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 FinalProject;
2
3  public class Card {
4
5      private int value;
6      private String name;
7
8      public int getValue() {
9          return value;
10     }
11     public void setValue(int value) {
12         this.value = value;
13     }
14     public String getName() {
15         return name;
16     }
17     public void setName(String name) {
18         this.name = name;
19     }
20
21     public void describe() {
22         System.out.println(name);
23     }
24
25
26 }
27
```

```java
package FinalProject;

import java.util.ArrayList;

public class Deck {

    private List <Card> deck = new ArrayList<Card>();

    public Deck() {
        for(int i = 0; i < 52; i++) {
            Card current = new Card();
            current.setValue((i % 13) + 2);
            StringBuilder currentName = new StringBuilder("");
            currentName.append(valueToName(current.getValue()));
            currentName.append(suitAssigner((i / 13) % 4));
            current.setName(currentName.toString());
            deck.add(current);
        }
    }

    public void Shuffle() {
        Collections.shuffle(deck);
        Collections.shuffle(deck);
        Collections.shuffle(deck);
    }

    public Card Draw() {
        Card spaceHold = deck.get(0);
        deck.remove(0);
        return spaceHold;
    }

    private String valueToName(int value) {
        if (value == 2) {
            return "Two of ";
        } else if (value == 3){
            return "Three of ";
        } else if (value == 4){
            return "Four of ";
        } else if (value == 5){
            return "Five of ";
        } else if (value == 6){
            return "Six of ";
        } else if (value == 7){
            return "Seven of ";
        } else if (value == 8){
            return "Eight of ";
        } else if (value == 9){
            return "Nine of ";
        } else if (value == 10){
            return "Ten of ";
        } else if (value == 11){
            return "Jack of ";
        } else if (value == 12){
            return "Queen of ";
        } else if (value == 13){
            return "King of ";
        } else if (value == 14){
            return "Ace of ";
        } else {
            return "";
        }
    }

    private String suitAssigner(int suitVal) {
        if(suitVal == 0) {
            return "Hearts";
        } else if (suitVal == 1) {
            return "Diamonds";
        } else if (suitVal == 2) {
            return "Clubs";
        } else if (suitVal == 3) {
            return "Spades";
        } else {
            return "";
        }
    }

    public void describeDeck() {
        for(int i = 0; i < deck.size(); i++) {
            deck.get(i).describe();
        }
    }


}
```

```java
package FinalProject;

import java.util.ArrayList;

public class Player {

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


    public Player(String name) {
        this.name = name;
        setScore(0);
    }


    public void Draw(Card drawnCard) {
        hand.add(drawnCard);
    }

    public void DrawMult(List<Card> cardList) {
        hand.addAll(cardList);
    }

    public Card Flip() {
            Card topCard = hand.get(0);
            hand.remove(0);
            return topCard;

    }

```

```java

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

    public void describe() {
        System.out.println(name + "'s Hand:");
        for(int i = 0; i < hand.size(); i++) {
            hand.get(i).describe();
        }
    }

    public String getName() {
        return name;
    }

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

    public int getScore() {
        return score;
    }

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

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


}
```

```java
package FinalProject;

public class App {

    public static void main(String[] args) {

        Deck appDeck = new Deck();
        Player player1 = new Player("Billy Bob");
        Player player2 = new Player("Cletus Jones");
        appDeck.Shuffle();

        for(int i = 0; i < 52; i++) {
            if(i % 2 == 0) {
                player1.Draw(appDeck.Draw());
            } else {
                player2.Draw(appDeck.Draw());
            }
        }

        System.out.println("Game of War between " + player1.getName() + " and " + player2.getName()+ ":");

        for(int i = 0; i < 26; i++) {
            Card player1Card = player1.Flip();
            Card player2Card = player2.Flip();
            if (player1Card.getValue() > player2Card.getValue()) {
                player1.incrementScore();
            } else if (player2Card.getValue() > player1Card.getValue()) {
                player2.incrementScore();
            }
        }

        System.out.println(player1.getName() + "'s Score: " + player1.getScore());
        System.out.println(player2.getName() + "'s Score: " + player2.getScore());
        if (player1.getScore() > player2.getScore()) {
            System.out.println("Winner: " + player1.getName());
        } else if (player2.getScore() > player1.getScore()) {
            System.out.println("Winner: " + player2.getName());;
        } else {
            System.out.println("Game ends in a Draw");
        }


    }

}

```

Screenshots of Running Application:

```
<terminated> App (1) [Java Application] C:\Program Files\Java\jdk-11.0.15\bin\javaw.exe  (Aug 5, 2022, 1:28:05 PM – 1:28:05 PM) [pid: 35108]
Game of War between Billy Bob and Cletus Jones:
Billy Bob's Score: 13
Cletus Jones's Score: 11
Winner: Billy Bob
```

```
<terminated> App (1) [Java Application] C:\Program Files\Java\jdk-11.0.15\bin\javaw.exe  (Aug 5, 2022, 1:28:35 PM – 1:28:35 PM) [pid: 34920]
Game of War between Billy Bob and Cletus Jones:
Billy Bob's Score: 11
Cletus Jones's Score: 14
Winner: Cletus Jones
```

```
<terminated> App (1) [Java Application] C:\Program Files\Java\jdk-11.0.15\bin\javaw.exe  (Aug 5, 2022, 1:29:43 PM – 1:29:43 PM) [pid: 39120]
Game of War between Billy Bob and Cletus Jones:
Billy Bob's Score: 12
Cletus Jones's Score: 12
Game ends in a Draw
```

URL to GitHub Repository:

https://github.com/kopatsis/Week6FinalProject

Bonus:

I also added a more complicated version of the app Called WarTwoPointO.java which plays war using the actual card game's methodology with continuous rounds until one person runs out of cards:

```java
package FinalProject;

import java.util.ArrayList;

public class WarTwoPointO {

    public static Player player1 = new Player("Billy Bob");
    public static Player player2 = new Player("Cletus Jones");
    public static boolean isDraw = false;
    public static boolean gameOver = false;
    public static boolean p1wins = false;
    public static List<Card> tieList = new ArrayList<Card>();
    public static int matchCount = 0;

    public static void main(String[] args) {

        Deck warDeck = new Deck();
        warDeck.Shuffle();

        for(int i = 0; i < 52; i++) {
            if(i % 2 == 0) {
                player1.Draw(warDeck.Draw());
            } else {
                player2.Draw(warDeck.Draw());
            }
        }

        while(player1.getHand().size() > 0 && player2.getHand().size() > 0 && gameOver == false) {
            Card p1card = player1.Flip();
            Card p2card = player2.Flip();
            matchCount++;
            if(player1.getHand().size() > 0 && player2.getHand().size() > 0) {
                if(p1card.getValue() == p2card.getValue()) {
                    if (player1.getHand().size() == 1 || player2.getHand().size() == 1) {
                        tieList.add(p2card);
                        tieList.add(p1card);
                        tieBreakerIf1(tieList);
                    } else {
                        tieList.add(p2card);
                        tieList.add(p1card);
                        tieBreakerIf2Plus(tieList);
                    }
                } else {
                    if(p1card.getValue() > p2card.getValue()) {
                        player1.Draw(p1card);
                        player1.Draw(p2card);
                    } else {
                        player2.Draw(p1card);
                        player2.Draw(p2card);
                    }
                }
            } else {
                if(p1card.getValue() == p2card.getValue()) {
                    gameOver = true;
                    isDraw = true;
                } else {
                    if(player1.getHand().size() == 0) {
                        if (p1card.getValue() > p2card.getValue()) {
                            player1.Draw(p1card);
                            player1.Draw(p2card);
                        } else {
                            gameOver = true;
                        }
                    } else {
                        if (p1card.getValue() > p2card.getValue()) {
                            gameOver = true;
                            p1wins = true;
                        } else {
                            player2.Draw(p1card);
                            player2.Draw(p2card);
                        } //Goes to p1card vs p2 card if statement
                    } //Goes to p1 hand size vs p2 hand size if statement
                } //Goes to p1 card value equals p2 card value if statement

            } //Goes to both hand sizes > 0 or not if statement
        } //Goes to full match's while loop

        if(isDraw == true) {
            System.out.println("The game ended in a draw after " + matchCount + " matches");
        } else if(p1wins == true) {
            System.out.println(player1.getName() + " wins the game after " + matchCount + " matches");
        } else {
            System.out.println(player2.getName() + " wins the game after " + matchCount + " matches");
        }

    }
```

```
    } //Goes to public static void main

    public static void tieBreakerIf2Plus(List<Card> tieList) {
        tieList.add(player1.Flip());
        tieList.add(player2.Flip());
        Card p1card = player1.Flip();
        Card p2card = player2.Flip();
        matchCount++;
        if(player1.getHand().size() > 0 && player2.getHand().size() > 0) {
            if(p1card.getValue() == p2card.getValue()) {
                if (player1.getHand().size() == 1 || player2.getHand().size() == 1) {
                    tieList.add(p2card);
                    tieList.add(p1card);
                    tieBreakerIf1(tieList);
                } else {
                    tieList.add(p2card);
                    tieList.add(p1card);
                    tieBreakerIf2Plus(tieList);
                }
            } else {
                if(p1card.getValue() > p2card.getValue()) {
                    player1.DrawMult(tieList);
                    tieList.clear();
                    player1.Draw(p1card);
                    player1.Draw(p2card);
                } else {
                    player2.DrawMult(tieList);
                    tieList.clear();
                    player2.Draw(p1card);
                    player2.Draw(p2card);
                }
            }
        } else {
            if(p1card.getValue() == p2card.getValue()) {
                gameOver = true;
                isDraw = true;
            } else {
                if(player1.getHand().size() == 0) {
                    if (p1card.getValue() > p2card.getValue()) {
                        player1.DrawMult(tieList);
                        tieList.clear();
                        player1.Draw(p1card);
                        player1.Draw(p2card);
                    } else {
                        gameOver = true;
                    }
```

```
                } else {
                    if (p1card.getValue() > p2card.getValue()) {
                        gameOver = true;
                        p1wins = true;
                    } else {
                        player2.DrawMult(tieList);
                        tieList.clear();
                        player2.Draw(p1card);
                        player2.Draw(p2card);
                    } //Goes to p1card vs p2 card if statement
                } //Goes to p1 hand size vs p2 hand size if statement
            } //Goes to p1 card value equals p2 card value if statement

        } //Goes to both hand sizes > 0 or not if statement
    } //Goes to tieBreakerIf2Plus method signature

    public static void tieBreakerIf1(List<Card> tieList) {
        Card p1card = player1.Flip();
        Card p2card = player2.Flip();
        matchCount++;
        if(p1card.getValue() == p2card.getValue()) {
            gameOver = true;
            isDraw = true;
        } else {
            if(player1.getHand().size() == 0) {
                if (p1card.getValue() > p2card.getValue()) {
                    player1.DrawMult(tieList);
                    tieList.clear();
                    player1.Draw(p1card);
                    player1.Draw(p2card);
                } else {
                    gameOver = true;
                }
            } else {
                if (p1card.getValue() > p2card.getValue()) {
                    gameOver = true;
                    p1wins = true;
                } else {
                    player2.DrawMult(tieList);
                    tieList.clear();
                    player2.Draw(p1card);
                    player2.Draw(p2card);
                } //Goes to card value comparisons for p2 hand size 0
            } //Goes to if p1 or p2 has the hand size of 0
        } //Goes to if it's a tie or not

    } //Goes to tieBreakerIf1 method signature

} //Goes to brackets encompassing entire class WarTwoPointO
```

```
<terminated> WarTwoPointO [Java Application] C:\Program Files\Java\jdk-11.0.15\bin\javaw.exe  (Aug 5, 2022, 1:33:55 PM – 1:33:55 PM) [pid: 25332]
Billy Bob wins the game after 974 matches
```

```
<terminated> WarTwoPointO [Java Application] C:\Program Files\Java\jdk-11.0.15\bin\javaw.exe  (Aug 5, 2022, 1:34:08 PM – 1:34:08 PM) [pid: 8436]
Cletus Jones wins the game after 4316 matches
```

```
<terminated> WarTwoPointO [Java Application] C:\Program Files\Java\jdk-11.0.15\bin\javaw.exe  (Aug 5, 2022, 1:34:31 PM – 1:34:31 PM) [pid: 30428]
The game ended in a draw after 6281 matches
```

I don’t recommend running this as it is intensely processor and memory heavy…