PROJECT GAME LOGIC

<BLACKJACK GAME>

CIS18C-42030

Tiffany Lin

Date: 05/20/2015

Introduction

Title: BlackJack Card Game

This is the card game commonly known as BlackJack.

The user is first introduced with two cards in hand and can also see what the dealer has in hand. Based on the cards displayed, the user can decide whether or not to be dealt another card or to stay with the cards he/she already has. The user then enters a response of YES or NO. If YES, then the program deals the player another card. Based on the new total of all the card values, the user can decide whether to continue to add another card to the hand ONLY if the user has not hit 21 yet.

If the user hits 21, the program immediately ends the user's turn and goes to the dealer to automatically calculate whether or not to deal for the dealer. If both under 21 and the user has a higher card value total than the dealer's, then the user wins. If both under 21 the dealer has the higher card value total, then the dealer wins. If they are both tied, then the person with more cards in hand will win. The dealer wins if it hits 21 and the user did not.

The final score for each individual is displayed on screen and there is an output that indicates whether the user won or the dealer won.

This game deals with probability and math calculation. Anyone can enjoy this game.

Summary

Project Size: about 340 lines

Number of variables: 14

Number of methods: 24

This Blackjack game program contains many concepts that were taught previously in class. Such concepts include hashing, hashmap, stacks, vectors, searching concept, etc. Possible additional features that could be added to this program is allowing the user to play multiple rounds or allowing the user to input money amount and gamble a certain amount each round.

Overall, I found the program to be challenging but filled with opportunities to apply what I've learned so far to this program. In the beginning, I did not have too much of an issue of creating a hash code that would be specific to this program. What I did find difficult was that the option of allowing the user to choose 1 or 11 if there was an Ace in their hand kept looping initially. Although it took me a while to pinpoint the issue, the resolve showed me the importance of not calling methods without considering the effects it has on the variables and on the output. I am quite proud of how many concepts I was able to insert into the program but I think there are definitely room for improvements and more features such as the multiple rounds mentioned earlier.

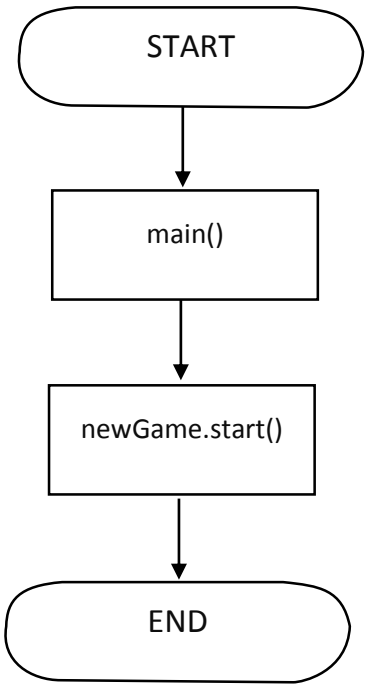
Description

The main point of the program is to enter whether or not user wants to be dealt another card based on the cards in hand (outputted on screen) and how to organize cards so that when user or dealer draws cards, they are random but not duplicate.

FlowChart

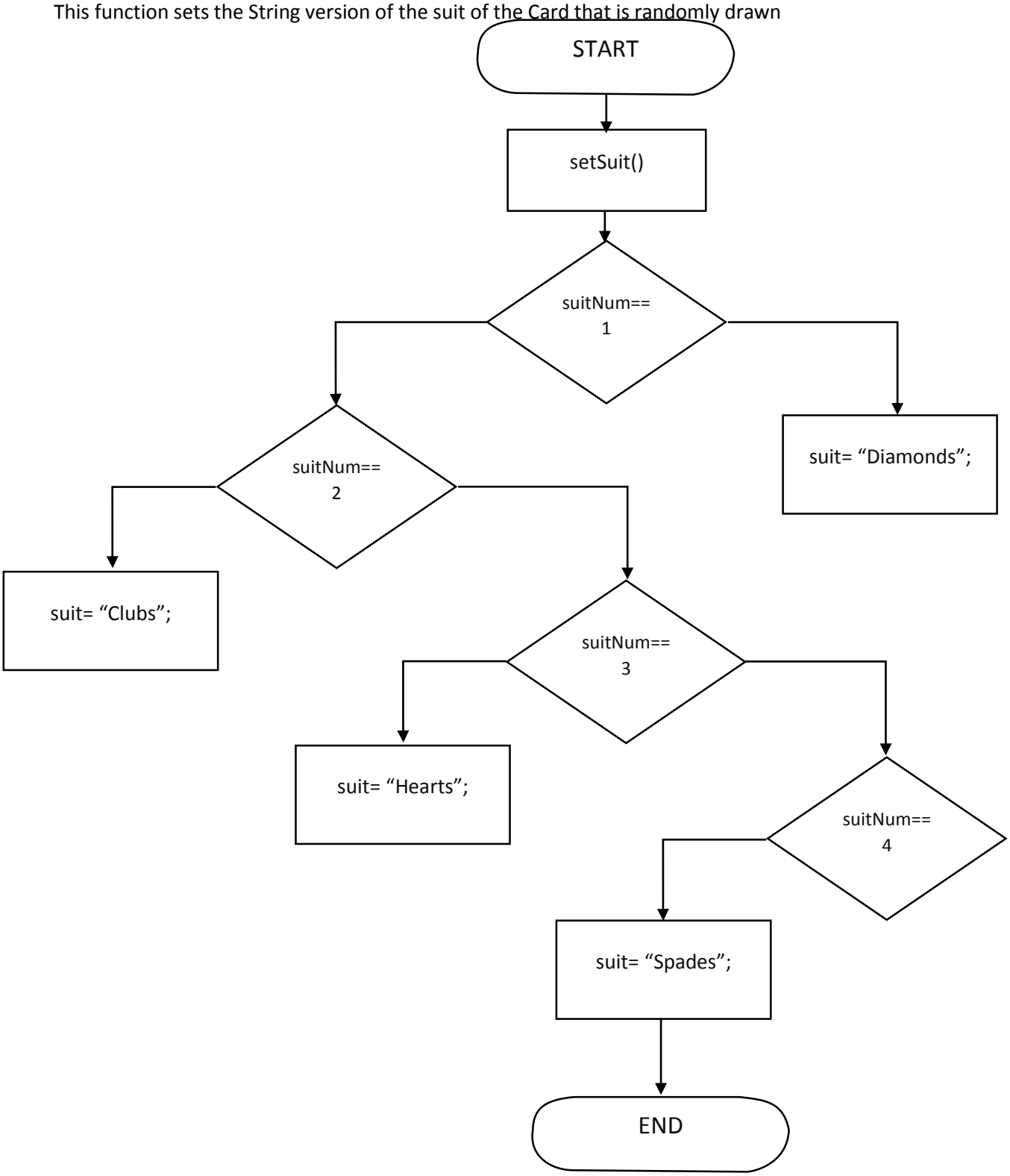
Function main(String[] args)

main() function initiates the new game automatically when program starts running.



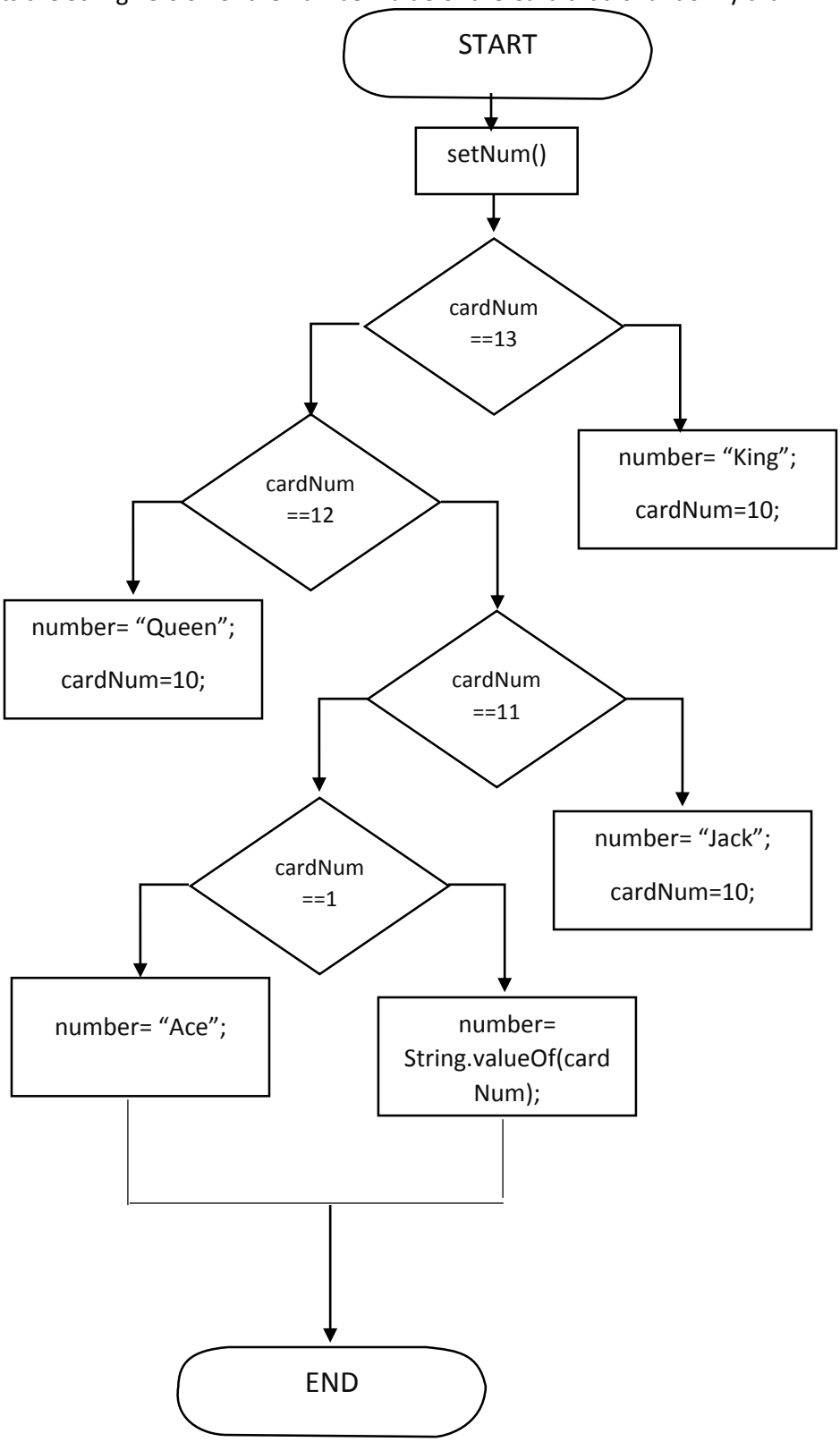
Function setSuit()

This function sets the String version of the suit of the Card that is randomly drawn



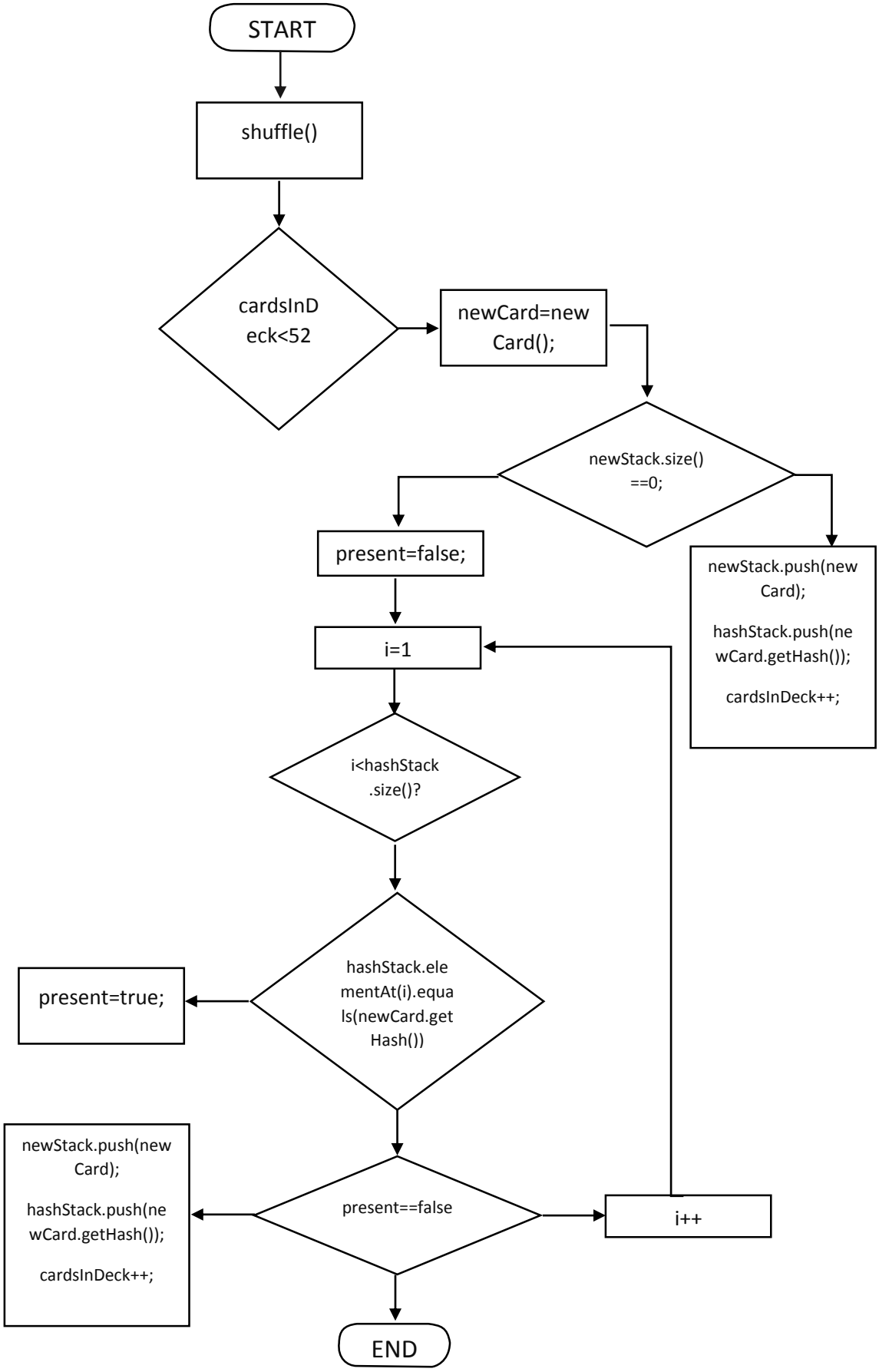
Function setNum()

This function sets the String version of the number value of the Card that is randomly drawn.



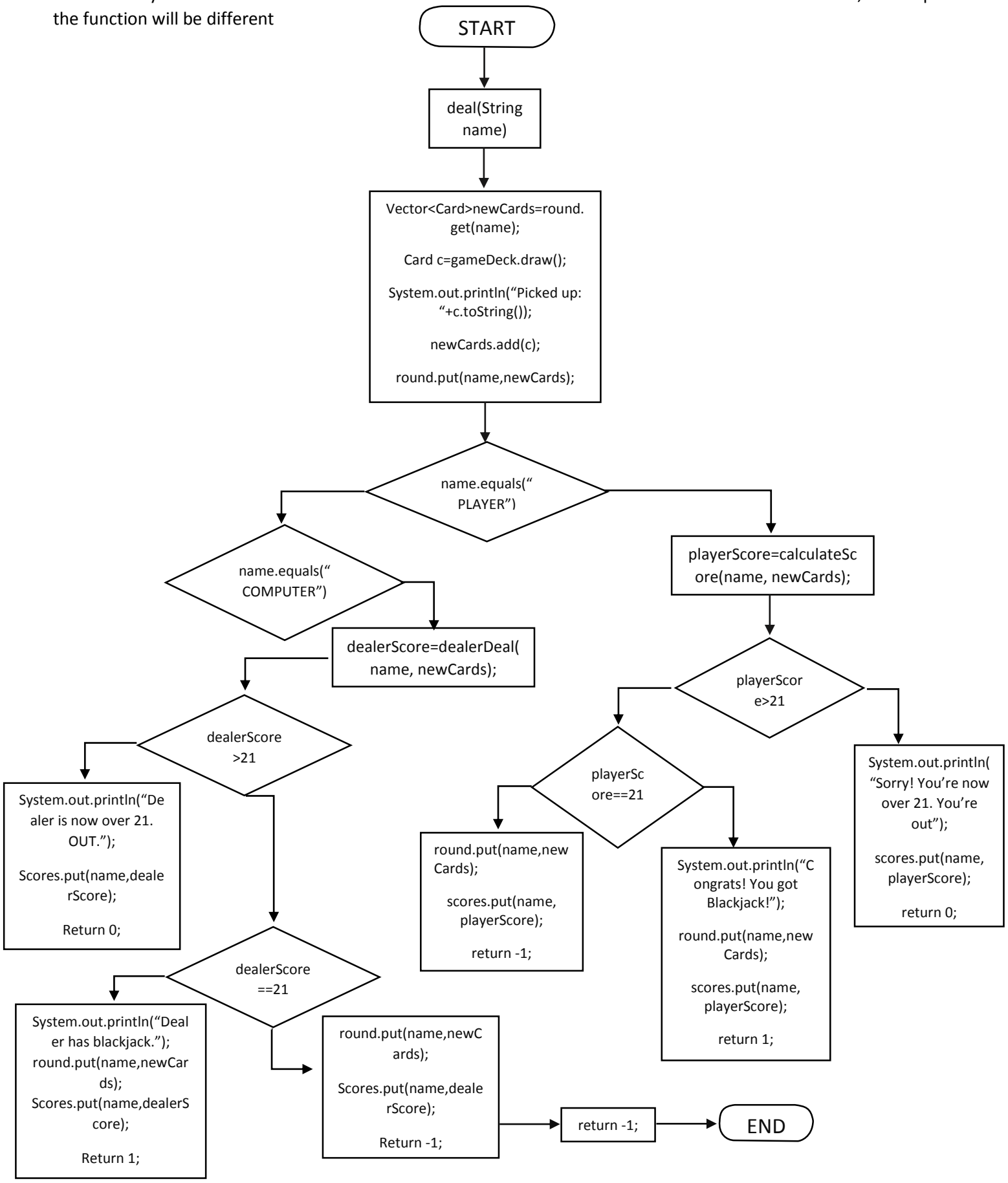
Function shuffle()

shuffle() function fills a stack with 52 cards and another stack with hash codes of each of the 52 cards. All the cards are distinct and reflect a realistic stack of cards



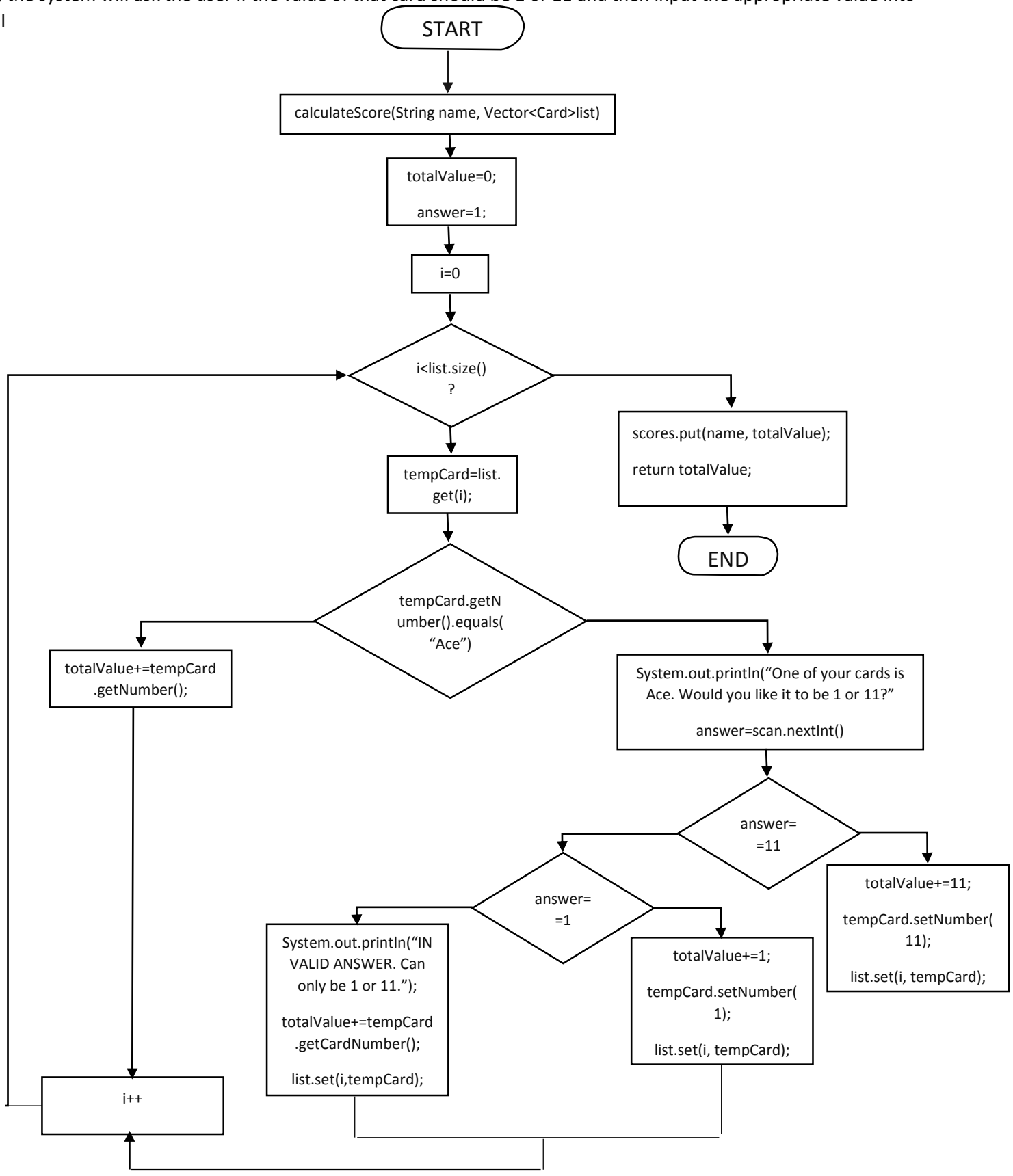
Function deal(String name)

deal Function takes the String name identifier and uses that as the key to store the vector of cards that will belong to the name. Depending on the name identifier, the deal function will either call the calculateScore() function or it will automatically make a decision of whether to draw cards. Based on the total value of the cards combined, the output of the function will be different



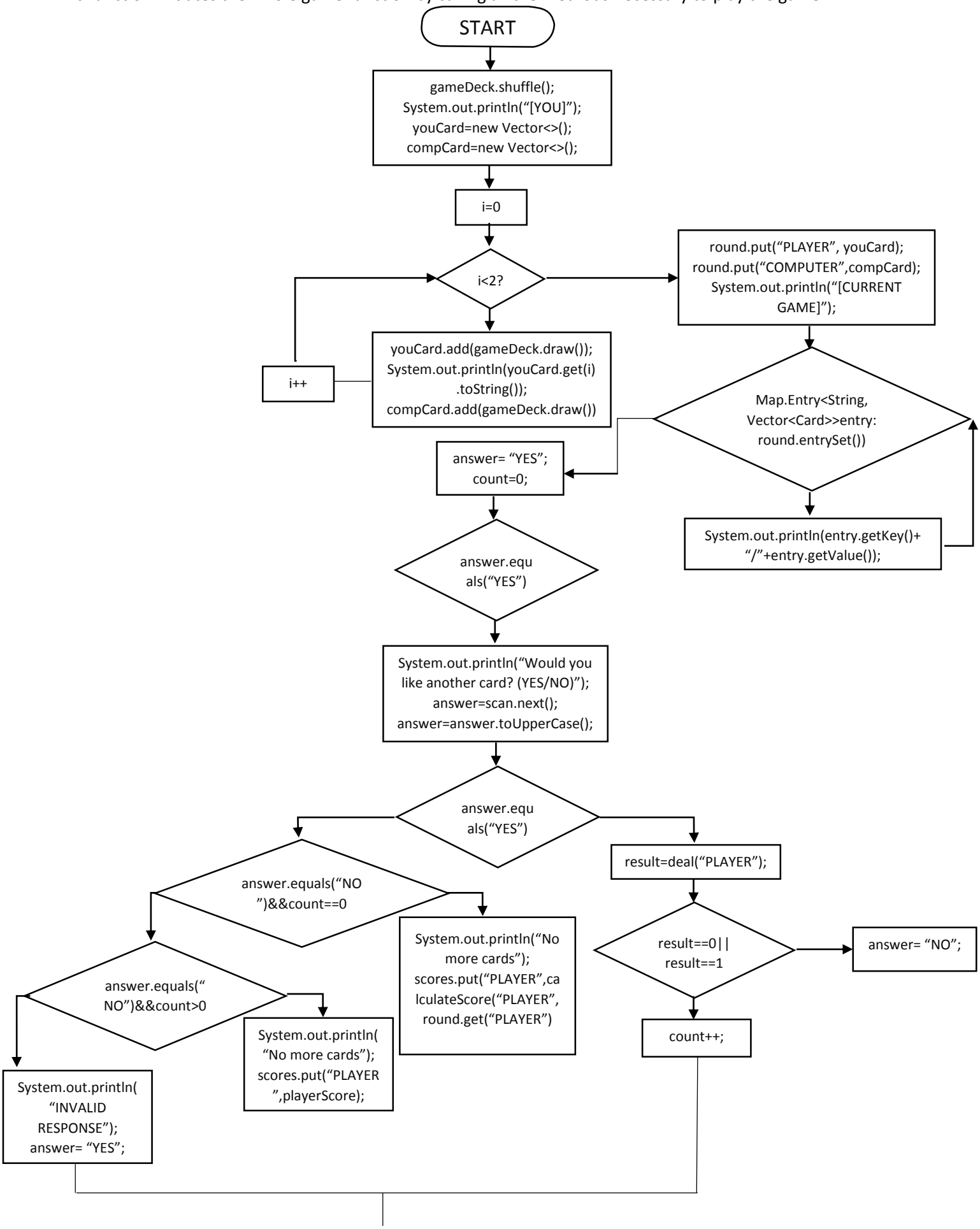
Function calculateScore(String name, Vector<Card>list)

This function calculates the total value of the cards in list and returns total value. If one of the cards in the vector is an Ace, the system will ask the user if the value of that card should be 1 or 11 and then input the appropriate value into total

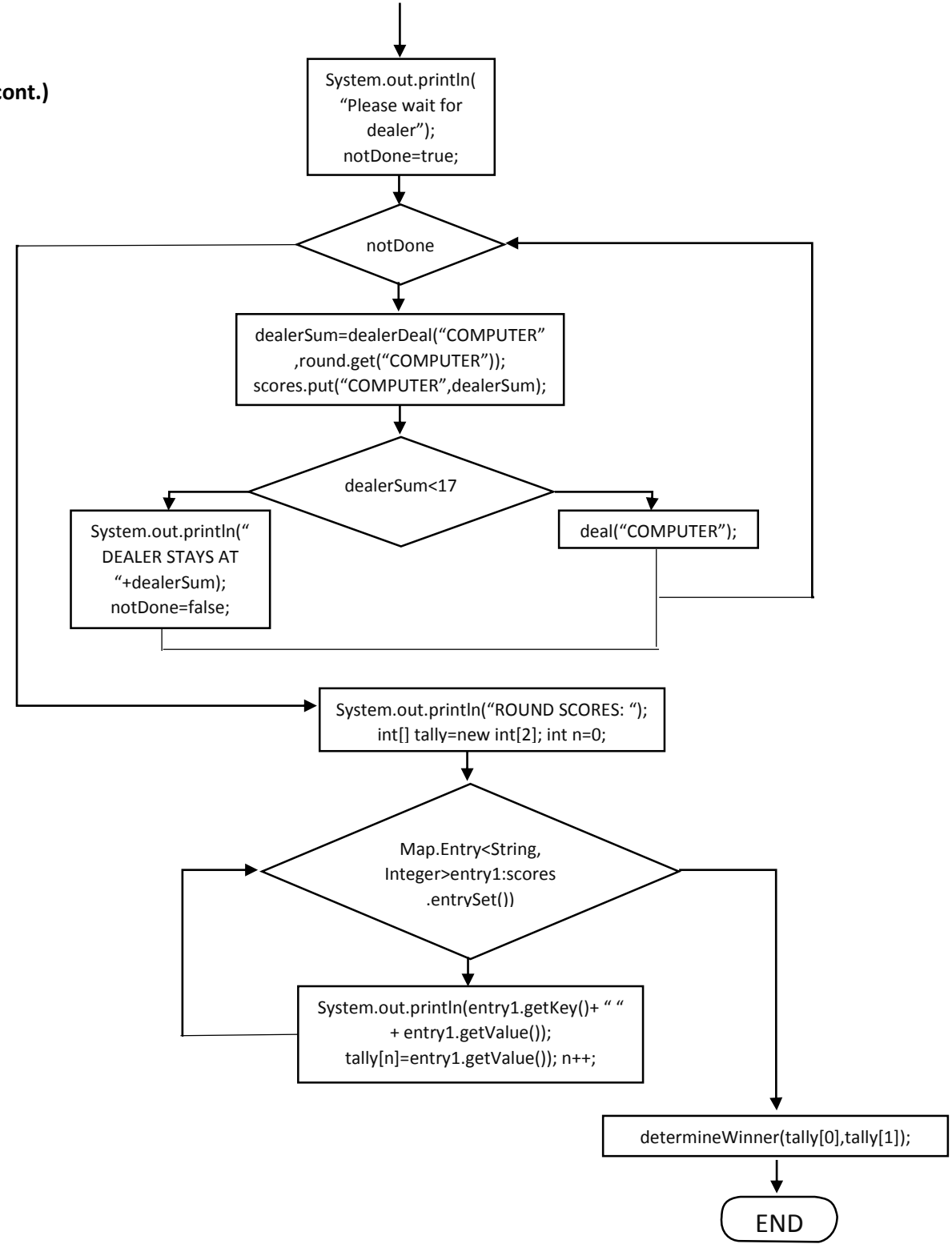


Function start()

This function initiates the whole game function by calling all the methods necessary to play the game



(cont.)


System.out.println("Please wait for dealer"); notDone=true;
notDone
dealerSum=dealerDeal("COMPUTER" ,round.get("COMPUTER")); scores.put("COMPUTER",dealerSum);
dealerSum<17
System.out.println(" DEALER STAYS AT "+dealerSum); notDone=false;
deal("COMPUTER");
System.out.println("ROUND SCORES: "); int[] tally=new int[2]; int n=0;
Map.Entry<String, Integer>entry1:scores .entrySet())
System.out.println(entry1.getKey()+ " " + entry1.getValue()); tally[n]=entry1.getValue()); n++;
determineWinner(tally[0],tally[1]);
END

Pseudocode

Initialize

Call shuffle() method

Declare HashMap for scores, Card

While size of player's hand and dealer's hand is not 2

    Call draw() method for the player

    Display the Card that is drawn

    Call draw() method for the dealer

Display the current hand of the player (user) and the dealer's hand

Display question of whether to add another card to user's hand

Get user's input

If input is "YES"

    Call deal() method

Else if "NO"

    Call calculateScore method to calculate scores for player

Else

    Display "Invalid Response"

Declare notDone to true

While(notDone)

    Set dealerSum to value returned by calling dealerDeal() method

    Put dealerSum into scores HashMap

If(dealerSum<17)

    Call deal() method for dealer

Else

    Display dealerSum

    Set notDone to false

Display Round Scores

Declare int[] tally of size 2, int n to 0

Go through each value in scores Hashmap and put value in tally array

Set n++

Display winner

Major Variables

Type	Variable Name	Description	Location
String	suit	suit of the Card	setSuit(), class Card
	number	String version of number value of Card	setNumber(), class Card
	hashCode	hash value of Card	Card(), class Card
int	cardNum	Numerical version of number value of Card	Card(), class Card
	suitNum	Numerical version of suit of Card	Card(), class Card
	cardsInDeck	Number of cards currently in deck	Deck(), class Deck
	playerScore	Score of the player	deal(), class Game
	dealerScore	Score of the dealer	deal(), class Game
Stack<Card>	newStack	Stack containing 52 cards	shuffle(), class Deck
Stack<String>	hashStack	Stack containing hash code of 52 cards	shuffle(), class Deck
HashMap<String,Vector<Card>>	round	Key: type of player, Value: vector of cards that player currently holds	start(), class Game
HashMap<String, Integer>	scores	Key: type of player, Value: total card value of the hand	start(), class Game
Deck	gameDeck	Object that holds the cards in a deck, hash-value of the cards in deck, and the size of the deck	start(), class Game
Scanner	scan	Object that reads input from user	start(), class Game
Game	newGame	Object that will hold the game	main()

Java Constructs

Chapter	New Syntax and Keywords	Location
2	Arrays	int[] tally=new int[2]; tally[n]=entry1.getValue();
3	Remove keyword	hashStack.remove(hashStack.size()-1);
	Add keyword	newCards.add(c);
5	Stacks	Stack<Card> newStack=new Stack<>(); Stack<String>hashStack=new Stack<>();
6	Vectors	Vector<Card>newCards=round.getName(name); Vector<Card> youCard=new Vector<>(); Vector<Card> compCard=new Vector<>();
18	Searching Concept	If(tempCard.getNumber().equals(“Ace”)) If(!list.get(i).getNumber().equals(“ACE”))
21	Hashing	hashCode=hash(suit,number); public String hash(String s, String n)

Reference

1. Textbook
2. API documentation (JAVA SE7 Platform API Specification)

Program

```
/*
 * Author: Tiffany Lin
 * Date: 05/20/2015
 * Project Game Logic: Game Class
 * Description: Create a BlackJack Game between user and computer dealer
 */
import java.util.HashMap;
import java.util.Vector;
import java.util.Map;
import java.util.Scanner;
public class Game{
	HashMap<String, Vector<Card>> round = new HashMap<String, Vector<Card>>();
	HashMap<String, Integer> scores = new HashMap<String, Integer>();
	Deck gameDeck;
	static int playerScore=0;
	static int dealerScore=0;
	static Scanner scan=new Scanner(System.in);
	public Game(){
		gameDeck=new Deck();
	}
	public int deal(String name){//if player p chooses to add another card to hand
			Vector<Card> newCards=round.get(name);//grab the current hand of player
name
			Card c=gameDeck.draw();
			System.out.println("Picked up: "+c.toString());
			newCards.add(c);//add the new randomly picked card to the current hand
			round.put(name, newCards);//put new card into existing hand
			if(name.equals("PLAYER")){
				playerScore = calculateScore(name, newCards);
				if(playerScore>21){//if total card value is over 21
					System.out.println("Sorry! You're now over 21. You're
out.");
					scores.put(name, playerScore);
					return 0;
				}else if(playerScore==21){//if exactly 21
					System.out.println("Congrats! You got Blackjack!");
					round.put(name, newCards);
					scores.put(name, playerScore);
					return 1;
				}else{//if under 21
					round.put(name, newCards);
					scores.put(name, playerScore);
					return -1;
				}
			}else if(name.equals("COMPUTER")){
				dealerScore=dealerDeal(name, newCards);
				if(dealerScore>21){//if total card value is over 21
					System.out.println("Dealer is now over 21. OUT.");
					scores.put(name, dealerScore);
					return 0;
				}else if(dealerScore==21){//if exactly 21
					System.out.println("Dealer has Blackjack!");
					round.put(name, newCards);
					scores.put(name, dealerScore);
					return 1;
				}else{//if under 21
					round.put(name, newCards);
					scores.put(name, dealerScore);
					return -1;
				}
			}
```

```
                return -1;
        }
        public int calculateScore(String name, Vector<Card>list){
                int totalValue=0;
                int answer=1;
        //size is dynamically changed
                for(int i=0;i<list.size();i++){
                        Card tempCard=list.get(i);
                        if(tempCard.getNumber().equals("Ace")){
                                System.out.println("One of your cards is Ace. Would you like the
Ace to be a value of 1 or 11?");
                                answer=scan.nextInt();
                                if(answer==11){
                                        totalValue+=11;
                                        //need to set the ACE in list equal to 11
                                        tempCard.setNumber(11);
                                        list.set(i, tempCard);
                                }else if(answer==1){
                                        totalValue+=1;
                                        tempCard.setNumber(1);
                                        list.set(i, tempCard);
                                }else{
                                        System.out.println("INVALID ANSWER. Can only be 1 or 11.
Automatic assignment of 1");
                                        totalValue+=tempCard.getCardNumber();
                                        list.set(i,tempCard);
                                }
                        }else{
                                totalValue+=tempCard.getCardNumber();
                        }
                }//put totalValue into scores
                scores.put(name,totalValue);
                return totalValue;
        }
        public int dealerDeal(String name, Vector<Card>list){
                int totalValue=0;
                int acePosition=0;
                boolean present=false;
                for(int i=0;i<list.size();i++){
                        if(!list.get(i).getNumber().equals("Ace")){
                                totalValue+=list.get(i).getCardNumber();
                        }else{
                                acePosition=i;
                                present=true;
                        }
                }
                Card tempCard=new Card();
                if (totalValue<=10&&present==true){
                        tempCard.setNumber(11);
                        list.set(acePosition, tempCard);
                        totalValue+=11;
                }else if(totalValue>10&& present==true){
                        tempCard.setNumber(1);
                        list.set(acePosition,tempCard);
                        totalValue+=1;
                }
                return totalValue;
        }
        public void determineWinner(int pScore, int dScore){
                if(pScore==21 && dScore!=21){//player has hit 21 and dealer did not
                        System.out.println("CONGRATS! YOU HAVE WON OVER THE DEALER!");
                }else if(dScore==21 && pScore!=21){//dealer has hit 21 and player did not
                        System.out.println("SORRY! YOU LOST! DEALER WON");
```

```
            }else if(dScore>21&&pScore<=21){//dealer went over 21 and player is still
under or at 21
                  System.out.println("CONGRATS! YOU HAVE WON OVER THE DEALER!");
            }else if(pScore>21 &&dScore<=21){//if player went over 21 and dealer is still
under or at 21
                  System.out.println("SORRY! YOU LOST! DEALER WON");
            }else if(pScore<dScore && dScore<21){//both under 21 & if dealer's score is
higher than player score
                  System.out.println("SORRY! YOU LOST! DEALER WON");
            }else if(dScore<pScore && pScore<21){// both under 21 & if player's score is
higher than dealer score
                  System.out.println("CONGRATS! YOU HAVE WON OVER THE DEALER!");
            }else if(dScore==pScore){
                  if(round.get("PLAYER").size()>round.get("COMPUTER").size()){
                        System.out.println("CONGRATS! YOU HAVE WON OVER THE DEALER!");
                  }else if(round.get("PLAYER").size()<round.get("COMPUTER").size()){
                        System.out.println("SORRY! YOU LOST! DEALER WON");
                  }else{
                        System.out.println("DEALER AND PLAYER TIED.");
                  }
            }else{//both out
                  System.out.println("BOTH LOST. CONSIDERED TIE.");
            }
      }
      public void start(){
            //shuffles deck
            gameDeck.shuffle();
            //add players to the game
            System.out.println("[YOU]");
            Vector<Card> youCard=new Vector<>();
            Vector<Card> compCard=new Vector<>();
            for(int i=0;i<2;i++){
                  youCard.add(gameDeck.draw());
                  System.out.println(youCard.get(i).toString());
                  compCard.add(gameDeck.draw());
            }
            round.put("PLAYER", youCard);
            round.put("COMPUTER", compCard);
            //print out current list of players with cards
            System.out.println("[CURRENT GAME]");
            for (Map.Entry<String, Vector<Card>> entry : round.entrySet())
            {
                System.out.println(entry.getKey() + "/" + entry.getValue());
            }
            //check to see if player wants to keep going***
            String answer="YES";
            int count=0;
            while(answer.equals("YES")){
                  System.out.println("Would you like to be dealt another card?(YES/NO)");
                  answer=scan.next();
                  answer=answer.toUpperCase();
                  if(answer.equals("YES")){
                        int result=deal("PLAYER");
                        if(result==0||result==1){
                              answer="NO";
                        }
                        count++;
                  }else if(answer.equals("NO")&&count==0){
                        System.out.println("You do not want more cards.");
                        scores.put("PLAYER",
calculateScore("PLAYER",round.get("PLAYER")));
                  }else if(answer.equals("NO")&&count>0){
                        System.out.println("You chose not to pick up anymore cards.");
```

```
                        scores.put("PLAYER", playerScore);
                  }else{
                        System.out.println("INVALID RESPONSE.");
                        answer="YES";
                  }
            }
            System.out.println("-------");
            System.out.println("Please wait for dealer...");
            System.out.println("-------");
            //COMPUTER'S TURN (DEALER)
            boolean notDone=true;
            while(notDone){
                  int dealerSum=dealerDeal("COMPUTER",round.get("COMPUTER"));
                  scores.put("COMPUTER",dealerSum);
                  if(dealerSum<17){
                        deal("COMPUTER");
                  }else{
                        System.out.println("DEALER STAYS AT "+dealerSum);
                        notDone=false;
                  }
            }
            //COMPARE PLAYER AND COMPUTER TO SEE WHO HAS THE HIGHER SCORE
            System.out.println("-------");
            System.out.println("ROUND SCORES: ");
            int[] tally=new int[2];int n=0;
            for(Map.Entry<String, Integer>entry1: scores.entrySet()){
                  System.out.println(entry1.getKey() + " "+entry1.getValue());
                  tally[n]=(entry1.getValue());
                  n++;
            }
            System.out.println("====================");
            determineWinner(tally[0],tally[1]);
      }
      public static void main(String[] args){
            System.out.println("WELCOME TO BLACKJACK GAME");
            System.out.println("========================");

            Game newGame=new Game();
            newGame.start();
      }
}


/*
 * Author: Tiffany Lin
 * Date: 05/20/2015
 * Project Game Logic: Deck Class
 * Description: Create a BlackJack Game between user and computer dealer
 */
import java.util.Stack;
public class Deck{
      Stack<Card> newStack=new Stack<>();
      Stack<String> hashStack=new Stack<>();
      int cardsInDeck;
      public Deck(){
            cardsInDeck=0;
            shuffle();
      }
      public void shuffle(){//resets deck and fills deck with cards
            Card newCard;
            while(cardsInDeck<52){//keep looping until we get deck all filled with cards
                  newCard=new Card();//pick a random card
                  if(newStack.size()==0){//if no cards in stack
```

```
                        newStack.push(newCard);
                        hashStack.push(newCard.getHash());
                        cardsInDeck++;
                  }else{//if more than 0 cards in stack
                        boolean present=false;
                        for(int i=0;i<hashStack.size();i++){
                              if(hashStack.elementAt(i).equals(newCard.getHash())){
                                    present=true;
                              }
                        }//if not found in hashstack
                        if(present==false){
                              newStack.push(newCard);
                              hashStack.push(newCard.getHash());
                              cardsInDeck++;
                        }
                  }
            }
      }
      public Card draw(){//pull the top card off of the stack of cards
            hashStack.remove(hashStack.size()-1);
            cardsInDeck--;
            return newStack.pop();
      }
      public void printCards(){
            for(int i=0;i<newStack.size();i++){
                  System.out.println(newStack.get(i).toString());
            }
      }
      public void printHash(){
            for(int i=0;i<hashStack.size();i++){
                  System.out.println(hashStack.get(i).toString());
            }
      }
}
```

```
/*
 * Author: Tiffany Lin
 * Date: 05/20/2015
 * Project Game Logic: Card Class
 * Description: Create a BlackJack Game between user and computer dealer
 */
import java.util.Random;
public class Card{
      private String suit;
      private String number;
      private String hashCode;
      private int cardNum;
      private int suitNum;
      public Card(){//pick a random card
            Random rand=new Random();
            suitNum=rand.nextInt(4)+1;
            cardNum=rand.nextInt(13)+1;
            setSuit();
            setNum();
            hashCode=hash(suit, number);
      }
      public void setSuit(){
                  if(suitNum==1){
```

```
                        suit="Diamonds";
                  }else if(suitNum==2){
                        suit="Clubs";
                        }else if(suitNum==3){
                        suit="Hearts";
                        }else if(suitNum==4){
                              suit="Spades";
                              }
      }
      public void setNumber(int n){
            cardNum=n;
      }
      public void setNum(){
            if(cardNum==13){
                  number="King";
                  cardNum=10;
                  }else if(cardNum==12){
                        number="Queen";
                        cardNum=10;
                        }else if(cardNum==11){
                              number="Jack";
                              cardNum=10;
                              }else if(cardNum==1){
                                    number="Ace";
                                    }else{//2-10
                                          number=String.valueOf(cardNum);
                                          }
      }
      public String getSuit(){return suit;}
      public String getNumber(){return number;}
      public String getHash(){return hashCode;}
      public int getSuitNumber(){return suitNum;}
      public int getCardNumber(){return cardNum;}
      public String hash(String s, String n){
            String code=s.substring(0,1)+n;//ex: H1 (Ace of Hearts) or S12 (Queen of
Spades)
            return code;
      }
      public String toString(){
            return number+" of "+suit;
      }
}
```