Intro to Java Week 6 Coding Assignment

Points possible: 70

Category	Criteria	% of Grade
Functionality	Does the code work?	25
Organization	Is the code clean and organized? Proper use of white space, syntax, and consistency are utilized. Names and comments are concise and clear.	25
Creativity	Student solved the problems presented in the assignment using creativity and out of the box thinking.	25
Completeness	All requirements of the assignment are complete.	25

Instructions: In Eclipse, or an IDE of your choice, write the code that accomplishes the objectives listed below. Ensure that the code compiles and runs as directed. Take screenshots of the code and of the running program (make sure to get screenshots of all required functionality) and paste them in this document where instructed below. Create a new repository on GitHub for this week's assignments and push this document, with your Java project code, to the repository. Add the URL for this week's repository to this document where instructed and submit this document to your instructor when complete.

Coding Steps:

For the final project you will be creating an automated version of the classic card game *WAR*.

1. Create the following classes.
    a. Card
        i. Fields
            1. **value** (contains a value from 2-14 representing cards 2-Ace)
            2. **name** (e.g. Ace of Diamonds, or Two of Hearts)
        ii. Methods
            1. Getters and Setters
            2. **describe** (prints out information about a card)
    b. Deck
        i. Fields
            1. **cards** (List of Card)
        ii. Methods

1. **shuffle** (randomizes the order of the cards)
2. **draw** (removes and returns the top card of the Cards field)
3. In the constructor, when a new Deck is instantiated, the Cards field should be populated with the standard 52 cards.

c. Player
   i. Fields
      1. **hand** (List of Card)
      2. **score** (set to 0 in the constructor)
      3. **name**
   ii. Methods
      1. **describe** (prints out information about the player and calls the describe method for each card in the Hand List)
      2. **flip** (removes and returns the top card of the Hand)
      3. **draw** (takes a Deck as an argument and calls the draw method on the deck, adding the returned Card to the hand field)
      4. **incrementScore** (adds 1 to the Player's score field)

2. Create a class called App with a main method.
3. Instantiate a Deck and two Players, call the shuffle method on the deck.
4. Using a traditional for loop, iterate 52 times calling the Draw method on the other player each iteration using the Deck you instantiated.
5. Using a traditional for loop, iterate 26 times and call the flip method for each player.
   a. Compare the value of each card returned by the two player's flip methods. Call the incrementScore method on the player whose card has the higher value.
6. After the loop, compare the final score from each player.
7. Print the final score of each player and either "Player 1", "Player 2", or "Draw" depending on which score is higher or if they are both the same.

Screenshots of Code:

```java
package finalProject;

public class App {

    public static void main(String[] args) {
        Player player1 = new Player("John");
        Player player2 = new Player("Mark");

        Deck gameDeck = new Deck();
        gameDeck.shuffleDeck();


        //dealing cards to players
        for(int i = 0; i < 52; i++) {
            Card tempCard = gameDeck.drawTopCard();


            if(i % 2 == 0) {
                player2.playerHand.add(tempCard);
            }
            else{
                player1.playerHand.add(tempCard);
            }

        }
```

```java

        //playing game
        for(int j = 0; j < 26; j++) {
            Card player1Card = player1.flipTopCard();
            Card player2Card = player2.flipTopCard();

            System.out.println("Round # : " + (j + 1));
            System.out.println("Player 1 plays: " + player1Card.describeCard() + " and Player 2 plays: " + player2Card.describeCard());


            if(player1Card.getValue() > player2Card.getValue()) {
                System.out.println("Player 1 wins round # " + (j+ 1));
                player1.incrementScore();
            }
            else if(player2Card.getValue() > player1Card.getValue()) {
                System.out.println("Player 2 wins round # " + (j + 1));
                player2.incrementScore();
            }
            System.out.println("\n");
        }

        System.out.println("Player 1 scored: " + player1.score + " points");
        System.out.println("Player 2 scored: " + player2.score + " points");

        if(player1.score > player2.score) {
            System.out.println("Player 1 is the winner!");
        }
        else if(player2.score > player1.score) {
            System.out.println("Player 2 is the winner!");
        }
        else {
            System.out.println("The game was a tie!");
        }

    }


}
```

```java
package finalProject;

import java.util.ArrayList;
import java.util.List;

public class Player {

    List<Card> playerHand = new ArrayList<Card>();
    int score;
    String playerName;

    //Constructor
    Player(String name){
        this.score = 0;
        this.playerName = name;

    }

    //describe method
    public void describePlayer() {
        System.out.println(playerName);
        System.out.println("\n");

        for(int i = 0; i < this.playerHand.size(); i++) {
            Card tempCard = this.playerHand.get(i);
            tempCard.describeCard();
        }
    }

    //flip card
    public Card flipTopCard(){
        Card topCard = this.playerHand.get(this.playerHand.size() - 1);
            this.playerHand.remove(this.playerHand.size() -1);

        return topCard;
    }

    //draw card
    public Card drawCardFromDeck() {
        Card newCard = this.playerHand.get(this.playerHand.size());

        return newCard;
    }

    //increment Score
    public void incrementScore() {
        this.score += 1;
    }

}
```

```java
package finalProject;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.List;

public class Deck {

    List<Card> deck = new ArrayList<>();

    List<String> suits = Arrays.asList("hearts", "diamonds", "spades", "clubs");
    List<String> values = Arrays.asList("2", "3", "4", "5", "6", "7", "8", "9", "10", "Jack", "Queen", "King", "Ace");

    //constructor
    Deck(){
        for(int i = 0; i < suits.size(); i++) {
            for(int j = 0; j < values.size(); j++) {
                String tempSuit = suits.get(i);
                String tempVal = values.get(j);

                Card tempCard = new Card(tempVal, tempSuit, j);

                deck.add(tempCard);
            }
        }

    }

    //shuffle
    public List<Card> shuffleDeck(){
        Collections.shuffle(deck);

        return deck;
    }

    //draw card
    public Card drawTopCard(){

        Card topCard = deck.get(deck.size() - 1);
        deck.remove(deck.size() - 1);

        return topCard;

    }


}
```

```java
package finalProject;

public class Card {

    int value;
    String name;

    //constructor
    public Card(String tempVal, String tempSuit, int j) {
        this.value = j;
        this.name = tempVal + " of " + tempSuit;
    }


    //describe card
    public String describeCard() {
        return this.name;
    }




        //getters and setters which the directions said to include, but it didn't say I had to use them.
    public int getValue() {
        return this.value;
    }

    public String getName() {
        return this.name;
    }

    public void setValue(int val) {
        this.value = val;
    }

    public void setName(String suit, String rank) {
        this.name = rank + " of " + suit;
    }


}
```

Screenshots of Running Application:

```
Round # : 1
Player 1 plays: 5 of hearts and Player 2 plays: 2 of diamonds
Player 1 wins round # 1


Round # : 2
Player 1 plays: 3 of spades and Player 2 plays: King of hearts
Player 2 wins round # 2


Round # : 3
Player 1 plays: Queen of hearts and Player 2 plays: Jack of diamonds
Player 1 wins round # 3


Round # : 4
Player 1 plays: King of clubs and Player 2 plays: 9 of spades
Player 1 wins round # 4


Round # : 5
Player 1 plays: 8 of spades and Player 2 plays: 6 of hearts
Player 1 wins round # 5


Round # : 6
Player 1 plays: 7 of clubs and Player 2 plays: 5 of diamonds
Player 1 wins round # 6


Round # : 7
Player 1 plays: 9 of clubs and Player 2 plays: 10 of hearts
Player 2 wins round # 7


Round # : 8
Player 1 plays: 8 of diamonds and Player 2 plays: 7 of diamonds
Player 1 wins round # 8


Round # : 9
Player 1 plays: King of spades and Player 2 plays: 2 of hearts
Player 1 wins round # 9


Round # : 10
Player 1 plays: Ace of spades and Player 2 plays: 4 of spades
Player 1 wins round # 10


Round # : 11
Player 1 plays: 6 of clubs and Player 2 plays: Jack of hearts
Player 2 wins round # 11


Round # : 12
Player 1 plays: 7 of spades and Player 2 plays: Ace of hearts
Player 2 wins round # 12
```

```
Round # : 13
Player 1 plays: Queen of spades and Player 2 plays: 10 of diamonds
Player 1 wins round # 13


Round # : 14
Player 1 plays: 8 of clubs and Player 2 plays: 4 of hearts
Player 1 wins round # 14


Round # : 15
Player 1 plays: 2 of clubs and Player 2 plays: 7 of hearts
Player 2 wins round # 15


Round # : 16
Player 1 plays: Queen of clubs and Player 2 plays: 9 of hearts
Player 1 wins round # 16


Round # : 17
Player 1 plays: Ace of diamonds and Player 2 plays: Queen of diamonds
Player 1 wins round # 17


Round # : 18
Player 1 plays: Ace of clubs and Player 2 plays: 3 of clubs
Player 1 wins round # 18


Round # : 19
Player 1 plays: 6 of spades and Player 2 plays: Jack of clubs
Player 2 wins round # 19


Round # : 20
Player 1 plays: 4 of clubs and Player 2 plays: 9 of diamonds
Player 2 wins round # 20


Round # : 21
Player 1 plays: 5 of clubs and Player 2 plays: 10 of spades
Player 2 wins round # 21


Round # : 22
Player 1 plays: 6 of diamonds and Player 2 plays: 5 of spades
Player 1 wins round # 22


Round # : 23
Player 1 plays: 3 of hearts and Player 2 plays: 2 of spades
Player 1 wins round # 23


Round # : 24
Player 1 plays: King of diamonds and Player 2 plays: 4 of diamonds
Player 1 wins round # 24
```

```
Round # : 25
Player 1 plays: 10 of clubs and Player 2 plays: 8 of hearts
Player 1 wins round # 25


Round # : 26
Player 1 plays: Jack of spades and Player 2 plays: 3 of diamonds
Player 1 wins round # 26


Player 1 scored: 18 points
Player 2 scored: 8 points
Player 1 is the winner!
```

URL to GitHub Repository:

https://github.com/Requ91/Week6