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
        iii.
            1. **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 | Suit.java | Value.java

```java

public class Card {

    // fields
    private Value value;
    private Suit suit;
    private String name;


    // constructor
    public Card(Value value, Suit suit) {
        this.value = value;
        this.suit = suit;
        name = value + " of " + suit;
    }


    // methods
    public void describe() {
        System.out.println(name);
    }


    // getters/setters
    public Value getValue() {
        return value;
    }
    public void setValue(Value value) {
        this.value = value;
    }
    public Suit getSuit() {
        return suit;
    }
    public void setSuit(Suit suit) {
        this.suit = suit;
    }
}

```

Card.java | Deck.java | Player.java | App.java | Suit.java | Value.java

```java

public enum Value {
    TWO, THREE, FOUR, FIVE, SIX, SEVEN, EIGHT, NINE, TEN, JACK, QUEEN, KING, ACE;

}

```

Card.java | Deck.java | Player.java | App.java | Suit.java | Value.java

```java

public enum Suit {
    DIAMONDS, CLUBS, HEARTS, SPADES;

}

```

Card.java | Deck.java | Player.java | App.java | Suit.java | Value.java

```java
import java.util.ArrayList;

public class Deck {

    // fields
    private ArrayList<Card> cards;


    // constructor
    public Deck() {
        // creates an new, ordered deck upon construction
        cards = new ArrayList<Card>();

        for (Suit suit : Suit.values() ) {
            for (Value value : Value.values() ) {
                cards.add(new Card(value, suit) );
            }
        } // end of loop
    }

```

Card.java | *Deck.java | Player.java | App.java | Suit.java | Value.java

```java

    // methods
    public void describe() {
        System.out.println("====PLAYING DECK====");

        int i = 0;
        for (Card card : cards) {
            i++;
            System.out.print(i + "-");
            card.describe();
        }
    } // *note* this method is not used in the application.
      //        it is only used to print a specific output for assignment submission.


    public void shuffle() {
        // shuffles order of cards in deck

        Card tempCard;
        Random r = new Random();

        for (int i = 0; i < cards.size(); i++) {

            // generate random index
            int randIndex = r.nextInt(51);

            // swaps card at index i with card at randIndex
            tempCard = cards.get(i);
            cards.set(i, cards.get(randIndex) );
            cards.set(randIndex, tempCard);

        } // end of loop
    }


    public Card draw() {
        // removes and returns the top card of the Cards field
        Card drawnCard = cards.get(0);
        cards.remove(0);

        return drawnCard;
    }
}

```

Card.java | Deck.java | Player.java | App.java | Suit.java | Value.java

```java
import java.util.ArrayList;

public class Player extends Deck{

    // fields
    private ArrayList<Card> hand;
    private int score;
    private String name;


    // constructor
    public Player(String name) {
        this.name = name;
        score = 0;
        hand = new ArrayList<Card>();
    }


    @Override
    public void describe() {
        // prints info about the player and calls the describe method for each card in the ha
        System.out.println("=======PLAYER INFO=======");
        System.out.println("Player name: " + name);
        System.out.println("Score: " + score);
        System.out.println("Hand: ");
        try {
            // includes a counter easily count the amount of cards
            int i = 1;
            for (Card card : hand) {
                System.out.print(i + "-");
                card.describe();
                i++;
            } // end of loop

        } catch (NullPointerException e) {
            System.out.println("-EMPTY-");

        } catch (Exception e) {
            System.out.println("Could not retrieve hand: " + e.toString() );
        }
    }
```

Card.java | Deck.java | *Player.java | App.java | Suit.java | Value.java

```java
    }


    public Card flip() {
        // removes and returns the top card of the hand
        Card flippedCard = hand.get(0);
        hand.remove(0);

        return flippedCard;
    }


    public void draw(Deck deck) {
        // takes a Deck as an argument and calls the draw method on the deck,
        // adding the returned Card to the hand field
        hand.add(deck.draw() );
    }


    public void incrementScore() {
        // adds 1 to the player's score
        score++;
    }


    // getters/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;
    }
}
```

Card.java | Deck.java | Player.java | App.java | Suit.java | Value.java

```java

public class App {
    /*
     * Promineo BESD: Week 6 java project
     * automated WAR! card game
     *
     */

    public static void main(String[] args) {

        // welcome message
        System.out.println("Welcome to (automated) WAR!\n");


        // instantiates two players
        Player p1 = new Player("One");
        Player p2 = new Player("Two");
        System.out.println("Hello players: " + p1.getName() + ", " + p2.getName() + "\n");


        // instantiates new, ordered deck
        System.out.println("Generating deck...");
        Deck playingDeck = new Deck();


        // shuffles the deck
        System.out.println("Shuffling deck...\n");
        playingDeck.shuffle();


        // each player takes turns taking cards from the top of the playing deck
        System.out.println("Players, take turns drawing cards from the deck...\n");

        for (int i = 0; i < 52; i++) {
            if (i % 2 == 0) {
                p1.draw(playingDeck);
            }
            if (i % 2 != 0) {
                p2.draw(playingDeck);
            }
        }

```

Card.java | Deck.java | Player.java | App.java | Suit.java | Value.java

```java

        System.out.println("Game starting!");

        // game loop - iterates 26 times
        for (int i = 1; i <= 26; i++) {

            System.out.println("====ROUND " + i + "========================");
            // player1 flip
            System.out.print(p1.getName() + " flipped over: ");
            Card p1card = p1.flip();
            p1card.describe();

            // player2 flip
            System.out.print(p2.getName() + " flipped over: ");
            Card p2card = p2.flip();
            p2card.describe();

            // decides winner of each round
            if (judge(p1card, p2card) == 1) {
                System.out.println(p1.getName() + " wins the round.\n");
                p1.incrementScore();

            } else if (judge(p1card, p2card) == 2) {
                System.out.println(p2.getName() + " wins the round.\n");
                p2.incrementScore();

            } else {
                System.out.println("Draw - no point awarded.\n");
            }
        } // end of loop

```

Card.java | Deck.java | Player.java | App.java | Suit.java | Value.java

```java

        // game over message
        System.out.println("====GAME OVER=======================");
        System.out.println(p1.getName() + "'s final score: " + p1.getScore() );
        System.out.println(p2.getName() + "'s final score: " + p2.getScore() );


        // prints game winner
        if (p1.getScore() > p2.getScore() ) {
            System.out.println("\nRESULTS: " + p1.getName() + " wins!");

        } else if (p1.getScore() < p2.getScore() ) {
            System.out.println("\nRESULTS: " + p2.getName() + " wins!");

        } else {
            System.out.println("\nRESULTS: DRAW!");
        }

        // end message
        System.out.println("Thank you for playing.");
    }

```

Card.java | Deck.java | Player.java | *App.java | Suit.java | Value.java

```java


    // methods
    public static int judge(Card p1card, Card p2card) {
        // returns 1 if p1 wins, returns 2 if p2 wins, returns 3 if draw

        if (getValue(p1card) > getValue(p2card) ) {
            return 1; // p1 win

        } else if (getValue(p1card) < getValue(p2card) ) {
            return 2; // p2 win

        } else {
            return 3; // draw
        }
    }


    public static int getValue(Card card) {
        // returns a card's corresponding value

        switch (card.getValue() ) {
            case TWO: return 2;
            case THREE: return 3;
            case FOUR: return 4;
            case FIVE: return 5;
            case SIX: return 6;
            case SEVEN: return 7;
            case EIGHT: return 8;
            case NINE: return 9;
            case TEN: return 10;
            case JACK: return 11;
            case QUEEN: return 12;
            case KING: return 13;
            case ACE: return 14;
            default: return 0;
        }
    }

}


```

Screenshots of Running Application:

```
Problems  @ Javadoc  Declaration  Console
<terminated> App (2) [Java Application] C:\Program Files\Java\
Generating deck...
====PLAYING DECK====
1-TWO of DIAMONDS
2-THREE of DIAMONDS
3-FOUR of DIAMONDS
4-FIVE of DIAMONDS
5-SIX of DIAMONDS
6-SEVEN of DIAMONDS
7-EIGHT of DIAMONDS
8-NINE of DIAMONDS
9-TEN of DIAMONDS
10-JACK of DIAMONDS
11-QUEEN of DIAMONDS
12-KING of DIAMONDS
13-ACE of DIAMONDS
14-TWO of CLUBS
15-THREE of CLUBS
16-FOUR of CLUBS
17-FIVE of CLUBS
18-SIX of CLUBS
19-SEVEN of CLUBS
20-EIGHT of CLUBS
21-NINE of CLUBS
22-TEN of CLUBS
23-JACK of CLUBS
24-QUEEN of CLUBS
25-KING of CLUBS
26-ACE of CLUBS
27-TWO of HEARTS
28-THREE of HEARTS
29-FOUR of HEARTS
30-FIVE of HEARTS
31-SIX of HEARTS
32-SEVEN of HEARTS
33-EIGHT of HEARTS
34-NINE of HEARTS
35-TEN of HEARTS
36-JACK of HEARTS
37-QUEEN of HEARTS
38-KING of HEARTS
39-ACE of HEARTS
40-TWO of SPADES
41-THREE of SPADES
42-FOUR of SPADES
43-FIVE of SPADES
44-SIX of SPADES
45-SEVEN of SPADES
46-EIGHT of SPADES
47-NINE of SPADES
48-TEN of SPADES
49-JACK of SPADES
50-QUEEN of SPADES
51-KING of SPADES
52-ACE of SPADES
```

Problems Javadoc Declaration Console

<terminated> App (2) [Java Application] C:\Program Files\Java

```
Shuffling deck...

====PLAYING DECK====
1-TWO of SPADES
2-SIX of DIAMONDS
3-ACE of DIAMONDS
4-FOUR of HEARTS
5-TEN of CLUBS
6-SEVEN of CLUBS
7-FOUR of SPADES
8-ACE of SPADES
9-KING of CLUBS
10-FIVE of DIAMONDS
11-QUEEN of HEARTS
12-QUEEN of DIAMONDS
13-JACK of SPADES
14-KING of DIAMONDS
15-EIGHT of DIAMONDS
16-FIVE of SPADES
17-NINE of CLUBS
18-KING of SPADES
19-ACE of CLUBS
20-NINE of SPADES
21-EIGHT of SPADES
22-FIVE of CLUBS
23-TEN of HEARTS
24-TWO of HEARTS
25-FOUR of CLUBS
26-EIGHT of CLUBS
27-SIX of SPADES
28-KING of HEARTS
29-THREE of CLUBS
30-THREE of HEARTS
31-SEVEN of SPADES
32-ACE of HEARTS
33-QUEEN of SPADES
34-TEN of DIAMONDS
35-FOUR of DIAMONDS
36-NINE of HEARTS
37-SIX of HEARTS
38-THREE of SPADES
39-SEVEN of DIAMONDS
40-EIGHT of HEARTS
41-JACK of CLUBS
42-NINE of DIAMONDS
43-THREE of DIAMONDS
44-TWO of DIAMONDS
45-TWO of CLUBS
46-QUEEN of CLUBS
47-JACK of HEARTS
48-SIX of CLUBS
49-TEN of SPADES
50-FIVE of HEARTS
51-JACK of DIAMONDS
52-SEVEN of HEARTS
```

Problems Javadoc Declaration Console

<terminated> App (2) [Java Application] C:\Program Files\Java\jdk-11.0.15\bin\javaw.e

```
Players, take turns drawing cards from the deck...

=======PLAYER INFO=======
Player name: One
Score: 0
Hand:
1-TWO of SPADES
2-ACE of DIAMONDS
3-TEN of CLUBS
4-FOUR of SPADES
5-KING of CLUBS
6-QUEEN of HEARTS
7-JACK of SPADES
8-EIGHT of DIAMONDS
9-NINE of CLUBS
10-ACE of CLUBS
11-EIGHT of SPADES
12-TEN of HEARTS
13-FOUR of CLUBS
14-SIX of SPADES
15-THREE of CLUBS
16-SEVEN of SPADES
17-QUEEN of SPADES
18-FOUR of DIAMONDS
19-SIX of HEARTS
20-SEVEN of DIAMONDS
21-JACK of CLUBS
22-THREE of DIAMONDS
23-TWO of CLUBS
24-JACK of HEARTS
25-TEN of SPADES
26-JACK of DIAMONDS
=======PLAYER INFO=======
Player name: Two
Score: 0
Hand:
1-SIX of DIAMONDS
2-FOUR of HEARTS
3-SEVEN of CLUBS
4-ACE of SPADES
5-FIVE of DIAMONDS
6-QUEEN of DIAMONDS
7-KING of DIAMONDS
8-FIVE of SPADES
9-KING of SPADES
10-NINE of SPADES
11-FIVE of CLUBS
12-TWO of HEARTS
13-EIGHT of CLUBS
14-KING of HEARTS
15-THREE of HEARTS
16-ACE of HEARTS
17-TEN of DIAMONDS
18-NINE of HEARTS
19-THREE of SPADES
20-EIGHT of HEARTS
21-NINE of DIAMONDS
22-TWO of DIAMONDS
23-QUEEN of CLUBS
24-SIX of CLUBS
25-FIVE of HEARTS
26-SEVEN of HEARTS
```

Problems Javadoc Declaration Console

<terminated> App (2) [Java Application] C:\Program Files\Java\

```
Game starting!
====ROUND 1=========================
One flipped over: TWO of SPADES
Two flipped over: SIX of DIAMONDS
Two wins the round.

====ROUND 2=========================
One flipped over: ACE of DIAMONDS
Two flipped over: FOUR of HEARTS
One wins the round.

====ROUND 3=========================
One flipped over: TEN of CLUBS
Two flipped over: SEVEN of CLUBS
One wins the round.

====ROUND 4=========================
One flipped over: FOUR of SPADES
Two flipped over: ACE of SPADES
Two wins the round.

====ROUND 5=========================
One flipped over: KING of CLUBS
Two flipped over: FIVE of DIAMONDS
One wins the round.

====ROUND 6=========================
One flipped over: QUEEN of HEARTS
Two flipped over: QUEEN of DIAMONDS
Draw - no point awarded.

====ROUND 7=========================
One flipped over: JACK of SPADES
Two flipped over: KING of DIAMONDS
Two wins the round.

====ROUND 8=========================
One flipped over: EIGHT of DIAMONDS
Two flipped over: FIVE of SPADES
One wins the round.

====ROUND 9=========================
One flipped over: NINE of CLUBS
Two flipped over: KING of SPADES
```

```
Problems  Javadoc  Declarati...  Console
<terminated> App (2) [Java Application] C:\Program Files\Java

====ROUND 21=========================
One flipped over: QUEEN of DIAMONDS
Two flipped over: THREE of DIAMONDS
One wins the round.

====ROUND 22=========================
One flipped over: TWO of CLUBS
Two flipped over: NINE of HEARTS
Two wins the round.

====ROUND 23=========================
One flipped over: JACK of SPADES
Two flipped over: ACE of DIAMONDS
Two wins the round.

====ROUND 24=========================
One flipped over: JACK of CLUBS
Two flipped over: SIX of SPADES
One wins the round.

====ROUND 25=========================
One flipped over: FIVE of CLUBS
Two flipped over: SIX of CLUBS
Two wins the round.

====ROUND 26=========================
One flipped over: KING of CLUBS
Two flipped over: SEVEN of SPADES
One wins the round.

====GAME OVER========================
One's final score: 13
Two's final score: 12

RESULTS: One wins!
Thank you for playing.
```

URL to GitHub Repository:

https://github.com/johnnylee-55/week6_Final_Project