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:

App:

```
1  package war;
2
3  public class App {
4      public static void main(String[] args) {
5
6          Deck deck = new Deck();
7          Player player1 = new Player();
8          Player player2 = new Player();
9
10         player1.setName("Marvel");
11         player2.setName("Murphy");
12
13         deck.shuffle(deck);
14
15         draw(deck, player1, player2);
16
17         play(player1, player2);
18         player1.describe();
19         player2.describe();
20
21         printWinner(player1, player2);
22     }
23         public static void draw(Deck deck, Player player1, Player player2) {
24         for(int i = 0; i < 52; i++) {
25             if(i % 2 != 0) {
26                 player1.draw(deck);
27             }
28             else {
29                 player2.draw(deck);
30         }
31         }
32     }
33         public static void play(Player player1, Player player2) {
34         for(int i = 0; i < 26; i++) {
35             Card card1 = player1.flip();
36             Card card2 = player2.flip();
37             if(card1.getValue() > card2.getValue()) {
38                 player1.incrementScore();
39             }
40             else if (card1.getValue() < card2.getValue()) {
41                 player2.incrementScore();
42
43         }
44         }
45     }
46         public static void printWinner(Player player1, Player player2) {
47         if(player1.getScore() == player2.getScore()) {
48             System.out.println("There has been a draw");
49         }
50         if(player1.getScore() > player2.getScore()) {
51             System.out.println(player1.toString() + " has won");
52         }
```

```
        }
    }
        public static void printWinner(Player player1, Player player2) {
        if(player1.getScore() == player2.getScore()) {
            System.out.println("There has been a draw");
        }
        if(player1.getScore() > player2.getScore()) {
            System.out.println(player1.toString() + " has won");
        }
        if(player2.getScore() > player1.getScore()) {
            System.out.println(player2.toString() + " has won");
        }
    }
}
```

Card:

```
1  package war;
2
3  public class Card {
4       private int value;
5       private String name;
6
7       public Card(int value, String name) {
8               this.value = value;
9               this.name = name;
10          }
11      public void describe() {
12          System.out.println(value + "of" + name);
13 }
14      public int getValue() {
15              return value;
16          }
17      public String getName() {
18              return name;
19      }
20 }
21
```

Deck

```java
package war;

import java.util.Collections;
import java.util.LinkedList;
import java.util.List;

public class Deck extends LinkedList<Card>{

    private List<Integer> values = List.of(2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14);
    private List<String> names = List.of(" of Hearts"," of Clubs"," of Diamonds"," of Spades");

    public Deck(){
        for(int i = 0; i <values.size(); i++) {
            Integer value = values.get(i);

            for(String name : names) {
                add(new Card(value, name));
        }
        }
    }
    public List<Card> shuffle(List<Card> deck){
        Collections.shuffle(deck);
        return deck;
    }
        public Card draw() {
        return remove(0);
    }
}
```

Player:

```java
package war;

import java.util.LinkedList;
import java.util.List;

public class Player {

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


    public void setName(String name) {
        this.name = name;
    }
    public void draw(Deck deck) {
        hand.add(deck.draw());
    }
    public Card flip() {
        return hand.remove(0);
    }
    public int getScore(){
        return score;
    }
    public void incrementScore() {
        score += 1;
    }
    public void describe() {
        System.out.println(name + ": " + score);
    }
    public String toString() {
        StringBuilder x = new StringBuilder();
        return name;
    }
}
```

Screenshots of Running Application:

```
33        public static void play(Player player1, Player player2) {
34        for(int i = 0; i < 26; i++) {
35            Card card1 = player1.flip();
36            Card card2 = player2.flip();
37            if(card1.getValue() > card2.getValue()) {
38                player1.incrementScore();
39            }
40            else if (card1.getValue() < card2.getValue()) {
41                player2.incrementScore();
42
```

Problems @ Javadoc Declaration Console

<terminated> App [Java Application] C:\Program Files\Amazon Corretto\jdk17.0.2_8\bin\javaw.exe (Apr 2, 2022, 4:30:23 PM – 4:30:23 PM)

```
Marvel: 16
Murphy: 9
Marvel has won
```

URL to GitHub Repository:

https://github.com/mttsgr/JavaWar