

# 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:

eclipse-workspace - Week6-Coding-Assignment/src/cardGame/Card.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer

- javawork1Lab1
- Java Week 1 Lab Part 1
- MoneyPractice
- Money
- Mountain-BM1
- Mountain Java FizzBuzz
- Mountain Java Week2
- Mountain Java Week3
- Mountain Java Week4
- Mountain Java Week5
- Mountain Java Week6
- Mountain Java WeekPart2
- Mountain-Week1
- MyFirstProject
- Test
- Week2
- Week2
- Week2 Labs
- Week3 Coding Assignment
- Week3 Labs
- Week3 Videos
- Week4 Labs
- Week4 Videos
- Week5 Coding Assignment
- Week5 Videos
- Week5 Video Menu Driven App
- Week6 Coding Assignment
- Week6 Videos

Card.java Deck.java Player.java App.java

```
1 package cardGame;
2
3 public class Card {
4
5     int value;
6     String name;
7
8     public void setValue(int newValue) {
9         this.value = newValue;
10    }
11
12    public int getValue() {
13        return value;
14    }
15
16    public void setName(String newName) {
17        this.name = newName;
18    }
19
20    public String getName() {
21        return name;
22    }
23
24    public void describe() {
25        System.out.println(value + " of " + name);
26    }
27
28 }
29
30
31
```

Problems | Inspect | Declaration | Console

terminated: App [1] Java Application C:\Program Files\Java\jdk-17.0.2\bin\java.exe [Apr 10, 2022, 11:12:37 PM - 11:12:37 PM]

Player 1 final score = 13  
Player 2 final score = 11  
Player 1

Workspace Smart Insert 31 / 2,483 100% 256M

eclipse-workspace - Week6-Coding-Assignment/src/cardGame/Deck.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer

- javawork1Lab1
- Java Week 1 Lab Part 1
- MoneyPractice
- Money
- Mountain-BM1
- Mountain Java FizzBuzz
- Mountain Java Week2
- Mountain Java Week3
- Mountain Java Week4
- Mountain Java Week5
- Mountain Java Week6
- Mountain Java WeekPart2
- Mountain-Week1
- MyFirstProject
- Test
- Week2
- Week2
- Week2 Labs
- Week3 Coding Assignment
- Week3 Labs
- Week3 Videos
- Week4 Labs
- Week4 Videos
- Week5 Coding Assignment
- Week5 Videos
- Week5 Video Menu Driven App
- Week6 Coding Assignment
- Week6 Videos

Card.java Deck.java Player.java App.java

```
1 package cardGame;
2
3 import java.util.ArrayList;
4
5 public class Deck {
6
7     List<Card> cards;
8
9     Map<Integer, String> suits = new HashMap<Integer, String>();
10
11     Map<Integer, String> cardNames = new HashMap<Integer, String>();
12
13     private void loadMaps() {
14         cardNames.put(1, "Two");
15         cardNames.put(2, "Three");
16         cardNames.put(3, "Four");
17         cardNames.put(4, "Five");
18         cardNames.put(5, "Six");
19         cardNames.put(6, "Seven");
20         cardNames.put(7, "Eight");
21         cardNames.put(8, "Nine");
22         cardNames.put(9, "Ten");
23         cardNames.put(10, "Jack");
24         cardNames.put(11, "Queen");
25         cardNames.put(12, "King");
26         cardNames.put(13, "Ace");
27         suits.put(0, "Spades");
28         suits.put(1, "Hearts");
29         suits.put(2, "Clubs");
30         suits.put(3, "Diamonds");
31     }
32
33     public Deck() {
34         loadMaps();
35         cards = new ArrayList<Card>();
36         for (int i = 0; i < 52; i++) {
37             Card c = new Card();
38             c.setValue(i);
39             c.setName(cardNames.get(i) + " of " + suits.get(i/13));
40             cards.add(c);
41         }
42     }
43
44     public void shuffle() {
45         Collections.shuffle(cards);
46     }
47
48     public Card draw() {
49         Card drawnCard = cards.get(0);
50         cards.remove(0);
51         return drawnCard;
52     }
53
54 }
55
56
57
```

Problems | Inspect | Declaration | Console

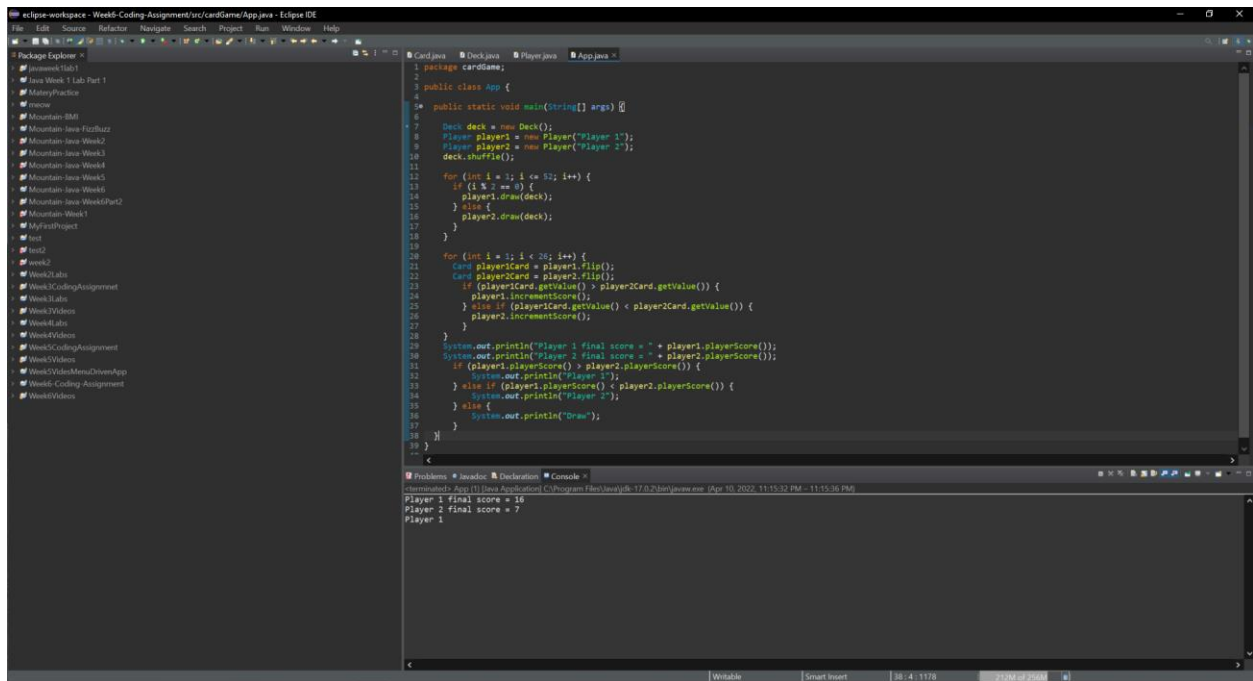
terminated: App [1] Java Application C:\Program Files\Java\jdk-17.0.2\bin\java.exe [Apr 10, 2022, 11:12:37 PM - 11:12:37 PM]

Player 1 final score = 13  
Player 2 final score = 11  
Player 1

Workspace Smart Insert 35 / 5,907 100% 256M

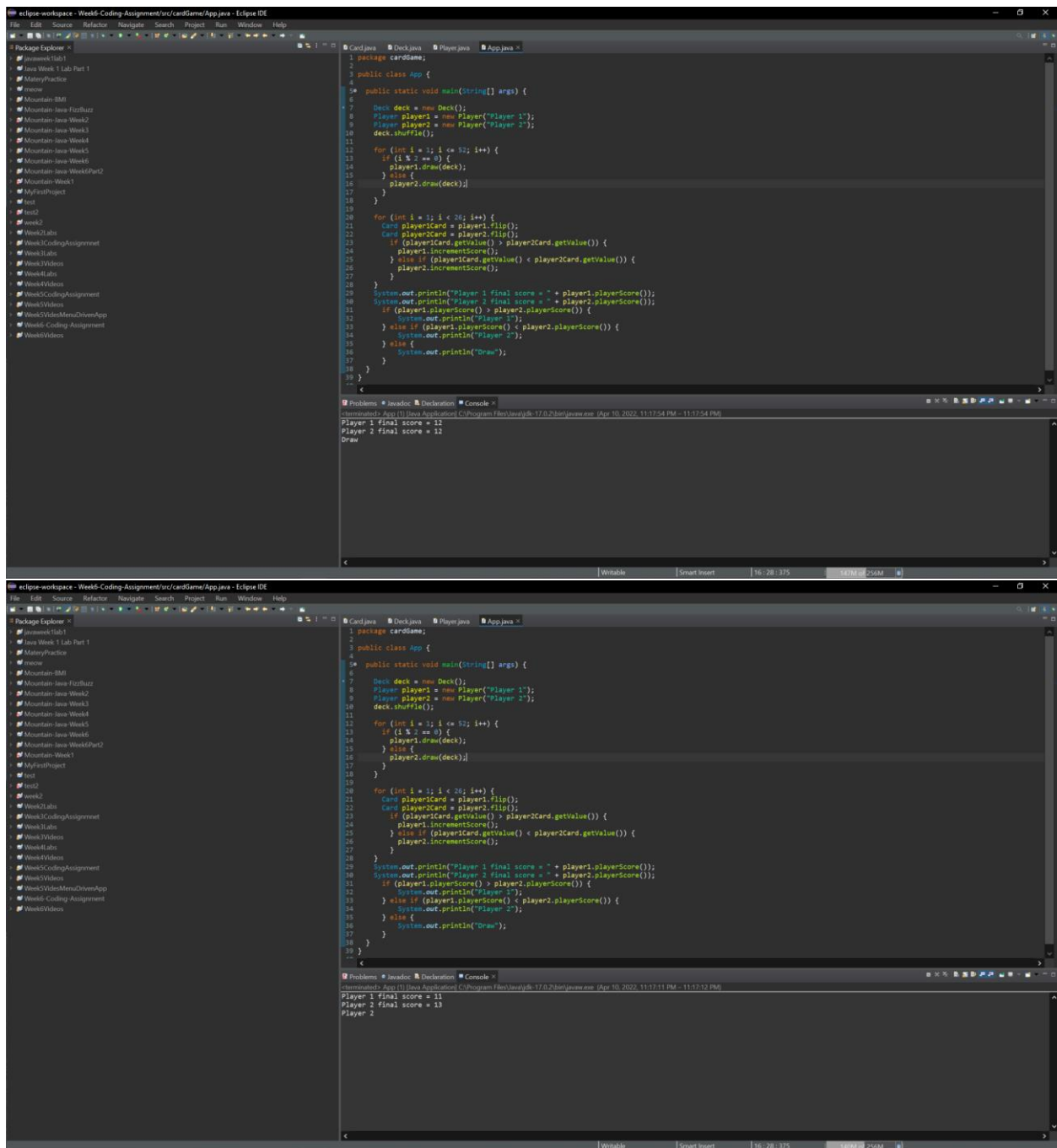


## Screenshots of Running Application:



```
1 package cardGame;
2
3 public class App {
4
5     public static void main(String[] args) {
6
7         Deck deck = new Deck();
8         Player player1 = new Player("Player 1");
9         Player player2 = new Player("Player 2");
10        deck.shuffle();
11
12        for (int i = 1; i <= 52; i++) {
13            if (i % 2 == 0) {
14                player1.draw(deck);
15            } else {
16                player2.draw(deck);
17            }
18        }
19
20        for (int i = 1; i <= 26; i++) {
21            Card player1Card = player1.flip();
22            Card player2Card = player2.flip();
23            if (player1Card.getValue() > player2Card.getValue()) {
24                player1.incrementScore();
25            } else if (player1Card.getValue() < player2Card.getValue()) {
26                player2.incrementScore();
27            }
28        }
29        System.out.println("Player 1 final score = " + player1.playerScore());
30        System.out.println("Player 2 final score = " + player2.playerScore());
31        if (player1.playerScore() > player2.playerScore()) {
32            System.out.println("Player 1");
33        } else if (player1.playerScore() < player2.playerScore()) {
34            System.out.println("Player 2");
35        } else {
36            System.out.println("Draw");
37        }
38    }
39 }
40 }
```

Player 1 final score = 16  
Player 2 final score = 7  
Player 1



URL to GitHub Repository:

<https://github.com/LoganMunsterman/Week6JavaCodingAssignment>