

# 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:

(note: just recently, I changed my IDE to night-mode and it occurred to me that this may be slightly harder to read for some graders (or easier?), if you would like me to send screenshots in the regular non-night-mode please let me know and I will accommodate.)

## Card Class:

```
1
2 public class Card {
3
4     private int value; //contains a value from 2-14 (representing 2 to ace high)
5     private int suit; //0-3: hearts, spades, diamonds, clubs
6
7
8     public Card (int value, int suit) { //constructor
9         this.value = value;
10        this.suit = suit;
11    }
12
13    public void describe() { //method that prints out information about a card.
14        System.out.println(value + " of " + suit);
15    }
16
17    //Getters & Setters
18    public int getValue() {
19        return value;
20    }
21
22
23    public void setValue(int value) {
24        this.value = value;
25    }
26
27
28    public int getSuit() {
29        return suit;
30    }
31
32    public void setSuit(int suit) {
33        this.suit = suit;
34    }
35 } //end class
36
```

## Deck Class:

```
Card.java App.java Deck.java Player.java
1 import java.util.ArrayList;
2 import java.util.Collections;
3 import java.util.List;
4
5 public class Deck {
6
7     public List<Card> deck; //list of cards, initialized in the constructor below
8
9     public Deck() { //deck constructor
10         deck = new ArrayList<>();
11         unpack();
12     }
13
14     public void shuffle() { //randomizes the order of the cards in the deck
15         Collections.shuffle(deck);
16     }
17
18     public Card draw() { // removes and returns the top card of the cards field
19         return deck.remove(deck.size() - 1);
20     }
21
22     private void unpack() {
23         for ( int suit = 0; suit < 4; suit++ ) { // builds the suits
24             for (int value = 2; value < 15; value ++ ) { //builds the values within the suits
25                 deck.add( new Card ( value , suit ) ); //creates the cards inside the deck
26             }
27         }
28     }
29
30 }
31
32
```

## Player Class:

```
Card.java App.java Deck.java *Player.java x
4 public class Player {
5
6 //Fields
7     public List<Card> hand; //list of card in hand
8     public int score; //set to 0 in the constructor
9     public String name; // player name
10
11 //Constructor
12●     public Player (int score, String name) {
13         this.score = 0;
14         this.name = name;
15         hand = new ArrayList<Card>();
16     }
17
18 //Methods
19 //1. describePlayer (prints out information about the player AND calls the describe method for each
20●     public void describePlayer() {
21         System.out.println( name + ", after drawing, has the following cards in their hand: ");
22         for ( Card card : hand ) {
23             card.describe();
24         }
25     }
26 //2. flip (removes and returns the top card of the Hand)
27●     public Card flip() {
28         return hand.remove(hand.size() - 1);
29     }
30 //3. draw (takes a Deck as an argument and calls the draw method on the deck, adding the returned Ca
31 //The function of this method occurs when the players draw their cards from the deck in the
32 //them to their hands.
33
34 //4. incrementScore (adds 1 to the Player's score field)
35●     public void incrementScore() {
36         score = score + 1;
37     }
38 } //end class
39
```

## App Class (main method):

### part 1:

```
Card.java  *App.java  Deck.java  *Player.java
1 public class App {
2
3     public static void main(String[] args) {
4
5         //3. Instantiate a Deck and two Players, call the shuffle method on the deck.
6         Deck gameDeck = new Deck();
7
8         Player p1 = new Player(0, "Player 1");
9         Player p2 = new Player(0, "Player 2");
10
11         gameDeck.shuffle();
12
13         //4. Using a traditional for loop, iterate 52 times calling the Draw method on the other player
14         for (int i = 0; i < 52; i++) {
15             if (i % 2 == 0) {
16                 p1.hand.add( gameDeck.draw() );
17             } else {
18                 p2.hand.add( gameDeck.draw() );
19             }
20         }
21         p1.describePlayer();
22         p2.describePlayer();
23         //5. Using a traditional for loop, iterate 26 times and call the flip method for each player.
24         //     a. Compare the value of each card returned by the two player's flip methods.
25         //     b. Call the incrementScore method on the player whose card has the higher value.
```

### part 2:

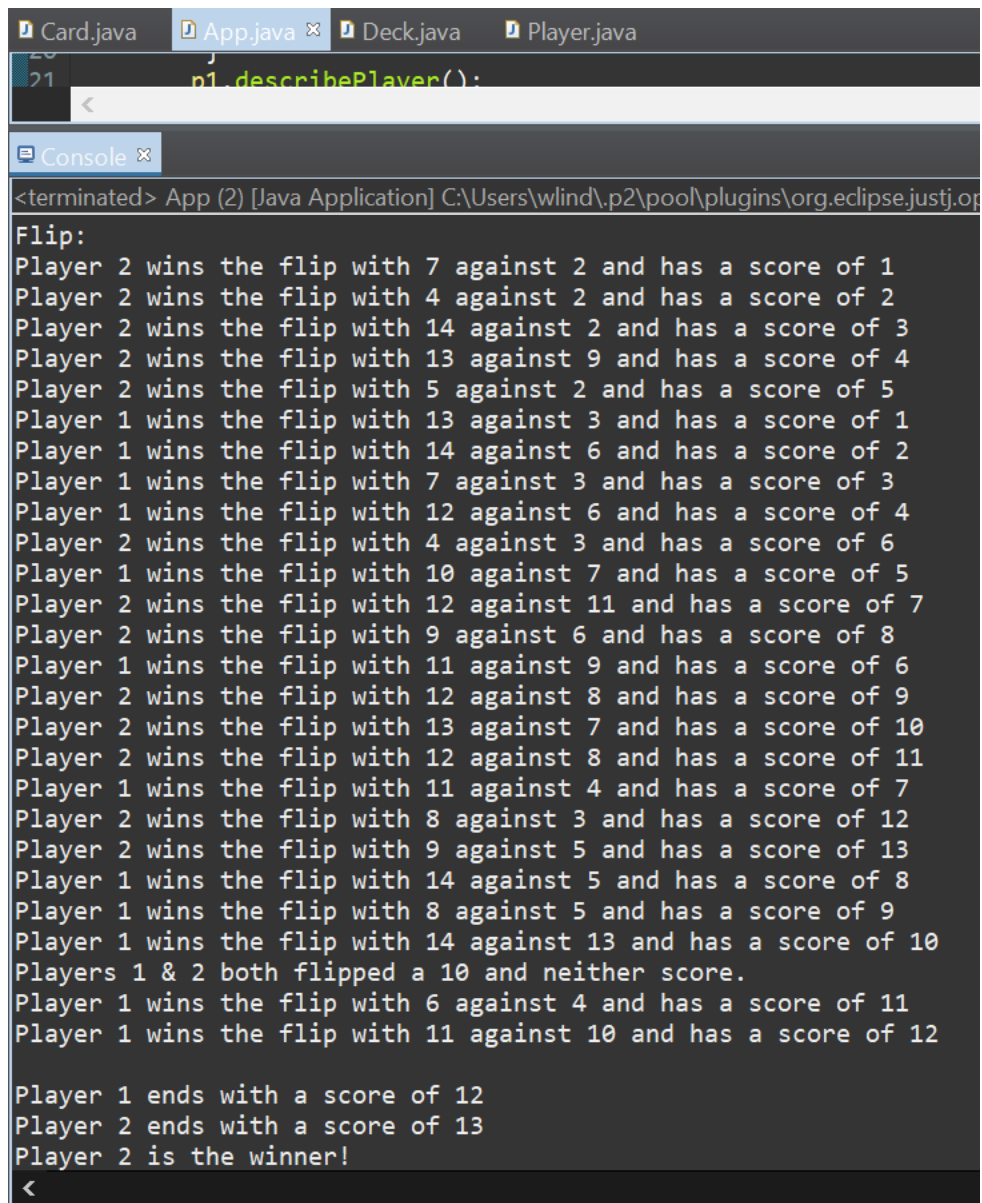
```
23 //5. Using a traditional for loop, iterate 26 times and call the flip method for each player.
24 //     a. Compare the value of each card returned by the two player's flip methods.
25 //     b. Call the incrementScore method on the player whose card has the higher value.
26 System.out.println("\nFlip:");
27 for (int p = 0; p < 26; p++) {
28     Card c1 = p1.flip();
29     Card c2 = p2.flip();
30     if (c1.getValue() > c2.getValue()) {
31         p1.incrementScore();
32         System.out.println("Player 1 wins the flip with " + c1.getValue() + " against " + c2.getValue() + " and has a score of " + p1.score);
33     } if (c2.getValue() > c1.getValue()){
34         p2.incrementScore();
35         System.out.println("Player 2 wins the flip with " + c2.getValue() + " against " + c1.getValue() + " and has a score of " + p2.score);
36     } if (c1.getValue() == c2.getValue()) {}
37     System.out.println("Players 1 & 2 both flipped a " + c1.getValue() + " and neither score."); //c1.getValue is only there to show the t
38 }
39
40
41 //6. After the loop, compare the final score from each player.
42 //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
43 System.out.println("\nPlayer 1 ends with a score of " + p1.score);
44 System.out.println("Player 2 ends with a score of " + p2.score);
45
46 if (p1.score > p2.score) {
47     System.out.println("Player 1 is the winner!");
48 } if (p2.score > p1.score) {
49     System.out.println("Player 2 is the winner!");
50 } if (p2.score == p1.score) {
51     System.out.println("Player 1 and Player 2 have tied in a draw!");
52 }
53
54 } //end main method
55
56 } //end App
```

## Screenshots of Running Application:

### Player's hand description:

```
20  
21 p1.describePlayer():  
<terminated> App (2) [Java Application] C:\Users\wland\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot  
Player 1, after drawing, has the following cards in their hand:  
11 of 3  
6 of 2  
10 of 1  
14 of 2  
8 of 0  
14 of 0  
5 of 1  
3 of 1  
11 of 0  
8 of 3  
7 of 3  
8 of 1  
11 of 2  
6 of 3  
11 of 1  
10 of 3  
3 of 0  
12 of 0  
7 of 2  
14 of 3  
13 of 2  
2 of 3  
9 of 0  
2 of 0  
2 of 1  
2 of 2  
Player 2, after drawing, has the following cards in their hand:  
10 of 2  
4 of 3  
10 of 0
```

### Flipping of cards and scoring:



The screenshot shows an Eclipse IDE with four Java files open: Card.java, App.java, Deck.java, and Player.java. The App.java file is active, showing a call to p1.describePlayer(). Below the code editor is a console window titled 'Console' showing the output of the application. The output displays a series of card flips and scores for two players, concluding with 'Player 2 is the winner!'.

```
<terminated> App (2) [Java Application] C:\Users\wland\p2\pool\plugins\org.eclipse.justj.o  
Flip:  
Player 2 wins the flip with 7 against 2 and has a score of 1  
Player 2 wins the flip with 4 against 2 and has a score of 2  
Player 2 wins the flip with 14 against 2 and has a score of 3  
Player 2 wins the flip with 13 against 9 and has a score of 4  
Player 2 wins the flip with 5 against 2 and has a score of 5  
Player 1 wins the flip with 13 against 3 and has a score of 1  
Player 1 wins the flip with 14 against 6 and has a score of 2  
Player 1 wins the flip with 7 against 3 and has a score of 3  
Player 1 wins the flip with 12 against 6 and has a score of 4  
Player 2 wins the flip with 4 against 3 and has a score of 6  
Player 1 wins the flip with 10 against 7 and has a score of 5  
Player 2 wins the flip with 12 against 11 and has a score of 7  
Player 2 wins the flip with 9 against 6 and has a score of 8  
Player 1 wins the flip with 11 against 9 and has a score of 6  
Player 2 wins the flip with 12 against 8 and has a score of 9  
Player 2 wins the flip with 13 against 7 and has a score of 10  
Player 2 wins the flip with 12 against 8 and has a score of 11  
Player 1 wins the flip with 11 against 4 and has a score of 7  
Player 2 wins the flip with 8 against 3 and has a score of 12  
Player 2 wins the flip with 9 against 5 and has a score of 13  
Player 1 wins the flip with 14 against 5 and has a score of 8  
Player 1 wins the flip with 8 against 5 and has a score of 9  
Player 1 wins the flip with 14 against 13 and has a score of 10  
Players 1 & 2 both flipped a 10 and neither score.  
Player 1 wins the flip with 6 against 4 and has a score of 11  
Player 1 wins the flip with 11 against 10 and has a score of 12  
  
Player 1 ends with a score of 12  
Player 2 ends with a score of 13  
Player 2 is the winner!  
<
```

URL to GitHub Repository:

<https://github.com/wlindstrom55/week-6-assignments>