd = !dealer.addCardToPlayersHand(newDeck.dealNextCard());

         if(dealer.getpdHandTotal()>21)
         {
            dealer.displayCardsInHand(true);
            System.out.println("Dealer's Score: " +dealer.getpdHandTotal());
            System.out.println("Dealer has busted");
            dealerEnd = true;

         }

      }else{
         dealer.displayCardsInHand(true);
         System.out.println("Dealer's Score: " +dealer.getpdHandTotal());
         System.out.println("Dealer Stays");
         dealerEnd = true;

      }

   }
   /**
    * This program will see whether the dealer or player has won
    * , tied(push) or lost. checks the scores to see who has busted
    * or who has a larger score while being under 21.
    */
   private void winLosePush()
   {
      int playerScore = player.getpdHandTotal();
      int dealerScore = dealer.getpdHandTotal();

      if(playerScore>dealerScore && playerScore<=21 || dealerScore >21)
      {
         System.out.println("congratulations," +playerName+ " won");
         System.out.println("Your Bet was: " +this.bet);
         System.out.println("Your Balance was: " +this.balance);
         System.out.println("You won: " +this.bet);
```

```
            this.balance = this.balance + this.bet + this.bet;
            System.out.println("Your Current Balance: " +balance);

        }else if(playerScore == dealerScore){
            System.out.println("Push, get your money back");
            this.balance = this.balance + this.bet;
            System.out.println("Your Current Balance: " +this.balance);

        }else{
            System.out.println("Sorry, you lost");
            System.out.println("You lost: " +bet);
            System.out.println("Your Current Balance: " +balance);

        }

    }
}
```

Normal game with a single hit from user and stay:

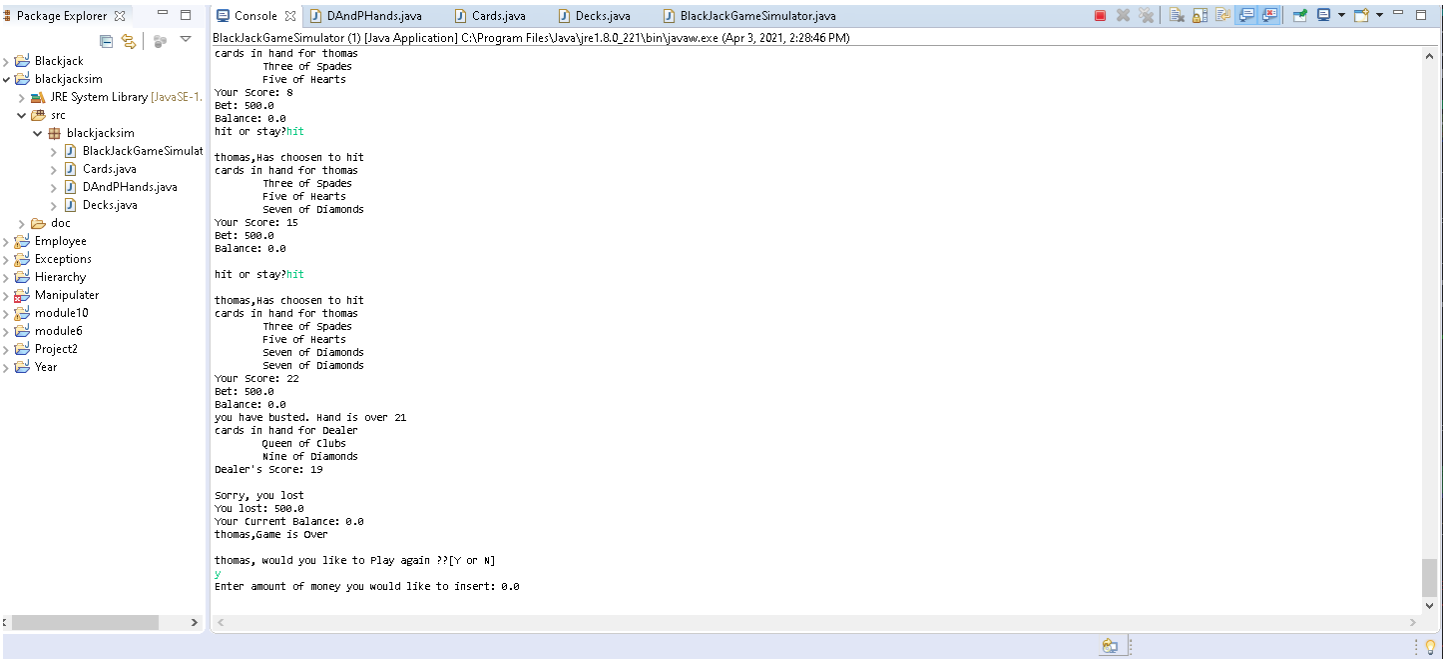




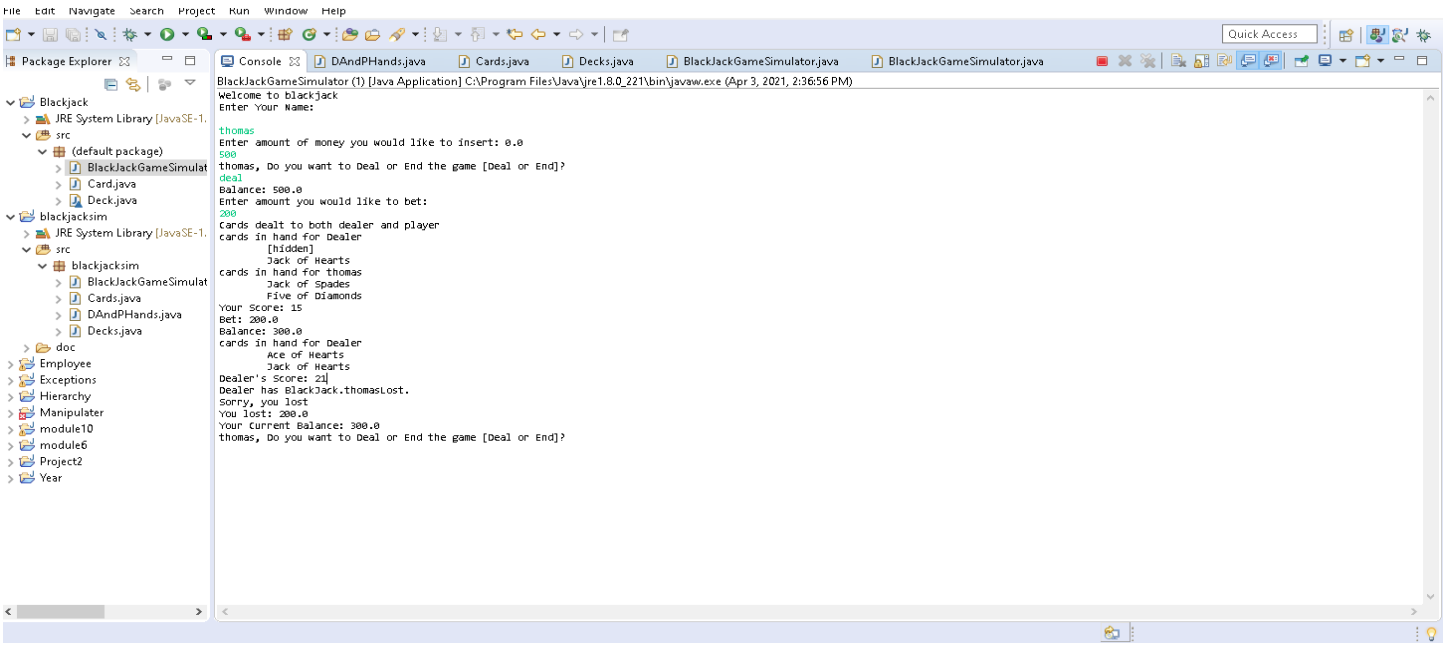
Continuation of the game were I tie with dealer:

```
Your Current Balance: 800.0
thomas, Do you want to Deal or End the game [Deal or End]?
deal
Balance: 800.0
Enter amount you would like to bet:
500
Cards dealt to both dealer and player
cards in hand for Dealer
        [hidden]
        Eight of Spades
cards in hand for thomas
        Queen of Clubs
        King of Clubs
Your Score: 20
Bet: 500.0
Balance: 300.0
hit or stay?stay

You Choose to stay, Dealer's turn

cards in hand for Dealer
        Seven of Spades
        Eight of Spades
Dealer's Score: 15
Dealer Hits

cards in hand for Dealer
        Seven of Spades
        Eight of Spades
        Five of Spades
Dealer's Score: 20
Dealer Stays

Push, get your money back
Your Current Balance: 800.0
thomas, Do you want to Deal or End the game [Deal or End]?
```

Output for when user loses all money and asks to play again:



When dealer has blackjack:



```
<terminated> Javadoc Generation
Loading source files for package blackjacksim...
Constructing Javadoc information...
Standard Doclet version 1.8.0_201
Building tree for all the packages and classes...
Generating C:\Users\thomas\Documents\blackjacksim\doc\blackjacksim\BlackJackGameSimulator.html...
Generating C:\Users\thomas\Documents\blackjacksim\doc\blackjacksim\Cards.html...
Generating C:\Users\thomas\Documents\blackjacksim\doc\blackjacksim\DAndPHands.html...
Generating C:\Users\thomas\Documents\blackjacksim\doc\blackjacksim\Decks.html...
Generating C:\Users\thomas\Documents\blackjacksim\doc\blackjacksim\package-frame.html...
Generating C:\Users\thomas\Documents\blackjacksim\doc\blackjacksim\package-summary.html...
Generating C:\Users\thomas\Documents\blackjacksim\doc\blackjacksim\package-tree.html...
Generating C:\Users\thomas\Documents\blackjacksim\doc\constant-values.html...
Generating C:\Users\thomas\Documents\blackjacksim\doc\blackjacksim\class-use\Decks.html...
Generating C:\Users\thomas\Documents\blackjacksim\doc\blackjacksim\class-use\DAndPHands.html...
Generating C:\Users\thomas\Documents\blackjacksim\doc\blackjacksim\class-use\Cards.html...
Generating C:\Users\thomas\Documents\blackjacksim\doc\blackjacksim\class-use\BlackJackGameSimulator.html...
Generating C:\Users\thomas\Documents\blackjacksim\doc\blackjacksim\package-use.html...
Building index for all the packages and classes...
Generating C:\Users\thomas\Documents\blackjacksim\doc\overview-tree.html...
Generating C:\Users\thomas\Documents\blackjacksim\doc\index-files\index-1.html...
Generating C:\Users\thomas\Documents\blackjacksim\doc\index-files\index-2.html...
Generating C:\Users\thomas\Documents\blackjacksim\doc\index-files\index-3.html...
Generating C:\Users\thomas\Documents\blackjacksim\doc\index-files\index-4.html...
Generating C:\Users\thomas\Documents\blackjacksim\doc\index-files\index-5.html...
Generating C:\Users\thomas\Documents\blackjacksim\doc\index-files\index-6.html...
Generating C:\Users\thomas\Documents\blackjacksim\doc\index-files\index-7.html...
Generating C:\Users\thomas\Documents\blackjacksim\doc\index-files\index-8.html...
Generating C:\Users\thomas\Documents\blackjacksim\doc\index-files\index-9.html...
Generating C:\Users\thomas\Documents\blackjacksim\doc\index-files\index-10.html...
Generating C:\Users\thomas\Documents\blackjacksim\doc\index-files\index-11.html...
Generating C:\Users\thomas\Documents\blackjacksim\doc\index-files\index-12.html...
Generating C:\Users\thomas\Documents\blackjacksim\doc\index-files\index-13.html...
Generating C:\Users\thomas\Documents\blackjacksim\doc\index-files\index-14.html...
Generating C:\Users\thomas\Documents\blackjacksim\doc\deprecated-list.html...
Building index for all classes...
Generating C:\Users\thomas\Documents\blackjacksim\doc\allclasses-frame.html...
Generating C:\Users\thomas\Documents\blackjacksim\doc\allclasses-noframe.html...
Generating C:\Users\thomas\Documents\blackjacksim\doc\index.html...
Generating C:\Users\thomas\Documents\blackjacksim\doc\help-doc.html...
```