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

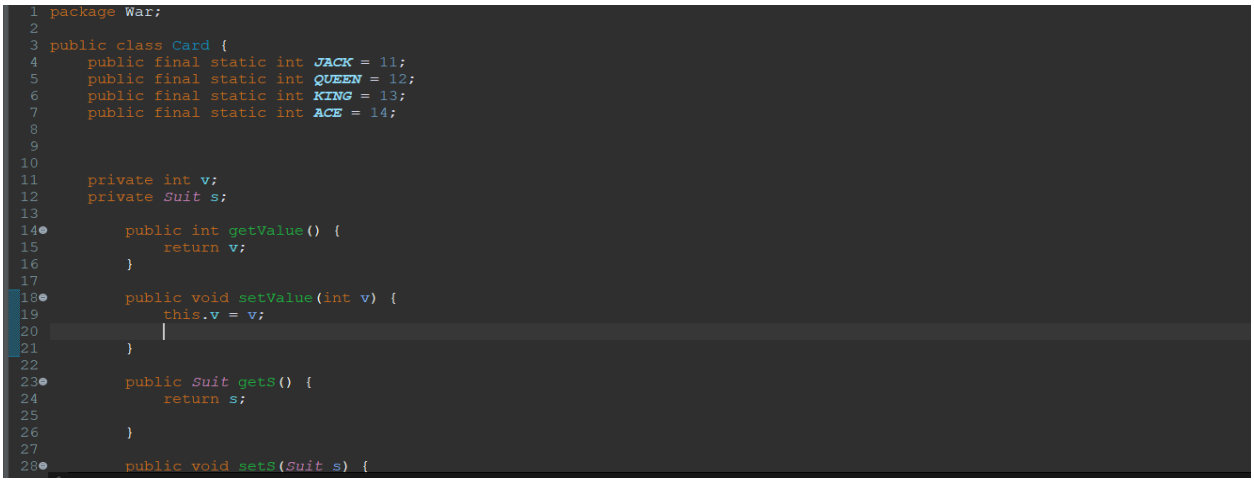
1. **hand** (List of Card)
2. **score** (set to 0 in the constructor)
3. **name**

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:



```
1  package War;
2
3  public class Card {
4      public final static int JACK = 11;
5      public final static int QUEEN = 12;
6      public final static int KING = 13;
7      public final static int ACE = 14;
8
9
10
11     private int v;
12     private Suit s;
13
14         public int getValue() {
15             return v;
16         }
17
18         public void setValue(int v) {
19             this.v = v;
20
21         }
22
23         public Suit getS() {
24             return s;
25
26         }
27
28         public void setS(Suit s) {
```

```
29            this.s = s;
30
31        }
32
33    public String describe() {
34        StringBuilder output = new StringBuilder();
35
36        output.append(getS().getTag().toString());
37        switch (v) {
38        case JACK:
39            output.append("J" + "]");
40            break;
41        case QUEEN:
42            output.append("Q" + "]");
43            break;
44        case KING:
45            output.append("K" + "]");
46            break;
47        case ACE:
48            output.append("A" + "]");
49            break;
50        default:
51            output.append(v + "]");
52        }
53
54        return(output.toString());
55
56        }
```

```
1  package War;
2
3  import java.util.ArrayList;
4  import java.util.List;
5  import java.util.Collections;
6
7
8
9  public class Deck {
10
11     private List<Card> cards = new ArrayList<Card>();
12
13     public Deck() {
14
15         String[] value = new String[14];
16
17                         value[0] = null;
18                         value[1] = "Two";
19                         value[2] = "Three";
20                         value[3] = "Four";
21                         value[4] = "Five";
22                         value[5] = "Six";
23                         value[6] = "Seven";
24                         value[7] = "Eight";
25                         value[8] = "Nine";
26                         value[9] = "Ten";
27                         value[10] = "J";
28                         value[11] = "Q";
```

```
                        value[12] = "K";
                        value[13] = "A";

    for(int i = 1; i <= 13; i++) {
        Card c = new Card();
        c.setS(Suit.HEART);
        c.setValue(i);
        cards.add(c);
    }
    for(int i = 1; i<= 13; i++) {
        Card c = new Card();
        c.setS(Suit.CLUB);
        c.setValue(i);
        cards.add(c);

    }
    for (int i = 1; i <= 13; i++) {
        Card c = new Card();
        c.setS(Suit.DIAMOND);
        c.setValue(i);
        cards.add(c);

    }
    for(int i = 1; i <= 13; i++) {
        Card c = new Card();
        c.setS(Suit.SPADE);
        c.setValue(i);
        cards.add(c);
```

```
        cards.add(c);

    }


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

    public Card draw() {
        return cards.remove(0);
    }

}
```

```java
package War;

import java.util.ArrayList;
import java.util.List;


public class Player {

        private List<Card> hand = new ArrayList<>();
        private int score = 0;
        private String name;

        public Player() {}

        public Player(String name, List<Card> hand, int score) {
            this.name = name;
            this.hand = hand;
            this.score = 0;

        }

        public void describe(Card card) {
            System.out.println(this.name + " draws: " + card.describe());
        }

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

```java
        public void setHand(List<Card> hand) {
            this.hand = hand;
        }

        public int getScore() {
            return score;

        }

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

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

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

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

        public String getName() {
            return name;
```

```java
        }

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

```java
package War;

public enum Suit {
// just to make things fancy
    DIAMOND("["+"\u2666"), CLUB("["+"\u2663"),SPADE("["+"\u2660"),HEART("["+"\u2665");
// my mentor helped me with the suits
    private String tag;
    Suit(String tag){
        this.tag = tag;
    }

    public String getTag() {
        return(tag);
    }
}
```

```java
package War;

public class App {
    public static void main(String[] args) {
//3.
        Deck deck = new Deck();
        deck.shuffle();
        Player p1 = new Player();
        p1.setName("P1");
        Player p2 = new Player();
        p2.setName("P2");


//4.
        for(int i =0; i <52; i++) {
            if(i % 2 == 0) {
                p1.draw(deck);
            }
            else {
                p2.draw(deck);
            }
        }

//5.
        int turn = 1;
        for(int i = 0; i < 26; i++) {
            System.out.println("********************");
            System.out.println("\t[ Turn:   " + turn + " ]");
```

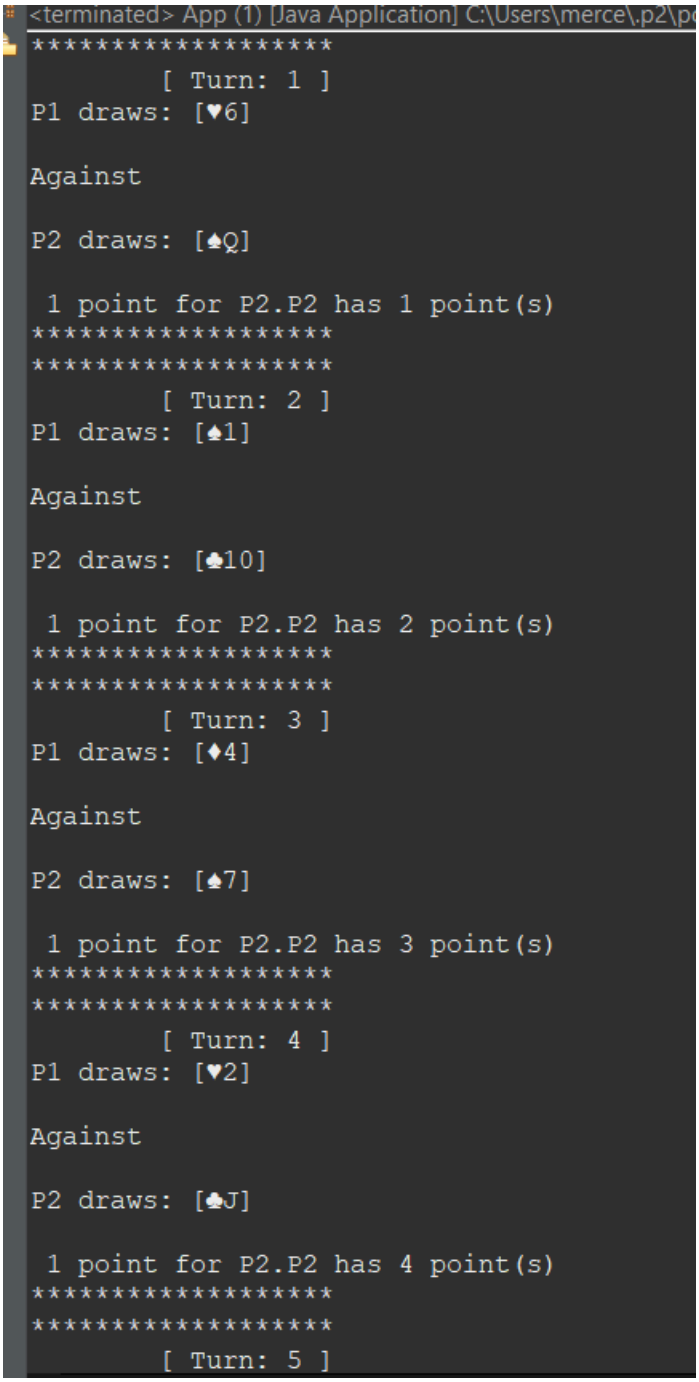
```java
        Card c1 = p1.flip();
        p1.describe(c1);
        System.out.println("\t ");
        System.out.println("Against");
        System.out.println("\t ");
        Card c2 = p2.flip();
        p2.describe(c2);
        System.out.println();
        if(c1.getValue() > c2.getValue()) {
            p1.incrementScore();
            System.out.println(" 1 point for " + p1.getName()+"." + p1.getName()+ " has "+ p1.getScore()+ " point(s)");
            System.out.println("********************");
        }else if(c1.getValue() < c2.getValue()) {
            p2.incrementScore();
            System.out.println(" 1 point for " + p2.getName()+"." + p2.getName()+ " has "+ p2.getScore()+ " point(s)");
            System.out.println("********************");
        }else {
            System.out.println("Tie game!");
        }
        turn += 1;
    }
//6 and 7.
        System.out.println("********************");
        System.out.println("\t [WINNER!!!]");
        System.out.println("********************");
    if(p1.getScore() > p2.getScore()) {
        System.out.println(p1.getName() + " with the final score of " + p1.getScore() + " and " + p2.getName() + " final score of " + p2.getScore()+"," + p1.getName() + " is the WINN
```

```java
    } else if(p1.getScore() < p2.getScore()) {
        System.out.println(p2.getName() + " with the final score of " + p2.getScore() + " and " + p1.getName() + " final score of " + p1.getScore()+"," + p2.getName() + " is the WINN
    } else {
        System.out.println(p1.getName() + " and " + p2.getName() + " tied with the score of " + p1.getScore());
    }
    System.out.println("******************** GAME OVER ********************");

    }
```

```
) + " and " + p2.getName() + " final score of " + p2.getScore()+"," + p1.getName() + " is the WINNER!!");
) + " and " + p1.getName() + " final score of " + p1.getScore()+"," + p2.getName() + " is the WINNER!!");
core of " + p1.getScore());
```

Screenshots of Running Application:



```
        [ Turn: 5 ]
P1 draws: [♥4]

Against

P2 draws: [♠J]

 1 point for P2.P2 has 5 point(s)
********************
********************
        [ Turn: 6 ]
P1 draws: [♣9]

Against

P2 draws: [♠5]

 1 point for P1.P1 has 1 point(s)
********************
********************
        [ Turn: 7 ]
P1 draws: [♣Q]

Against

P2 draws: [♦8]

 1 point for P1.P1 has 2 point(s)
********************
********************
        [ Turn: 8 ]
P1 draws: [♠3]

Against

P2 draws: [♠K]

 1 point for P2.P2 has 6 point(s)
********************
********************
        [ Turn: 9 ]
P1 draws: [♥9]
```

```
        [ Turn: 9 ]
P1 draws: [♥9]

Against

P2 draws: [♥8]

 1 point for P1.P1 has 3 point(s)
********************
********************
        [ Turn: 10 ]
P1 draws: [♠8]

Against

P2 draws: [♠2]

 1 point for P1.P1 has 4 point(s)
********************
********************
        [ Turn: 11 ]
P1 draws: [♣K]

Against

P2 draws: [♠9]

 1 point for P1.P1 has 5 point(s)
********************
********************
        [ Turn: 12 ]
P1 draws: [♦10]

Against

P2 draws: [♥3]

 1 point for P1.P1 has 6 point(s)
********************
********************
        [ Turn: 13 ]
```

```
        [ Turn: 13 ]
P1 draws: [♣2]

Against

P2 draws: [♠6]

 1 point for P2.P2 has 7 point(s)
********************
********************
        [ Turn: 14 ]
P1 draws: [♦5]

Against

P2 draws: [♥7]

 1 point for P2.P2 has 8 point(s)
********************
********************
        [ Turn: 15 ]
P1 draws: [♦3]

Against

P2 draws: [♣6]

 1 point for P2.P2 has 9 point(s)
********************
********************
        [ Turn: 16 ]
P1 draws: [♥Q]

Against

P2 draws: [♦9]

 1 point for P1.P1 has 7 point(s)
********************
********************
        [ Turn: 17 ]
P1 draws: [♠8]
```

```
        [ Turn: 17 ]
P1 draws: [♠8]

Against

P2 draws: [♣3]

 1 point for P1.P1 has 8 point(s)
********************
********************
        [ Turn: 18 ]
P1 draws: [♣4]

Against

P2 draws: [♣5]

 1 point for P2.P2 has 10 point(s)
********************
********************
        [ Turn: 19 ]
P1 draws: [♥10]

Against

P2 draws: [♦J]

 1 point for P2.P2 has 11 point(s)
********************
********************
        [ Turn: 20 ]
P1 draws: [♥5]

Against

P2 draws: [♣7]

 1 point for P2.P2 has 12 point(s)
********************
********************
        [ Turn: 21 ]
P1 draws: [♦1]
```

```
        [ Turn: 21 ]
P1 draws: [♦1]

Against

P2 draws: [♥K]

 1 point for P2.P2 has 13 point(s)
********************
********************
        [ Turn: 22 ]
P1 draws: [♦2]

Against

P2 draws: [♠10]

 1 point for P2.P2 has 14 point(s)
********************
********************
        [ Turn: 23 ]
P1 draws: [♣1]

Against

P2 draws: [♦6]

 1 point for P2.P2 has 15 point(s)
********************
********************
        [ Turn: 24 ]
P1 draws: [♦Q]

Against

P2 draws: [♦K]

 1 point for P2.P2 has 16 point(s)
********************
********************
        [ Turn: 25 ]
P1 draws: [♦7]
```

```
        [ Turn: 25 ]
P1 draws: [♦7]

Against

P2 draws: [♥1]

 1 point for P1.P1 has 9 point(s)
********************
********************
        [ Turn: 26 ]
P1 draws: [♠4]

Against

P2 draws: [♥J]

 1 point for P2.P2 has 17 point(s)
********************
********************
         {WINNER!!!}
********************
P2 with the final score of 17 and P1 final score of 9,P2 is the WINNER!!
******************** GAME OVER ********************
```

URL to GitHub Repository: