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)

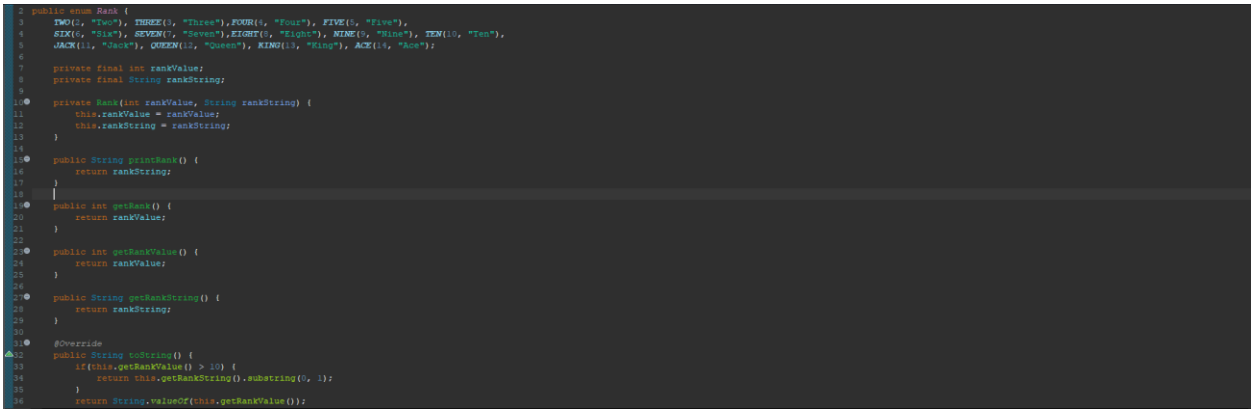
2. Create a class called App with a main method.
3. Instantiate a Deck and two Players, call the shuffle method on the deck.
4. Using a traditional for loop, iterate 52 times calling the Draw method on the other player each iteration using the Deck you instantiated.
5. Using a traditional for loop, iterate 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:

Suits Enum:

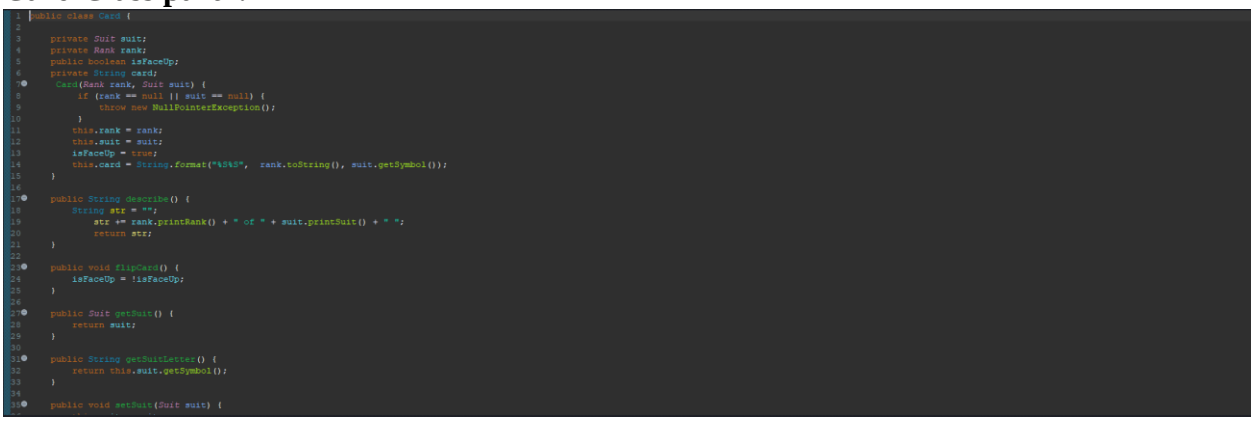
```
public enum Suit {
    HEARTS("Hearts", "H"),
    SPADES("Spades", "S"),
    DIAMONDS("Diamonds", "D"),
    CLUBS("Clubs", "C");

    private final String suitName;
    private String symbol;

    Suit(String suitName, String symbol) {
        this.suitName = suitName;
        this.symbol = symbol;
    }

    public String getSymbol() {
        return symbol;
    }

    public void setSymbol(String symbol) {
        this.symbol = symbol;
    }

    public String getSuitName() {
        return suitName;
    }

    public String printSuit() {
        return suitName;
    }
}
```

Rank Enum:



```
public enum Rank {
    TWO(2, "Two"), THREE(3, "Three"),FOUR(4, "Four"), FIVE(5, "Five"),
    SIX(6, "Six"), SEVEN(7, "Seven"),EIGHT(8, "Eight"), NINE(9, "Nine"), TEN(10, "Ten"),
    JACK(11, "Jack"), QUEEN(12, "Queen"), KING(13, "King"), ACE(14, "Ace");

    private final int rankValue;
    private final String rankString;

    private Rank(int rankValue, String rankString) {
        this.rankValue = rankValue;
        this.rankString = rankString;
    }

    public String printRank() {
        return rankString;
    }

    public int getRank() {
        return rankValue;
    }

    public int getRankValue() {
        return rankValue;
    }

    public String getRankString() {
        return rankString;
    }

    @Override
    public String toString() {
        if(this.getRankValue() > 10) {
            return this.getRankString().substring(0, 1);
        }
        return String.valueOf(this.getRankValue());
```

Card Class part1:



```
public class Card {

    private Suit suit;
    private Rank rank;
    public boolean isFaceUp;
    private String card;
     Card(Rank rank, Suit suit) {
         if (rank == null || suit == null) {
             throw new NullPointerException();
         }
        this.rank = rank;
        this.suit = suit;
        isFaceUp = true;
        this.card = String.format("%5%5",  rank.toString(), suit.getSymbol());
    }

    public String describe() {
        String str = "";
            str += rank.printRank() + " of " + suit.printSuit() + " ";
            return str;
    }

    public void flipCard() {
        isFaceUp = !isFaceUp;
    }

    public Suit getSuit() {
        return suit;
    }

    public String getSuitLetter() {
        return this.suit.getSymbol();
    }

    public void setSuit(Suit suit) {
```

Card Class part2:

```
    public Rank getRank() {
        return rank;
    }

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

    public String getRankLetter() {
        return this.rank.toString();
    }

    public Integer getCardPoints() {
        return this.rank.getRankValue();
    }

    public String getCard() {
        return card;
    }

    public void setCard(String card) {
        this.card = card;
    }

    public int compareTo(Card other) {
        return this.getCardPoints().compareTo(other.getCardPoints());
        }

    public boolean isSameSuit(Card other) {
        return this.suit.equals(other.suit);
        }

    public boolean isSameRank(Card other) {
        return this.rank.equals(other.rank);
```

Card Class part3:

```
@Override
    public boolean equals(Object obj) {
    if(obj == null) {
        return false;
    }
    if (!Card.class.isAssignableFrom(getClass())) {
        return false;
    }
    final Card other = (Card) obj;
    return this.suit == other.suit && this.rank == other.rank;
    }

@Override
    public String toString() {
    return this.rank.getRankString() + " of " + this.suit.getSuitName();
}
}
```

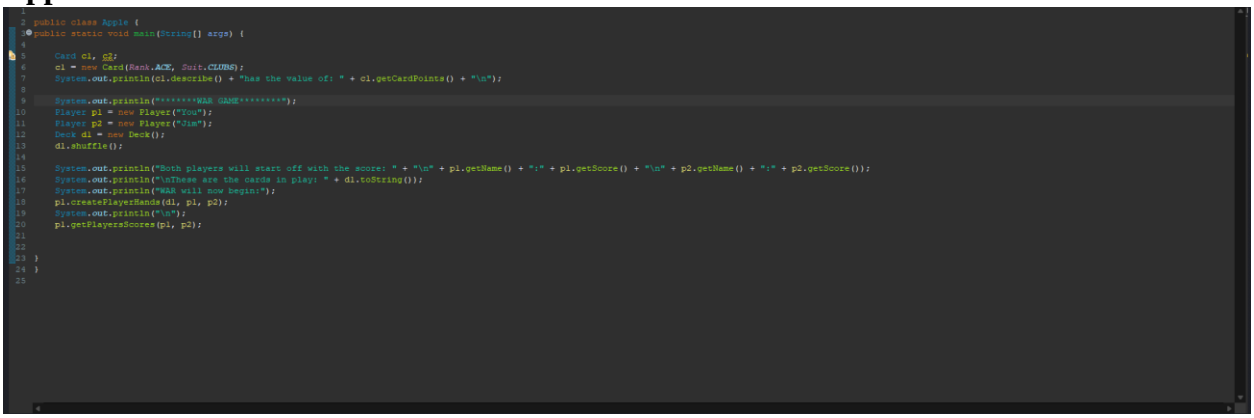
Deck Class part1:

```
import java.util.Collections;
import java.util.List;
import java.util.ArrayList;

public class Deck {

    List<Card> cards = new ArrayList<Card>();

    Deck() {
        createDeck();
    }

    private void createDeck() {
    for (Suit s : Suit.values()) {
        for (Rank r : Rank.values()) {
            cards.add(new Card(r, s));
        }
    }
}

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

    public List<Card> getCards() {
        return this.cards;
    }

    public int getCardsCount() {
        return this.cards.size();
    }

    public Card dealCard() {
        return this.cards.remove(this.cards.size() - 1);
    }
```

Deck Class part2:

```
    public void addCard(Card c) {
        this.cards.add(c, c);
    }

    @Override
    public String toString() {
        return this.cards.toString();
    }
    }
```

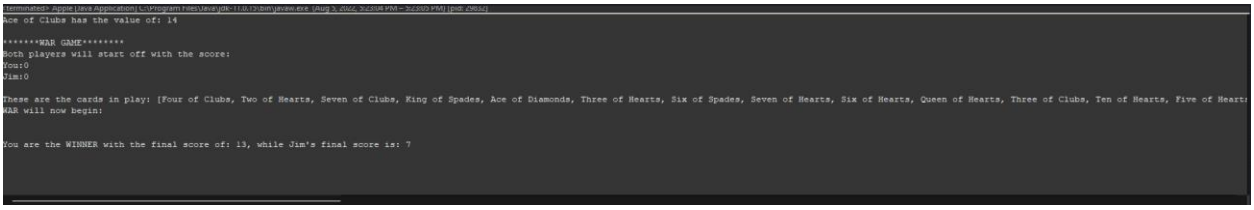
Application:



```
public class Apple {
public static void main(String[] args) {

    Card c1, c2;
    c1 = new Card(Rank.ACE, Suit.CLUBS);
    System.out.println(c1.describe() + "has the value of: " + c1.getCardPoints() + "\n");

    System.out.println("*******WAR GAME********");
    Player p1 = new Player("You");
    Player p2 = new Player("Jim");
    Deck d1 = new Deck();
    d1.shuffle();

    System.out.println("Both players will start off with the score: " + "\n" + p1.getName() + ":" + p1.getScore() + "\n" + p2.getName() + ":" + p2.getScore());
    System.out.println("\nThese are the cards in play: " + d1.toString());
    System.out.println("WAR will now begin:");
    p1.createPlayerHands(d1, p1, p2);
    System.out.println("\n");
    p1.getPlayersScores(p1, p2);


}
}
```

Screenshots of Running Application:

AceofClubs+Winner:



```
<terminated> Apple [Java Application] C:\Program Files\Java\jdk-11.0.13\bin\javaw.exe (Aug 3, 2022, 3:23:04 PM – 3:23:05 PM) [pid: 23824]
Ace of Clubs has the value of: 14

*******WAR GAME********
Both players will start off with the score:
You:0
Jim:0

These are the cards in play: [Four of Clubs, Two of Hearts, Seven of Clubs, King of Spades, Ace of Diamonds, Three of Hearts, Six of Spades, Seven of Hearts, Six of Hearts, Queen of Hearts, Three of Clubs, Ten of Hearts, Five of Hearts
WAR will now begin:


You are the WINNER with the final score of: 13, while Jim's final score is: 7
```

KingofClubs+Loser:

```
<terminated> Apple [Java Application] C:\Program Files\Java\jdk-11.0.15\bin\javaw.exe (Aug 5, 2022, 5:28:24 PM – 5:28:24 PM) [pid: 24548]
King of Clubs has the value of: 13

*******WAR GAME********
Both players will start off with the score:
You:0
Jim:0

These are the cards in play: [Ten of Diamonds, Seven of Spades, Eight of Spades, Ace of Hearts, Two of Diamonds, Queen of Hearts, Nine of Diamonds, Six of Spades, King of Clubs, Two of Spades, Eight of Clubs, Queen of Clubs, Seven of
WAR will now begin:


You are the LOSER with the final score of: 10, while Jim's final score is: 14
```

FiveofClubs+Draw:

URL to GitHub Repository:

RPador/PromineoTech (github.com)