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:

Card Class:

Card.java | Deck.java | Player.java | App.java

```java
package warCardGame;

public class Card {

    private String name;
    private String ranks;
    private int value;

    //Constructor
    public Card(String name, String ranks, int value) {
        this.name = name;
        this.ranks = ranks;
        this.value = value;
    }


    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getValue() {
        return value;
    }
    public void setValue(int value) {
        this.value = value;
    }

    public String getRank() {
        return ranks;
    }

    public void setRank(String rank) {
        this.ranks = rank;
    }

    //toString to allow me to print the values
    @Override
    public String toString() {
        return name + " of " + ranks;
    }

    public String describe() {
        return this.toString();
    }
```

Deck Class:

Card.java | Deck.java | Player.java | App.java

```java
package warCardGame;

import java.util.ArrayList;

public class Deck {
    public static final String[] Suits = {"Clubs", "Diamonds", "Hearts", "Spades"};
    public static final String[] Ranks = {"2", "3", "4", "5", "6", "7", "8", "9", "10", "Jack", "Queen", "King", "Ace"};
    int value = 2;
    List<Card> deck = new ArrayList<Card>();

    //Constructor
    public Deck() {
        createDeck();
    }

    //method to build up a deck
    public void createDeck() {
        for(int i = 0; i < 4; i++) {
            for(int j = 0; j < 13; j++) {
                this.deck.add(new Card(Ranks[j], Suits[i], value++));
            } value = 2;
        }
    }

    public void shuffleCards() {
        Collections.shuffle(this.deck);
    }
```

```
    public Card draw() {
        Card topCard = this.deck.get(0);
        deck.remove(0);
        return topCard;
    }

    public List<Card> getDeck(){
        return deck;
    }

    public void setDeck(List<Card> deck) {
        this.deck = deck;
    }
//  @Override
//  public String toString() {
//      return Ranks + " of " + Suits;
//  }
}
```

Player Class:

```
package warCardGame;

import java.util.ArrayList;

public class Player {
    List<Card> hand = new ArrayList<Card>();
    int score;
    String name;

    public Player(String name) {
        this.score = 0;
        this.name = name;
}
    public void describe() {
        System.out.println(name + " " + "has a score of " + score + "!");
            for(Card card : hand) {
                card.describe();
            }
    }

    public Card flip() {
        Card topCard = this.hand.get(0);
        hand.remove(0);
        return topCard;
    }

    public void draw(Deck deck) {
        this.hand.add(deck.draw());
    }


   public void incrementScore() {
       this.score++;
   }
}
```

App Class:

Card.java Deck.java Player.java App.java

```java
package warCardGame;

public class App {

    public static void main(String[] args) {
    Deck newDeck = new Deck();

    newDeck.shuffleCards();
    Player p1 = new Player("Harvey");
    Player p2 = new Player("Dee");


    for(int i = 1; i < 26; i++) {
        p1.hand.add(newDeck.draw());
        p2.hand.add(newDeck.draw());
    }

    p1.describe();
    p2.describe();

    for(int i = 1; i < 26; i++) {
        Card p1Card = p1.flip();
        Card p2Card = p2.flip();
        if(p1Card.getValue() > p2Card.getValue()) {
            p1.incrementScore();
        } else if(p2Card.getValue() > p1Card.getValue()) {
            p2.incrementScore();
        }


    }
    if(p1.score > p2.score) {
        System.out.println(p1.name + " has a total score of " + p1.score);
        System.out.println(p2.name + " has a total score of " + p2.score);
        System.out.println("The winner today is " + p1.name + "!");
    } else if(p2.score > p1.score) {
        System.out.println(p2.name + " has a total score of " + p2.score);
        System.out.println(p1.name + " has a total score of " + p1.score);
        System.out.println("The winner today is " + p2.name + "!");
    } else {
        System.out.println(p1.name + " has a total score of " + p1.score);
        System.out.println(p2.name + " has a total score of " + p2.score);
        System.out.println("No one is a winner today, the final score is a Draw!");
    }
}
}
```

Screenshots of Running Application:

Problems Javadoc Declaration Console

```
<terminated> App (3) [Java Application] C:\Program Files\Java\jdk-11.0.13\bin\javaw.exe  (Jul 17, 2022, 8:21:56 AM – 8:21:56 AM) [pid: 18576]
Harvey has a score of 0!
Dee has a score of 0!
Harvey has a total score of 13
Dee has a total score of 11
The winner today is Harvey!
```

URL to GitHub Repository:

https://github.com/harvisd/JavaWeek6FinalProject