

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.

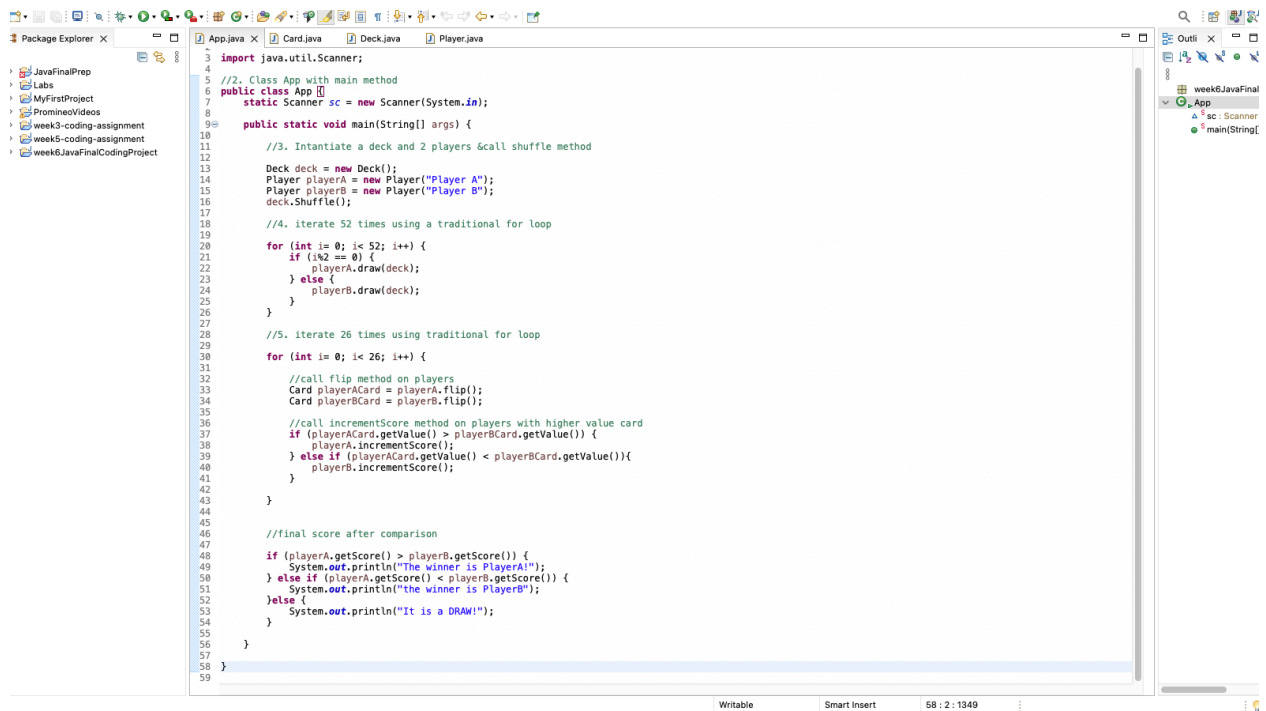
Coding Steps:

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)

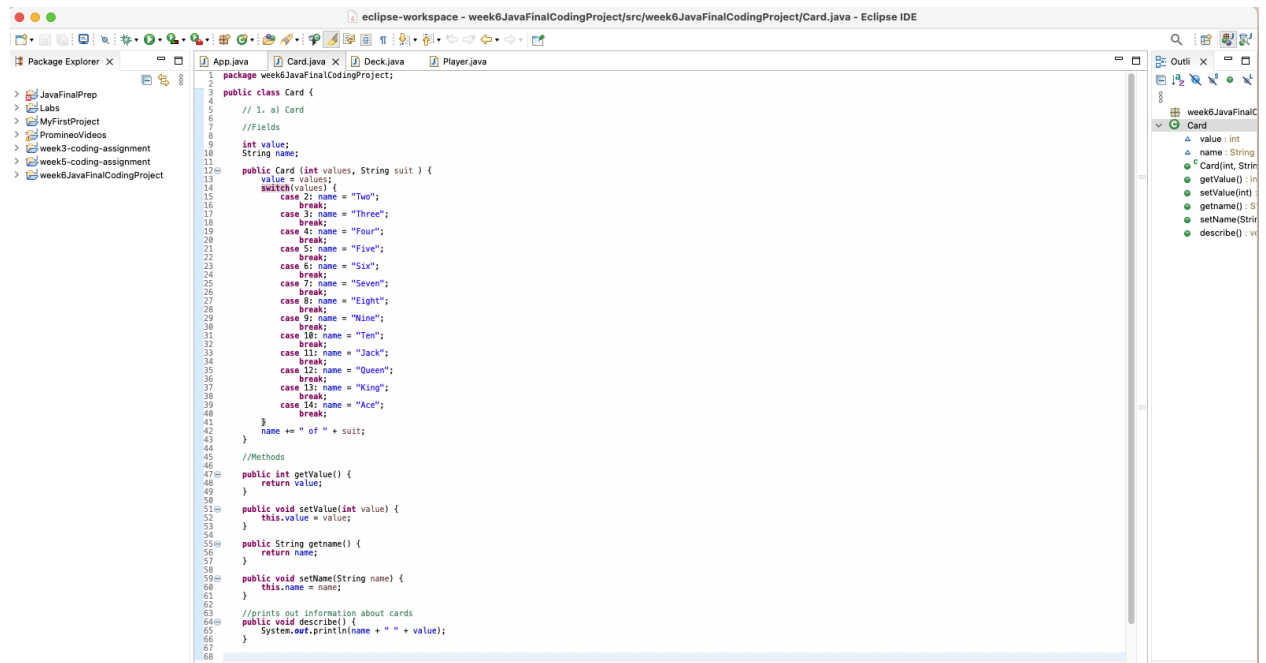
3. In the constructor, when a new Deck is instantiated, the Cards field should be populated with the standard 52 cards.
- c. Player
 - i. Fields
 1. **hand** (List of Card)
 2. **score** (set to 0 in the constructor)
 3. **name**
 - ii. Methods
 1. **describe** (prints out information about the player 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:



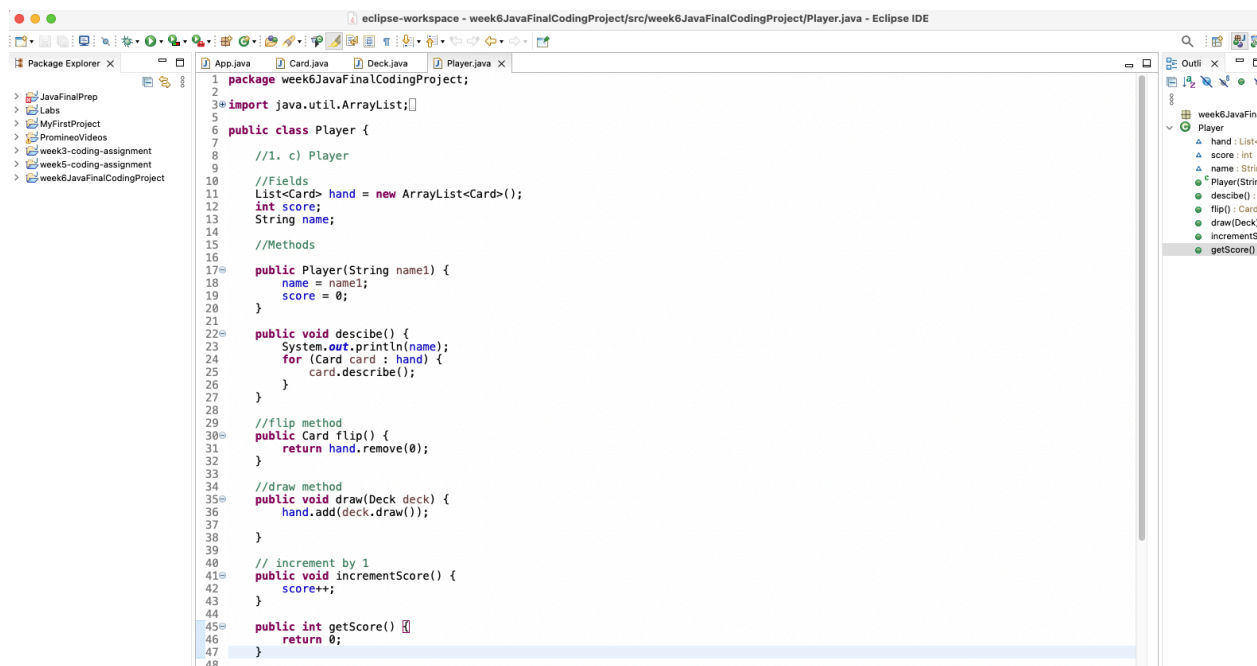
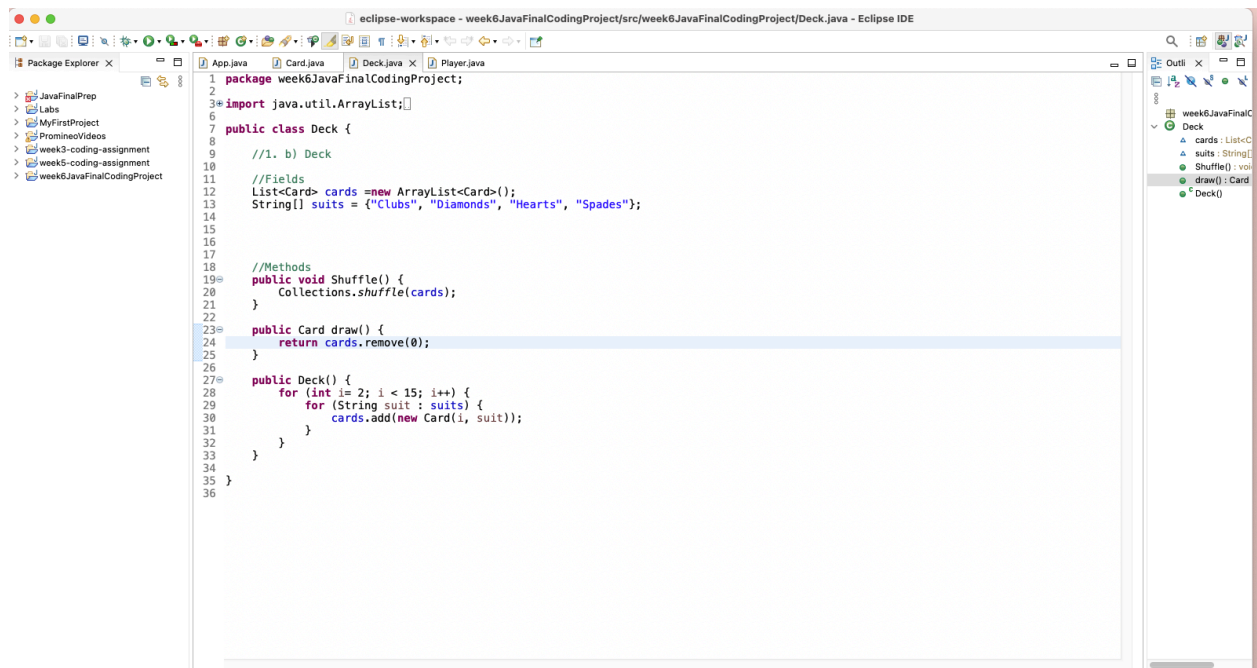
This screenshot shows the Eclipse IDE with the 'App.java' file open. The code implements a card game logic where two players, Player A and Player B, are dealt cards from a deck. The game proceeds through several steps: shuffling the deck, dealing cards to both players, and then a series of iterations where cards are flipped and scores are updated based on the card values. The game ends by comparing the final scores to determine the winner or if it's a draw.

```
1 import java.util.Scanner;
2
3 //2. Class App with main method
4 public class App {
5     static Scanner sc = new Scanner(System.in);
6
7     public static void main(String[] args) {
8
9         //3. Instantiate a deck and 2 players & call shuffle method
10
11         Deck deck = new Deck();
12         Player playerA = new Player("Player A");
13         Player playerB = new Player("Player B");
14         deck.Shuffle();
15
16         //4. Iterate 52 times using a traditional for loop
17
18         for (int i = 0; i < 52; i++) {
19             if (i % 2 == 0) {
20                 playerA.draw(deck);
21             } else {
22                 playerB.draw(deck);
23             }
24         }
25
26         //5. Iterate 26 times using traditional for loop
27
28         for (int i = 0; i < 26; i++) {
29
30             //call flip method on players
31             Card playerCard = playerA.flip();
32             Card playerBCard = playerB.flip();
33
34             //call incrementScore method on players with higher value card
35             if (playerACard.getValue() > playerBCard.getValue()) {
36                 playerA.incrementScore();
37             } else if (playerACard.getValue() < playerBCard.getValue()) {
38                 playerB.incrementScore();
39             }
40
41             //final score after comparison
42
43             if (playerA.getScore() > playerB.getScore()) {
44                 System.out.println("The winner is Player A!");
45             } else if (playerA.getScore() < playerB.getScore()) {
46                 System.out.println("The winner is Player B");
47             } else {
48                 System.out.println("It is a DRAW!");
49             }
50
51         }
52     }
53 }
54
55
56
57
58
59
```

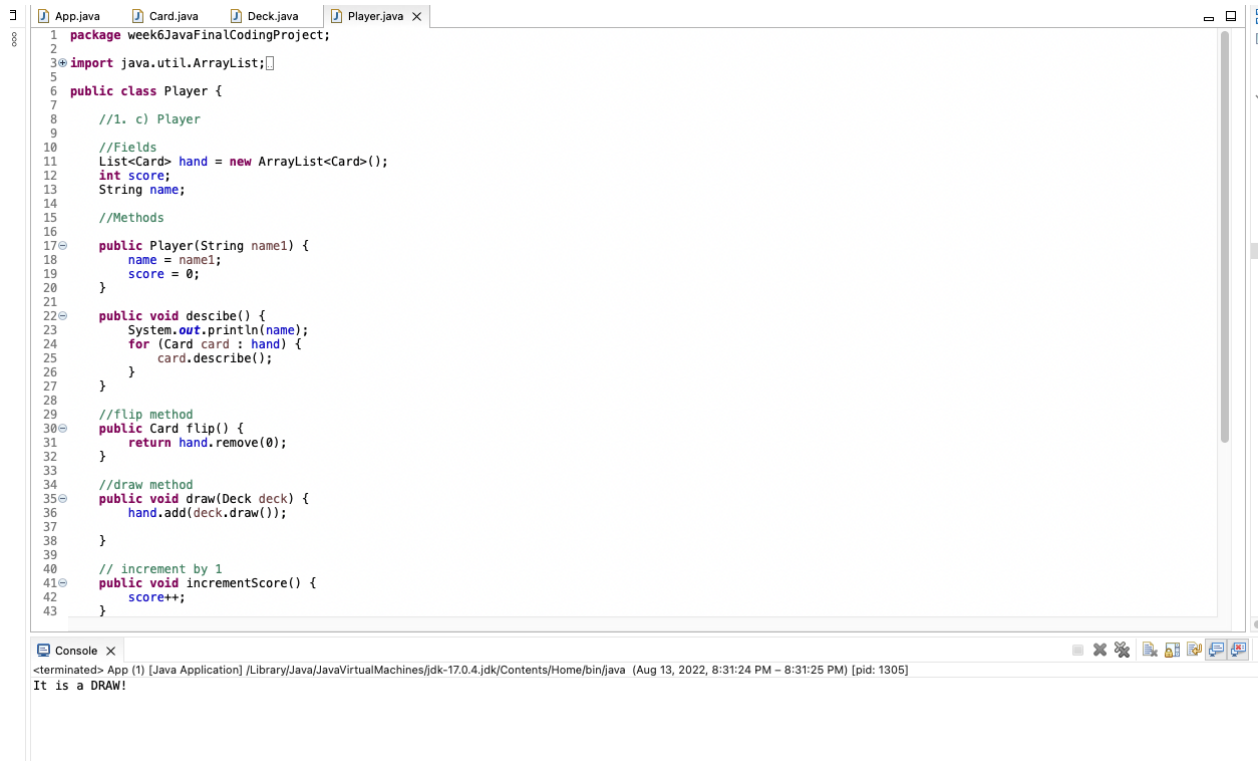


This screenshot shows the Eclipse IDE with the 'Card.java' file open. The code defines a 'Card' class with attributes for 'value' and 'name'. It includes a constructor that takes a value and a suit, and a method to generate the card's name based on its value. The class also has methods to get and set the value and name, and a 'describe()' method that prints out the card's details.

```
1 package week6.JavaFinalCodingProject;
2
3 public class Card {
4     // 1. a) Card
5     //Fields
6     int value;
7     String name;
8
9     public Card (int values, String suit) {
10         value = values;
11         switch(values) {
12             case 2: name = "Two";
13             break;
14             case 3: name = "Three";
15             break;
16             case 4: name = "Four";
17             break;
18             case 5: name = "Five";
19             break;
20             case 6: name = "Six";
21             break;
22             case 7: name = "Seven";
23             break;
24             case 8: name = "Eight";
25             break;
26             case 9: name = "Nine";
27             break;
28             case 10: name = "Ten";
29             break;
30             case 11: name = "Jack";
31             break;
32             case 12: name = "Queen";
33             break;
34             case 13: name = "King";
35             break;
36             case 14: name = "Ace";
37             break;
38         }
39         name += " of " + suit;
40     }
41
42     //Methods
43
44     public int getValue() {
45         return value;
46     }
47
48     public void setValue(int value) {
49         this.value = value;
50     }
51
52     public String getName() {
53         return name;
54     }
55
56     public void setName(String name) {
57         this.name = name;
58     }
59
60     //prints out information about cards
61     public void describe() {
62         System.out.println(name + " = " + value);
63     }
64 }
65
66
67
68
```



Screenshots of Running Application:



The screenshot shows an IDE with four tabs: App.java, Card.java, Deck.java, and Player.java. The Player.java tab is active, displaying the following code:

```
1 package week6JavaFinalCodingProject;
2
3 import java.util.ArrayList;
4
5
6 public class Player {
7
8     //1. c) Player
9
10    //Fields
11    List<Card> hand = new ArrayList<Card>();
12    int score;
13    String name;
14
15    //Methods
16
17    public Player(String name1) {
18        name = name1;
19        score = 0;
20    }
21
22    public void describe() {
23        System.out.println(name);
24        for (Card card : hand) {
25            card.describe();
26        }
27    }
28
29    //flip method
30    public Card flip() {
31        return hand.remove(0);
32    }
33
34    //draw method
35    public void draw(Deck deck) {
36        hand.add(deck.draw());
37    }
38
39    // increment by 1
40    public void incrementScore() {
41        score++;
42    }
43 }
```

The console window at the bottom shows the following output:

```
<terminated> App (1) [Java Application] /Library/Java/JavaVirtualMachines/jdk-17.0.4.jdk/Contents/Home/bin/java (Aug 13, 2022, 8:31:24 PM - 8:31:25 PM) [pid: 1305]
It is a DRAW!
```

URL to GitHub Repository:

<https://github.com/AshaAbdulkadir/Week6-Final-coding-assignment->