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.

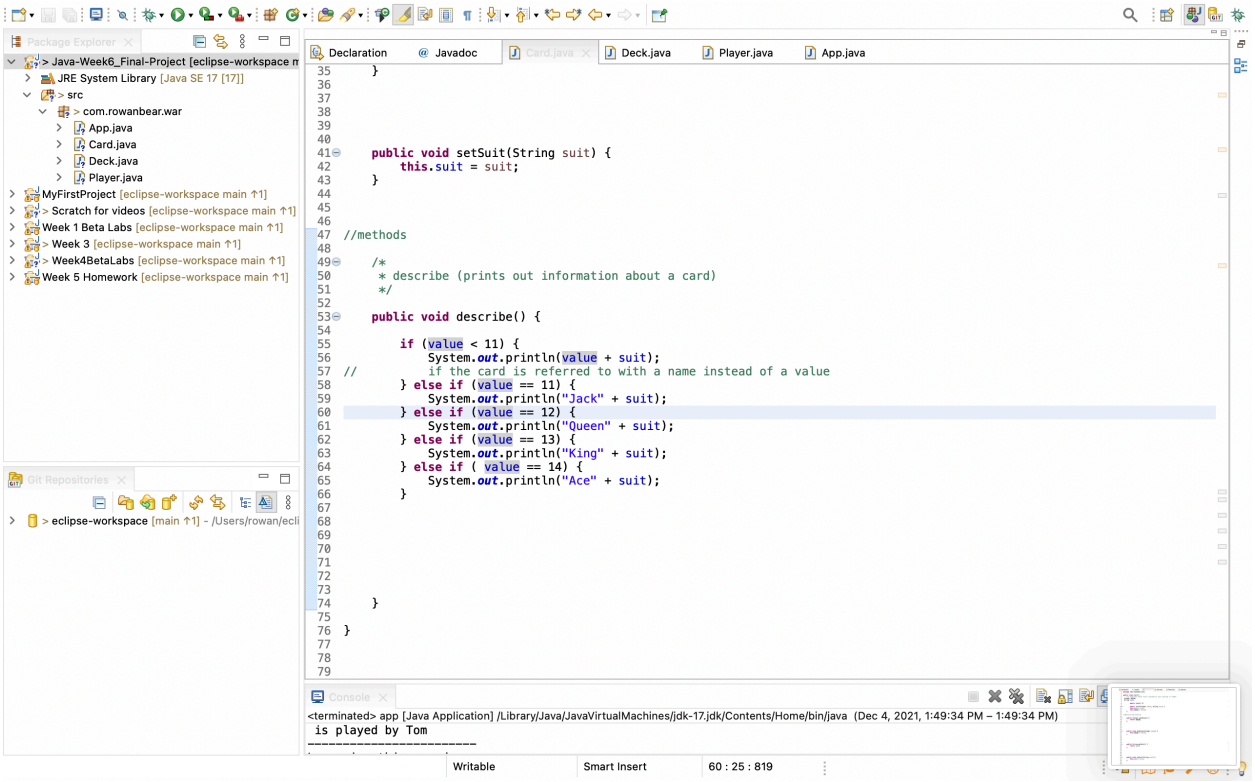
a. Compare the value of each card returned by the two player's flip methods. Call the incrementScore method on the player whose card has the higher value.

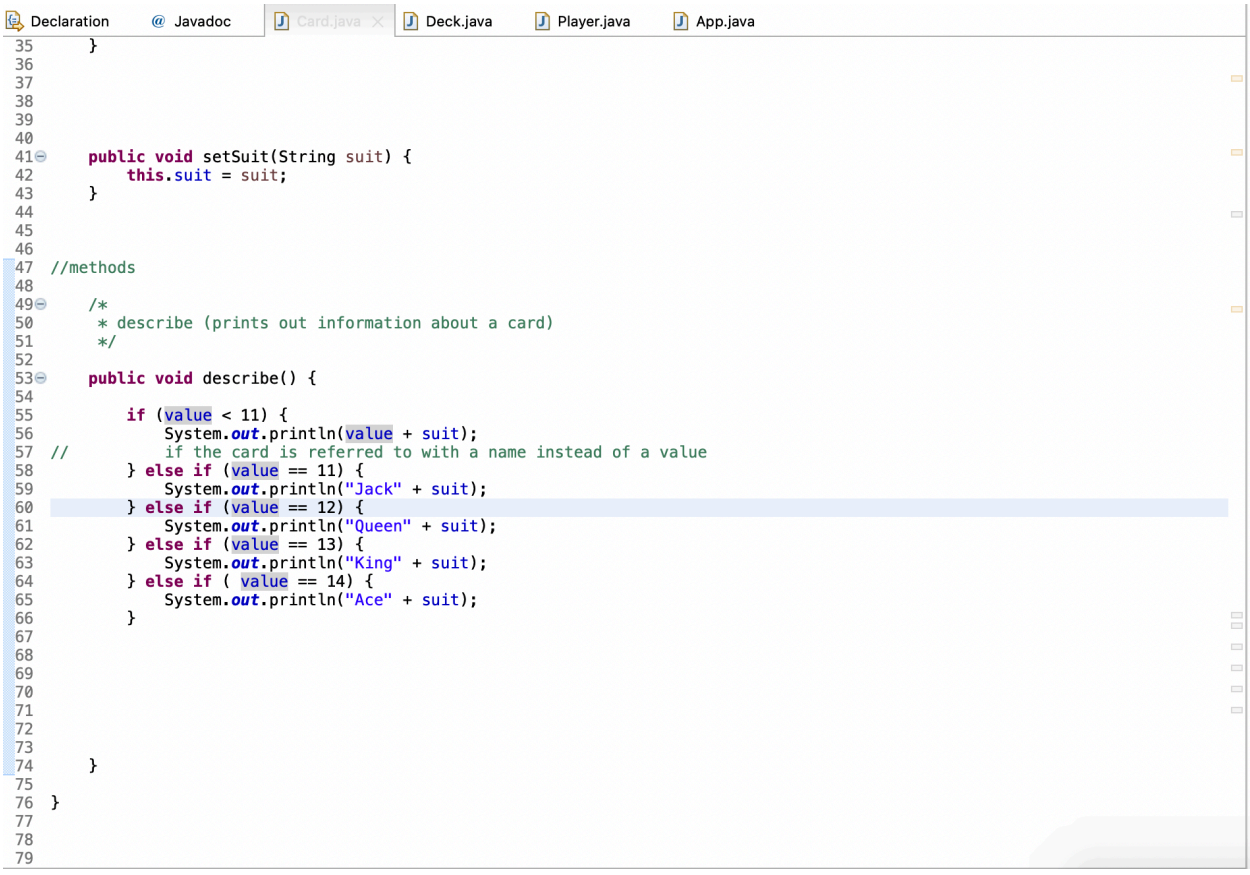
6. After the loop, compare the final score from each player.
7. Print the final score of each player and either "Player 1", "Player 2", or "Draw" depending on which score is higher or if they are both the same.

Screenshots of Code:

Declaration | Javadoc | Card.java | Deck.java | Player.java | App.java

```java
package com.rowanbear.war;

public class Card {
    //I used the field "suit" instead of just calling it "name"
 Integer value;
 String suit;

        public Card() {}

        public Card(Integer value, String suit) {
        this.suit = suit;
        this.value = value;
    }

//getters and setters

    public Integer getValue() {
        return value;
    }




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




    public String getSuit() {
        return suit;
    }




    public void setSuit(String suit) {
        this.suit = suit;
    }

```


Package Explorer
> Java-Week6_Final-Project [eclipse-workspace m
JRE System Library [Java SE 17 [17]]
> src
> com.rowanbear.war
App.java
Card.java
Deck.java
Player.java
MyFirstProject [eclipse-workspace main ↑1]
> Scratch for videos [eclipse-workspace main ↑1]
Week 1 Beta Labs [eclipse-workspace main ↑1]
> Week 3 [eclipse-workspace main ↑1]
> Week4BetaLabs [eclipse-workspace main ↑1]
Week 5 Homework [eclipse-workspace main ↑1]
Git Repositories
> eclipse-workspace [main ↑1] - /Users/rowan/ecli
Declaration
Javadoc
Card.java
Deck.java
Player.java
App.java
35 }
36
37
38
39
40
41 public void setSuit(String suit) {
42 this.suit = suit;
43 }
44
45
46
47 //methods
48
49 /*
50 * describe (prints out information about a card)
51 */
52
53 public void describe() {
54
55 if (value < 11) {
56 System.out.println(value + suit);
57 // if the card is referred to with a name instead of a value
58 } else if (value == 11) {
59 System.out.println("Jack" + suit);
60 } else if (value == 12) {
61 System.out.println("Queen" + suit);
62 } else if (value == 13) {
63 System.out.println("King" + suit);
64 } else if (value == 14) {
65 System.out.println("Ace" + suit);
66 }
67
68
69
70
71
72
73
74 }
75
76 }
77
78
79
Console
<terminated> app [Java Application] /Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java (Dec 4, 2021, 1:49:34 PM – 1:49:34 PM)
is played by Tom
Writable
Smart Insert
60 : 25 : 819


Declaration
Javadoc
Card.java
Deck.java
Player.java
App.java
35 }
36
37
38
39
40
41 public void setSuit(String suit) {
42 this.suit = suit;
43 }
44
45
46
47 //methods
48
49 /*
50 * describe (prints out information about a card)
51 */
52
53 public void describe() {
54
55 if (value < 11) {
56 System.out.println(value + suit);
57 // if the card is referred to with a name instead of a value
58 } else if (value == 11) {
59 System.out.println("Jack" + suit);
60 } else if (value == 12) {
61 System.out.println("Queen" + suit);
62 } else if (value == 13) {
63 System.out.println("King" + suit);
64 } else if (value == 14) {
65 System.out.println("Ace" + suit);
66 }
67
68
69
70
71
72
73
74 }
75
76 }
77
78
79

Declaration | Javadoc | Card.java | Deck.java | Player.java | App.java

```java
package com.rowanbear.war;

import java.util.ArrayList;

import java.util.Collections;

public class Deck {

    ArrayList<Card> cards;

    public Deck() {

    this.cards = new ArrayList<Card>();


                    //creates ArrayList [2, ... 15]
                    ArrayList<Integer> cardValue = new ArrayList<Integer>();
                    for (int i = 2; i < 15; i++) {
                        cardValue.add(i);
                    }

                    //In the constructor, when a new Deck is instantiated, the Cards field should be populated with the standard 52 cards.


                    //to make hearts cards
                    for (Integer i : cardValue) {
                            Card card = new Card(i, " Of Hearts" );
                            cards.add(card);
                    }

                    //to make spade cards
                    for (Integer i : cardValue) {
                        Card card = new Card(i, " Of Spades" );
                        cards.add(card);
                }

                    //to make diamond cards
                    for (Integer i : cardValue) {
                        Card card = new Card(i, " Of Diamonds" );
                        cards.add(card);
                }

                    //to make clubs cards
                    for (Integer i : cardValue) {
```

Declaration | Javadoc | Card.java | Deck.java | Player.java | App.java

```java
                    for (Integer i : cardValue) {
                        Card card = new Card(i, " Of Diamonds" );
                        cards.add(card);
                }

                    //to make clubs cards
                    for (Integer i : cardValue) {
                        Card card = new Card(i, " Of Clubs" );
                        cards.add(card);
                }




    }

    //methods

/*
 * shuffle (randomizes the order of the cards)
 *
 * draw (removes and returns the top card of the Cards field)

 */

    public void shuffle() {
        Collections.shuffle(cards);
    }


    public Card draw() {

        return cards.remove(0);

    }

    public void describe() {
        for (Card card : cards) {
            card.describe();
        }
    }


}
```

Declaration | Javadoc | Card.java | Deck.java | Player.java | App.java

```java
package com.rowanbear.war;

import java.util.ArrayList;

public class Player {

    ArrayList<Card> hand;

    int score;

    String name;

    public Player () {
        this.hand = new ArrayList<Card>();
        this.score = 0;
        this.name = "Default Name";
    }

    //getters and setters

    public ArrayList<Card> getHand() {
        return hand;
    }

    public void setHand(ArrayList<Card> hand) {
        this.hand = hand;
    }

    public int getScore() {
        return score;
    }

    public void setScore(int score) {
        this.score = score;
    }

    public String getName() {
        return name;
    }

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

    //methods
```

Declaration | Javadoc | Card.java | Deck.java | Player.java | App.java

```java
 * describe - prints out info about player and calls describe method for each card in Hand List
 *
 * flip - removes and returns the top card of the Hand
 *
 * draw - takes a Deck as an argument and calls the draw method on the deck, adding the returned Card to the hand field.
 *
 * incrementScore - adds a point to player's score field
 *
 *
 */

    public void describe() {
        System.out.println("This is " + this.name + ". \nThey have these cards in their hand: ");
        for (Card card : this.hand) {
            card.describe();
        }

        System.out.println(name + "'s current score is: " + score);


    }

    //flip method also prints out what is happening to the console to make it more exciting, like a real game!

    public Card flip() {
        Card card = hand.get(0);
        hand.remove(hand.get(0));
        System.out.println("-------------------------");
        card.describe();
        System.out.println(" is played by " + this.name + "\n-------------------------");

        return card;
    }

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

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


}
```

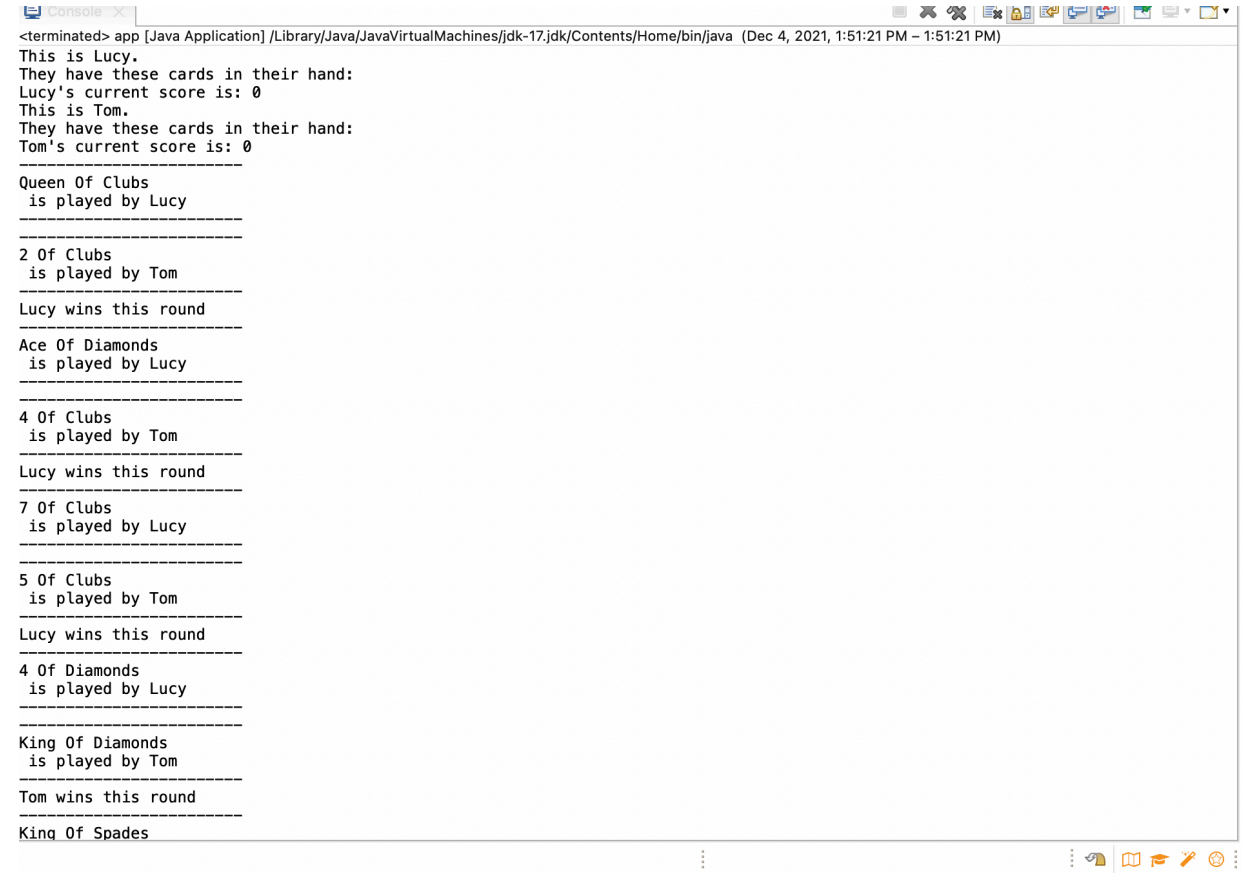
Declaration | Javadoc | Card.java | Deck.java | Player.java | App.java

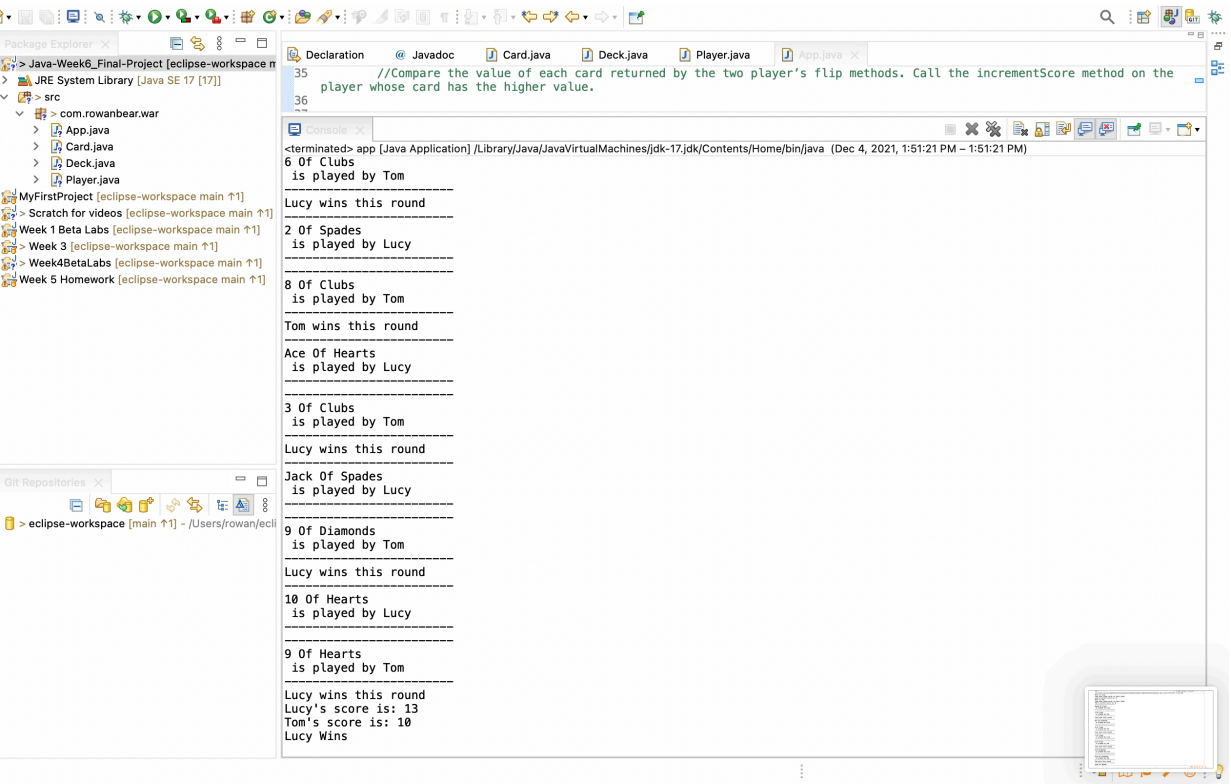
```java
package com.rowanbear.war;

public class App {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        //Instantiate a Deck and two Players, call the shuffle method on the deck.

        Deck exampleDeck = new Deck();

        exampleDeck.shuffle();

        Player playerOne = new Player();
        playerOne.name = "Lucy";
        playerOne.describe();


        Player playerTwo = new Player();
        playerTwo.name = "Tom";
        playerTwo.describe();

        // Using a traditional for loop, iterate 52 times calling the Draw method on the other player each iteration using the Deck you instantiated.

        for (int i = 2; i < 54; i++) {
            if (i%2 == 0) {
                playerOne.draw(exampleDeck);
            } else {
                playerTwo.draw(exampleDeck);
            }
        }

        //Using a traditional for loop, iterate 26 times and call the flip method for each player.

        //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.


        for (int i = 2; i < 28; i++) {


            Card card1 = playerOne.flip();
            Card card2 = playerTwo.flip();

```

Declaration | Javadoc | Card.java | Deck.java | Player.java | *App.java

```java
        //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.


        for (int i = 2; i < 28; i++) {


            Card card1 = playerOne.flip();
            Card card2 = playerTwo.flip();

            if (card1.value > card2.value) {
                System.out.println(playerOne.name + " wins this round");
                playerOne.incrementScore();
            } else if (card1.value < card2.value) {
                System.out.println(playerTwo.name + " wins this round");
                playerTwo.incrementScore();
            } else if (card1.value == card2.value){
                System.out.println("Draw: Neither player wins this round.");

            }

        }

            //After the loop, compare the final score from each player.
            //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.

            System.out.println(playerOne.name + "'s score is: " + playerOne.score);

            System.out.println(playerTwo.name + "'s score is: " + playerTwo.score);


            if (playerOne.score > playerTwo.score) {
                System.out.println(playerOne.name + " Wins");
            } else if (playerOne.score < playerTwo.score) {
                System.out.println(playerTwo.name + " Wins");
            } else if (playerOne.score == playerTwo.score) {
                System.out.println("Draw");
            }


    }

}
```

Screenshots of Running Application:





URL to GitHub Repository: https://github.com/Rowan-Bear/Java-Week6_Final-Project.git