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

```java
package week6Assignment;

//Step 1.a Setting up everything for the card class.
// Getters, setters, and description.

class Card {

    private static int value;
    private static String name;


    // recipe here???

    public Card(int i, String string) {
        value = i;
        name = string;
    }

    //step 1.a.II
    public String getName()
    {
        return name;
    }

    public void setName(String name)
    {
        Card.name= name;
    }

    public static int getValue()
    {
        return value;
    }

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

    //step 1.a.III
    public void description()
    {
        System.out.println("Card is "+name);
        System.out.println("The value of the card is "+value);
    }


}

```

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

```java
package week6Assignment;
import java.util.Collections;
import java.util.List;

//Step 1.b
public class Deck {

    //step1.b.I&II Creating a list of cards
    //There's probably an easier way to do this, I'll have to come back later.

    static List<Card> deck;
    Card card1= new Card(2,"2 of Diamonds");
    Card card2= new Card(3,"3 of Diamonds");
    Card card3= new Card(4,"4 of Diamonds");
    Card card4= new Card(5,"5 of Diamonds");
    Card card5= new Card(6,"6 of Diamonds");
    Card card6= new Card(7,"7 of Diamonds");
    Card card7= new Card(8,"8 of Diamonds");
    Card card8= new Card(9,"9 of Diamonds");
    Card card9= new Card(10,"10 of Diamonds");
    Card card10= new Card(11,"Jack of Diamonds");
    Card card11= new Card(12,"Queen of Diamonds");
    Card card12= new Card(13,"King of Diamonds");
    Card card13= new Card(14,"Ace of Diamonds");
    Card card14= new Card(2,"2 of Spades");
    Card card15= new Card(3,"3 of Spades");
    Card card16= new Card(4,"4 of Spades");
    Card card17= new Card(5,"5 of Spades");
    Card card18= new Card(6,"6 of Spades");
    Card card19= new Card(7,"7 of Spades");
    Card card20= new Card(8,"8 of Spades");
    Card card21= new Card(9,"9 of Spades");
    Card card22= new Card(10,"10 of Spades");
    Card card23= new Card(11,"Jack of Spades");
    Card card24= new Card(12,"Queen of Spades");
    Card card25= new Card(13,"King of Spades");
    Card card26= new Card(14,"Ace of Spades");
    Card card27= new Card(2,"2 of Clubs");
    Card card28= new Card(3,"3 of Clubs");
    Card card29= new Card(4,"4 of Clubs");
    Card card30= new Card(5,"5 of Clubs");
    Card card31= new Card(6,"6 of Clubs");
    Card card32= new Card(7,"7 of Clubs");
    Card card33= new Card(8,"8 of Clubs");
    Card card34= new Card(9,"9 of Clubs");
    Card card35= new Card(10,"10 of Clubs");
    Card card36= new Card(11,"Jack of Clubs");
    Card card37= new Card(12,"Queen of Clubs");
    Card card38= new Card(13,"King of Clubs");
    Card card39= new Card(14,"Ace of Clubs");
    Card card40= new Card(2,"2 of Hearts");
```

```
    Card card39= new Card(14,"Ace of Clubs");
    Card card40= new Card(2,"2 of Hearts");
    Card card41= new Card(3,"3 of Hearts");
    Card card42= new Card(4,"4 of Hearts");
    Card card43= new Card(5,"5 of Hearts");
    Card card44= new Card(6,"6 of Hearts");
    Card card45= new Card(7,"7 of Hearts");
    Card card46= new Card(8,"8 of Hearts");
    Card card47= new Card(9,"9 of Hearts");
    Card card48= new Card(10,"10 of Hearts");
    Card card49= new Card(11,"Jack of Hearts");
    Card card50= new Card(12,"Queen of Hearts");
    Card card51= new Card(13,"King of Hearts");
    Card card52= new Card(14,"Ace of Hearts");

    //step1.b.II  Adding shuffle method. Not too sure about this part, ask mentor.

    public static void shuffle(Deck deck1) {
    Collections.shuffle(deck);
    }


    //step1.c.III

    public Card draw() {
        Card drawnCard = deck.get(0);
        deck.remove(drawnCard);
        return drawnCard;
    }

    public static void createPlayersHands(Deck deck, Player player1, Player player2) {
        for (int i =0; i <52; i++) {
            if (i % 2 ==0)
                player1.draw(deck);
        }  {
            player2.draw(deck);
        }

    }


}
```

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

```java
package week6Assignment;

import java.util.ArrayList;

//Step 1.c
public class Player {

    //1.c.I.1 Creating a list of cards for player hand.
    //Also not sure if this is correct. Ask mentor.

    ArrayList<Card> hand = new ArrayList<>();

    //1.c.I.2&3
    int score = 0;
    String name;

    //1.c.II.1




    public void describe() {

        System.out.println("============================================================");
        System.out.println(name+" Player Details as Follows");
        System.out.println("Player Score:"+score);
        System.out.println("Player Hand is as follows");
        System.out.println("------------------------------------------------------------");

        for (Card card : hand) {
            card.description();
        }

        System.out.println("============================================================");
    }

    //1.c.II.2

    public Card flip(ArrayList<Card> hand) {
        return hand.remove(0);
    }

    //1.c.II.3
    //Unable to figure out how to call description of card in player hand. Ask mentor.

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

    //1.d.II.4 Super easy score-implement
    public void incrementScore() {
```

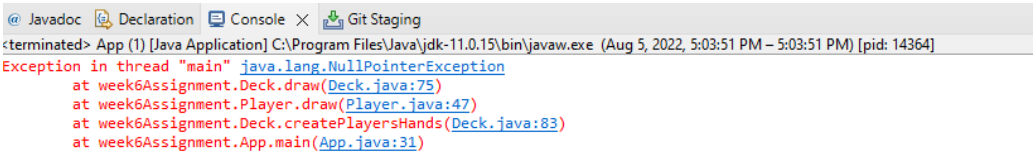
```
    public void incrementScore() {
        score++;
    }

    public Object handValue() {
        Object[]obArr = hand.toArray();
        Integer[]intArr = new Integer[obArr.length];
        return intArr[0];
    }

}

```

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

```
package week6Assignment;

public class App {

    //Step 2.

    public static void main(String[] args) {



        String winner = null; //For steps 6/7.

        //Step 3.
        Player player1 = new Player();
        player1.name = "Alexandra";


        Player player2 = new Player();
        player2.name = "Sophia";


        Deck deck1 = new Deck();



        //Does this even work? Depends on what instructor says about shuffle in Deck class.
        Deck.shuffle(deck1);

        //Step 4. Drawing 42 times to each player. I also have no idea what I'm doing here. Need to fix draw in deck first.

        Deck.createPlayersHands(deck1, player1, player2);

        //Step 5. Gameplay. This works. Kind of. Check if flip is doing what flip does.


        for (int i = 0; i < 26; i++) {

            Object[]obArr1 = player1.hand.toArray();
            Integer[]intArr1 = new Integer[obArr1.length];
            Object[]obArr2 = player1.hand.toArray();
            Integer[]intArr2 = new Integer[obArr2.length];

            if (intArr1[0] != intArr2[0]) {
                if (intArr1[0] > intArr2[0]) {
                    player1.incrementScore();
                }
                else {
                    player2.incrementScore();
                }

```

```
51              player1.flip(player1.hand);
52              player2.flip(player2.hand);
53              }
54          }
55
56          //step6/7. This works, don't touch!
57          System.out.println("*****************");
58          System.out.println("***Final Score***");
59          System.out.println("*****************");
60          System.out.println("Player 1: "+player1.score);
61          System.out.println("Player 2: "+player2.score);
62
63          if (player1.score != player2.score) {
64              if (player1.score > player2.score) {
65                  winner += "Player 1";
66              }
67              else {
68                  winner += "Player 2";
69              }
70          }
71          else {
72              winner += "Draw";
73          };
74          System.out.println("The winner is:");
75          System.out.println(winner);
76      }
77
78  }
79
```

Screenshots of Running Application:



I'm well aware my code doesn't work. I've received no errors in the Eclipse itself and I've run out of time to de-bug before Tuesday next week, which if I fixed the errors my score would be 70 from being turned in late. Though if I turn it in now with broken code, I get a 75 without functionality but not being late.

On a personal level, I am committed to returning to this to ensure it works, but for now I can't afford it.

URL to GitHub Repository:

https://github.com/AshenedPheonix/week6assignmentWIP