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
         // fields
         private Suit suit;
private String name;
         public Card(Value value, Suit suit) {
              this.suit = suit;
name = value + " of " + suit;
         public void describe() {
    System.out.println(name);
 19●
         // getters/setters
 25●
         public Value getValue() {
    return value;
28●
         public void setValue(Value value) {
 310
 34●
         public void setSuit(Suit suit) {
Card.java
             Deck.java
                          🗾 Player.java 🗾 App.java
                                                     ☑ Suit.java
☑ Value.java ×
 1
         TWO, THREE, FOUR, FIVE, SIX, SEVEN, EIGHT, NINE, TEN, JACK, QUEEN, KING, ACE;

☑ Card.java

             Deck.java

☑ App.java
☑ Suit.java × ☑ Value.java

                          🚺 Player.java
         DIAMONDS, CLUBS, HEARTS, SPADES;
```

```
🚺 Card.java 🚺 *Deck.java 🗶 🚺 Player.java
                                   App.java
                                              🗾 Suit.java
                                                         🚺 Value.java
240
          System.out.println("====PLAYING DECK====");
            for (Card card : cards) {
                System.out.print(i + "-");
                card.describe();
                    it is only used to print a specific output for assignment submission.
       public void shuffle() {
37●
            // shuffles order of cards in deck
            Card tempCard;
            Random r = new Random();
            for (int i = 0; i < cards.size(); i++) {</pre>
                // 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
57●
            Card drawnCard = cards.get(0);
            cards.remove(0);
            return drawnCard;
```

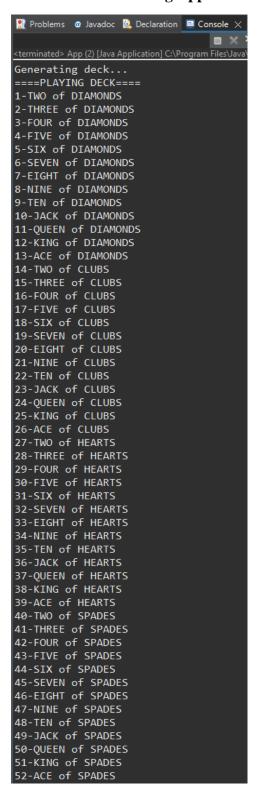
```
Card.java
              Deck.java
                             🗾 Player.java 💹 App.java 🗶 🗾 Suit.java
                                                                          🗾 Value.java
             * <u>Promineo</u> BESD: Week 6 java project
* automated WAR! card game
                 // welcome message
                 System.out.println("Welcome to (automated) WAR!\n");
                // instantiates two players
Player p1 = new Player("One");
Player p2 = new Player("Two");
                // instantiates new, ordered deck
System.out.println("Generating deck...");
Deck playingDeck = new 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");
                      if (i % 2 == 0) {
   p1.draw(playingDeck);
                      }
if (i % 2 != 0) {
                           p2.draw(playingDeck);
```

```
| Card_java | Deckjava | Deckjava
```

```
☑ Card.java

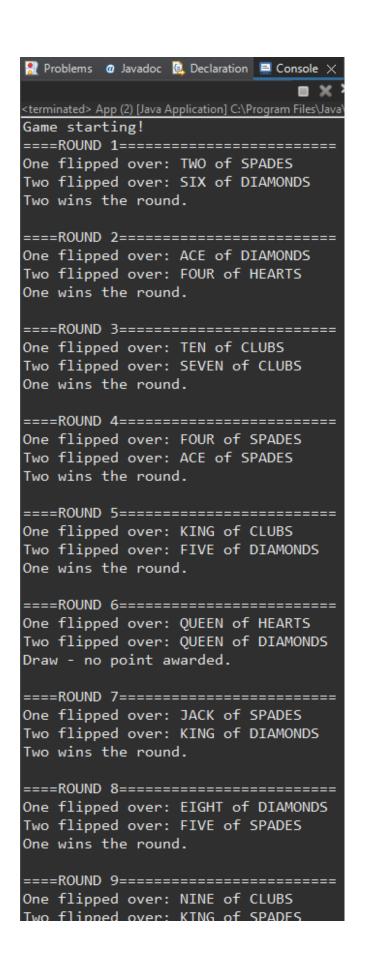
            Deck.java
                          ☑ Player.java
☑ *App.java × ☑ Suit.java
                                                                   🚺 Value.java
               if (getValue(p1card) > getValue(p2card) ) {
               } else if (getValue(p1card) < getValue(p2card) ) {</pre>
                    return 2; // p2 win
                    return 3; // draw
               // 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;
```

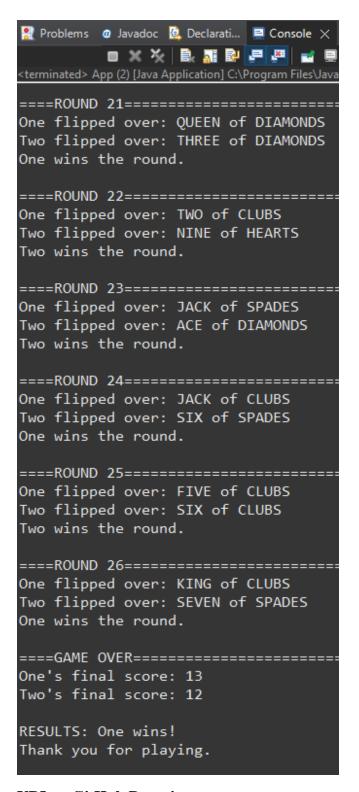
Screenshots of Running Application:











URL to GitHub Repository:

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