

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

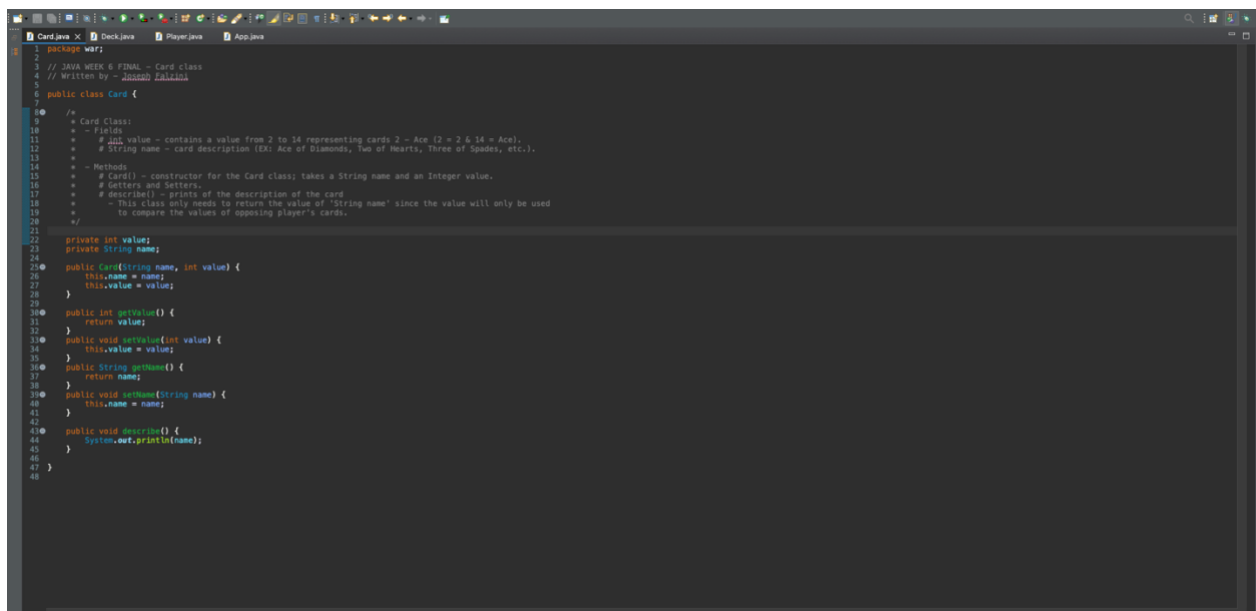
i. Fields

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 war;
2
3 // JAVA WEEK 6 FINAL - Card class
4 // Written by - Jannah Salinas
5
6 public class Card {
7
8     /*
9     * Card Class:
10    * - Fields
11    *   # int value - contains a value from 2 to 14 representing cards 2 - Ace (2 = 2 & 14 = Ace).
12    *   # String name - card description (Ex: Ace of Diamonds, Two of Hearts, Three of Spades, etc.).
13    * - Methods
14    *   # Card() - constructor for the Card class; takes a String name and an Integer value.
15    *   # Setters and Getters.
16    *   # describe() - prints the description of the card
17    *   # This class only needs to return the value of 'String name' since the value will only be used
18    *     to compare the values of opposing player's cards.
19    */
20
21     private int value;
22     private String name;
23
24     public Card(String name, int value) {
25         this.name = name;
26         this.value = value;
27     }
28
29     public int getValue() {
30         return value;
31     }
32
33     public void setValue(int value) {
34         this.value = value;
35     }
36
37     public String getName() {
38         return name;
39     }
40
41     public void setName(String name) {
42         this.name = name;
43     }
44
45     public void describe() {
46         System.out.println(name);
47     }
48 }
```

```

54 // Spades
55 deckOfCards.add(new Card("Ace of Spades", 14));
56 deckOfCards.add(new Card("Two of Spades", 11);
57 deckOfCards.add(new Card("Three of Spades", 13);
58 deckOfCards.add(new Card("Four of Spades", 12);
59 deckOfCards.add(new Card("Five of Spades", 10);
60 deckOfCards.add(new Card("Six of Spades", 9);
61 deckOfCards.add(new Card("Seven of Spades", 7);
62 deckOfCards.add(new Card("Eight of Spades", 8);
63 deckOfCards.add(new Card("Nine of Spades", 6);
64 deckOfCards.add(new Card("Ten of Spades", 5);
65 deckOfCards.add(new Card("Jack of Spades", 13);
66 deckOfCards.add(new Card("Queen of Spades", 12);
67 deckOfCards.add(new Card("King of Spades", 14);
68 // Diamonds
69 deckOfCards.add(new Card("Ace of Diamonds", 14);
70 deckOfCards.add(new Card("Two of Diamonds", 11);
71 deckOfCards.add(new Card("Three of Diamonds", 13);
72 deckOfCards.add(new Card("Four of Diamonds", 12);
73 deckOfCards.add(new Card("Five of Diamonds", 10);
74 deckOfCards.add(new Card("Six of Diamonds", 9);
75 deckOfCards.add(new Card("Seven of Diamonds", 7);
76 deckOfCards.add(new Card("Eight of Diamonds", 8);
77 deckOfCards.add(new Card("Nine of Diamonds", 6);
78 deckOfCards.add(new Card("Ten of Diamonds", 5);
79 deckOfCards.add(new Card("Jack of Diamonds", 13);
80 deckOfCards.add(new Card("Queen of Diamonds", 12);
81 deckOfCards.add(new Card("King of Diamonds", 14);
82
83 }
84
85 public void shuffle() {
86     Collections.shuffle(deckOfCards); // Collections.shuffle() is used to shuffle lists in Java.
87 }
88
89 public Card draw() {
90     Card drawn = deckOfCards.get(0); // Grab the top card.
91     deckOfCards.remove(0); // Removes card from deck.
92     return drawn;
93 }
94
95 }
96
97
98
99
100

```

```
1 package war;
2
3 import java.util.ArrayList;
4
5 // JAVA WEEK 6 FINAL - Player class
6 // Written by - Fahim Fahim
7
8 public class Player {
9
10     /*
11     * Player Class:
12     * Fields:
13     * # List<Card> hand - List of Card.
14     * # int score - Player's score, set to 0 by the constructor.
15     * # String name - Player name.
16     *
17     * Methods:
18     * # Player(String name) - class constructor, takes a String name for the Player name.
19     * # describe() - prints out information about the player, and calls the describe method for each card in List hand.
20     * # flip() - removes and returns the top card of the player's hand.
21     * # draw(deck deck) - takes a Deck as an argument and takes the top card from the deck and puts it into the player's hand.
22     * # incrementScore() - adds 1 to the Player's total score.
23     * # getName() - returns the player's name.
24     * # getScore() - returns the player's score.
25     */
26
27     private List<Card> hand = new ArrayList<Card>();
28     private int score;
29     private String name;
30
31     public Player(String name) {
32         this.name = name;
33         score = 0;
34     }
35
36     public void describe() {
37         System.out.println("Player name: " + name);
38         for (int i = 0; i < hand.size(); i++) {
39             Card card = hand.get(i);
40             card.describe();
41         }
42     }
43
44     public Card flip() {
45         Card flipped = hand.get(0);
46         hand.remove(0);
47         return flipped;
48     }
49
50     public void draw(deck deck) {
51         Card draw = deck.draw();
52         hand.add(draw);
53     }
54
55     public void incrementScore() {
56         score += 1;
57     }
58
59     public String getName() {
60         return name;
61     }
62
63     public int getScore() {
64         return score;
65     }
66 }
67
68
```

```
1 package war;
2
3 // JAVA WEEK 6 FINAL - App class
4 // Written by - Fahim Fahim
5
6 public class App {
7
8     /*
9     * PROJECT INSTRUCTIONS:
10     * 1. Initialize a Deck and two Players, call the shuffle method on the deck.
11     * 2. Using a traditional for-loop, iterate 52 times calling the draw() method on the other player each iteration using
12     * the Deck you instantiated.
13     * 3. Using a traditional for-loop, iterate 26 times and call the flip() method for each player.
14     * 4. Compare the value of each card returned by the two player's flip() methods. Call the incrementScore() method
15     * on the player with the highest card value.
16     * 5. After the loop, compare the final score from each player.
17     * 6. Print the final score of each player and either "Player 1", "Player 2", or "Draw" depending on which score is higher
18     * or if they are both the same.
19     */
20
21     /*
22     * WAR Card Game Rules:
23     *
24     * # This project simplifies the game and does not require a "War" in the case of a draw.
25     * # A draw results in no points given to either Player.
26     *
27     * 1. Each player turns up a card at the same time and the player with the higher card takes both cards and puts them,
28     * face down, on the bottom of his deck.
29     * 2. If the cards are the same rank, it is War. Each player turns up one card face down and one card face up.
30     * The player with the higher cards takes both piles (six cards). If the turned-up cards are again the same rank,
31     * each player places another card face down and turns another card face up.
32     */
33
34     public static void main(String[] args) {
35         Deck deck = new Deck();
36         Player player1 = new Player("Joseph");
37         Player player2 = new Player("Julianne");
38         deck.shuffle();
39
40         // 26 cards per player
41         for (int i = 0; i < 52; i++) {
42             if (i % 2 == 0) {
43                 player1.draw(deck);
44             } else if (i % 2 == 1) {
45                 player2.draw(deck);
46             }
47         }
48
49         // player1.describe(); - used to verify each player gets 26 cards.
50         // player2.describe(); - used to verify each player gets 26 cards.
51
52         // 26 Rounds of Play: each iteration tells the round # and each player's card, along with the winner of the round.
53         for (int i = 0; i < 26; i++) {
54             System.out.println("Round " + (i + 1) + ": ");
55             Card p1 = player1.flip();
56             Card p2 = player2.flip();
57             int p1Value = p1.getValue();
58             int p2Value = p2.getValue();
59
60             System.out.println(player1.getName() + "'s card - " + p1.getName());
61             System.out.println(player2.getName() + "'s card - " + p2.getName());
62
63             if (p1Value > p2Value) {
64                 System.out.println(player1.getName() + " wins the round!\n");
65                 player1.incrementScore();
66             } else if (p1Value < p2Value) {
67                 System.out.println(player2.getName() + " wins the round!\n");
68                 player2.incrementScore();
69             } else {
70                 // Draw
71             }
72         }
73     }
74 }
```

```
50 // player1.describe(); - used to verify each player gets 26 cards.
51 // player2.describe(); - used to verify each player gets 26 cards.
52
53 // 26 Rounds of Play; each iteration tells the round # and each player's card, along with the winner of the round.
54 for (int i = 0; i < 26; i++) {
55     System.out.println("Round " + (i + 1) + ":");
56     Card p1 = player1.flip();
57     Card p2 = player2.flip();
58     int p1Value = p1.getValue();
59     int p2Value = p2.getValue();
60
61     System.out.println(player1.getName() + "'s card - " + p1.getName());
62     System.out.println(player2.getName() + "'s card - " + p2.getName());
63
64     if (p1Value > p2Value) {
65         System.out.println(player1.getName() + " wins the round!\n");
66         player1.incrementScore();
67     } else if (p1Value < p2Value) {
68         System.out.println(player2.getName() + " wins the round!\n");
69         player2.incrementScore();
70     } else {
71         System.out.println("DRAW! No score changes!\n");
72     }
73 }
74
75 int player1Score = player1.getScore();
76 int player2Score = player2.getScore();
77
78 System.out.println("Results: " + player1.getName() + " - " + player1Score +
79     " | " + player2.getName() + " - " + player2Score);
80
81 if (player1Score > player2Score) {
82     System.out.println(player1.getName() + " has won the game!!!");
83 } else if (player1Score < player2Score) {
84     System.out.println(player2.getName() + " has won the game!!!");
85 } else {
86     System.out.println("DRAW!");
87 }
88 }
89 }
90 }
91 }
92 }
93 }
```

Screenshots of Running Application:

The screenshot shows the Eclipse IDE with a Java application running. The console output displays the results of 15 rounds of a card game. Each round shows the cards for Joseph and Julianna, and the winner of the round. The game ends in a draw after 15 rounds.

```
Round 1:
Joseph's card - Ten of Diamonds
Julianna's card - Two of Diamonds
Joseph wins the round!

Round 2:
Joseph's card - Jack of Clubs
Julianna's card - Five of Clubs
Joseph wins the round!

Round 3:
Joseph's card - One of Spades
Julianna's card - King of Clubs
Julianna wins the round!

Round 4:
Joseph's card - Jack of Hearts
Julianna's card - Four of Diamonds
Joseph wins the round!

Round 5:
Joseph's card - Ten of Spades
Julianna's card - Five of Diamonds
Joseph wins the round!

Round 6:
Joseph's card - Seven of Spades
Julianna's card - Five of Hearts
Joseph wins the round!

Round 7:
Joseph's card - Seven of Clubs
Julianna's card - Four of Hearts
Joseph wins the round!

Round 8:
Joseph's card - King of Spades
Julianna's card - Four of Clubs
Joseph wins the round!

Round 9:
Joseph's card - Three of Diamonds
Julianna's card - Three of Hearts
DRAW! No score changes!

Round 10:
Joseph's card - Seven of Hearts
Julianna's card - Nine of Spades
Julianna wins the round!

Round 11:
Joseph's card - Seven of Diamonds
Julianna's card - Queen of Diamonds
Julianna wins the round!

Round 12:
Joseph's card - Two of Spades
Julianna's card - King of Hearts
Julianna wins the round!

Round 13:
Joseph's card - King of Diamonds
Julianna's card - Eight of Diamonds
Joseph wins the round!

Round 14:
Joseph's card - Jack of Spades
Julianna's card - Ace of Clubs
Julianna wins the round!

Round 15:
```

```
Problems | JavaDoc | Declaration | Console X
C:\mininetes> App [1] [Java Application] [Volumes\Eclipse\Eclipse.app\Contents\Eclipse\plugins\org.eclipse.jdt.ui\org.eclipse.jdt.ui.full.macosx.x86_64_17.0.1\20220015-1416\run\bin\java (Aug 5, 2022, 11:30:00 PM - 11:30:00 PM) [pid: 1361]
Julianna's card - Eight of Diamonds
Joseph wins the round!

Round 14:
Joseph's card - Jack of Spades
Julianna's card - Ace of Clubs
Julianna wins the round!

Round 15:
Joseph's card - Eight of Hearts
Julianna's card - Queen of Hearts
Julianna wins the round!

Round 16:
Joseph's card - Six of Hearts
Julianna's card - Ace of Spades
Julianna wins the round!

Round 17:
Joseph's card - Three of Spades
Julianna's card - Ten of Clubs
Julianna wins the round!

Round 18:
Joseph's card - Four of Spades
Julianna's card - Queen of Spades
Julianna wins the round!

Round 19:
Joseph's card - Nine of Clubs
Julianna's card - One of Clubs
Joseph wins the round!

Round 20:
Joseph's card - One of Hearts
Julianna's card - Six of Spades
Julianna wins the round!

Round 21:
Joseph's card - Eight of Clubs
Julianna's card - Nine of Hearts
Julianna wins the round!

Round 22:
Joseph's card - Eight of Spades
Julianna's card - Two of Clubs
Joseph wins the round!

Round 23:
Joseph's card - Five of Spades
Julianna's card - One of Diamonds
Joseph wins the round!

Round 24:
Joseph's card - Ace of Diamonds
Julianna's card - Six of Diamonds
Joseph wins the round!

Round 25:
Joseph's card - Queen of Clubs
Julianna's card - Jack of Diamonds
Joseph wins the round!

Round 26:
Joseph's card - Two of Hearts
Julianna's card - Six of Clubs
Julianna wins the round!

Results: Joseph - 13 || Julianna - 12
Joseph has won the game!!
```

URL to GitHub Repository:

<https://github.com/jfalzini99/java-war-card-game>