

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*.

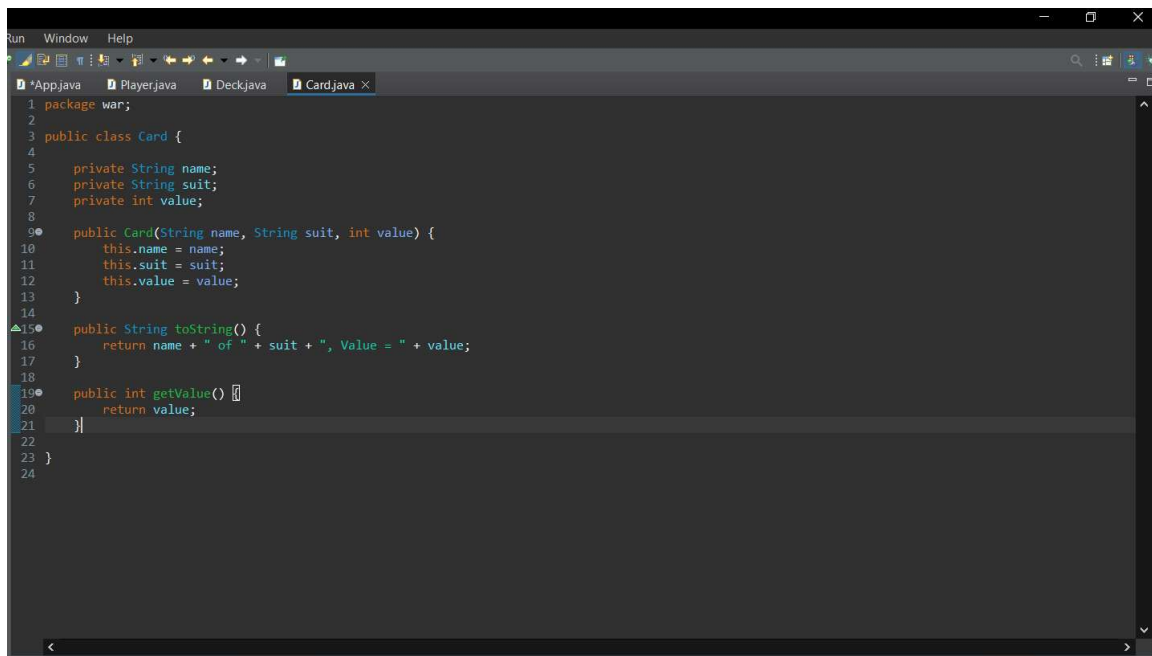
- Create the following classes.
 - Card
 - Fields
 - **value** (contains a value from 2-14 representing cards 2-Ace)
 - **name** (e.g. Ace of Diamonds, or Two of Hearts)
 - Methods
 - Getters and Setters

- **describe** (prints out information about a card)
- Deck
 - Fields
 - **cards** (List of Card)
 - Methods
 - **shuffle** (randomizes the order of the cards)
 - **draw** (removes and returns the top card of the Cards field)
 - In the constructor, when a new Deck is instantiated, the Cards field should be populated with the standard 52 cards.
- Player
 - Fields
 - **hand** (List of Card)
 - **score** (set to 0 in the constructor)
 - **name**
 - Methods
 - **describe** (prints out information about the player and calls the describe method for each card in the Hand List)
 - **flip** (removes and returns the top card of the Hand)
 - **draw** (takes a Deck as an argument and calls the draw method on the deck, adding the returned Card to the hand field)
 - **incrementScore** (adds 1 to the Player's score field)
- Create a class called App with a main method.
- Instantiate a Deck and two Players, call the shuffle method on the deck.
- Using a traditional for loop, iterate 52 times calling the Draw method on the other player each iteration using the Deck you instantiated.
- Using a traditional for loop, iterate 26 times and call the flip method for each player.
 - 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.

- After the loop, compare the final score from each player.
- 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 public class Card {
4
5     private String name;
6     private String suit;
7     private int value;
8
9     public Card(String name, String suit, int value) {
10         this.name = name;
11         this.suit = suit;
12         this.value = value;
13     }
14
15     public String toString() {
16         return name + " of " + suit + ", Value = " + value;
17     }
18
19     public int getValue() {
20         return value;
21     }
22 }
23
24
```

```
Run Window Help
App.java Player.java Deck.java Card.java
1 package war;
2
3 import java.util.LinkedList;
4
5
6
7 public class Deck {
8
9     List<String> cardNames = List.of("Two", "Three", "Four", "Five", "Six", "Seven", "Eight", "Nine", "Ten", "Jack",
10         "Queen", "King", "Ace");
11     List<String> suits = List.of("Diamonds", "Clubs", "Hearts", "Spades");
12     List<Card> cards = new LinkedList<>();
13
14     public Deck() {
15         for (int i = 0; i < cardNames.size(); i++) {
16             String cardName = cardNames.get(i);
17             int value = i + 2;
18
19             for (String suit : suits) {
20                 cards.add(new Card(cardName, suit, value));
21             }
22         }
23     }
24
25     public String toString() {
26         StringBuilder b = new StringBuilder();
27         b.append("\nCards in deck:\n");
28         for (Card card : cards) {
29             b.append(card).append("\n");
30         }
31         return b.toString();
32     }
33
34     public List<Card> getCards() {
35         return cards;
36     }
37 }
```

```
Run Window Help
App.java Player.java Deck.java Card.java
22
23 }
24
25 public String toString() {
26     StringBuilder b = new StringBuilder();
27     b.append("\nCards in deck:\n");
28     for (Card card : cards) {
29         b.append(card).append("\n");
30     }
31     return b.toString();
32 }
33
34 public List<Card> getCards() {
35     return cards;
36 }
37
38 public void shuffle() {
39     Random random = new Random();
40     List<Card> tempCards = new LinkedList<>(cards);
41
42     cards.clear();
43
44     while (!tempCards.isEmpty()) {
45         int pos = random.nextInt(tempCards.size());
46         cards.add(tempCards.remove(pos));
47     }
48 }
49
50 public int size() {
51     return cards.size();
52 }
53
54 }
55 }
```

```
ect Run Window Help
App.java Player.java Deck.java Card.java
1 package war;
2
3 import java.util.LinkedList;
4
5
6 public class Player {
7
8     private String name;
9     private List<Card> hand = new LinkedList<>();
10    private int score;
11
12    public Player(String name) {
13        this.name = name;
14    }
15
16    public String toString() {
17        return name;
18    }
19
20    public void draw(Deck deck) {
21        hand.add(deck.getCards().remove(0));
22    }
23
24    public List<Card> getHand() {
25        return hand;
26    }
27
28    public Card flip() {
29        return hand.remove(0);
30    }
31
32    public void incrementScore() {
33        score += 1;
34    }
35
36 }
```

```
Run Window Help
App.java Player.java Deck.java Card.java
8     private String name;
9     private List<Card> hand = new LinkedList<>();
10    private int score;
11
12    public Player(String name) {
13        this.name = name;
14    }
15
16    public String toString() {
17        return name;
18    }
19
20    public void draw(Deck deck) {
21        hand.add(deck.getCards().remove(0));
22    }
23
24    public List<Card> getHand() {
25        return hand;
26    }
27
28    public Card flip() {
29        return hand.remove(0);
30    }
31
32    public void incrementScore() {
33        score += 1;
34    }
35
36    public int getScore() {
37        return score;
38    }
39
40 }
41
```

```

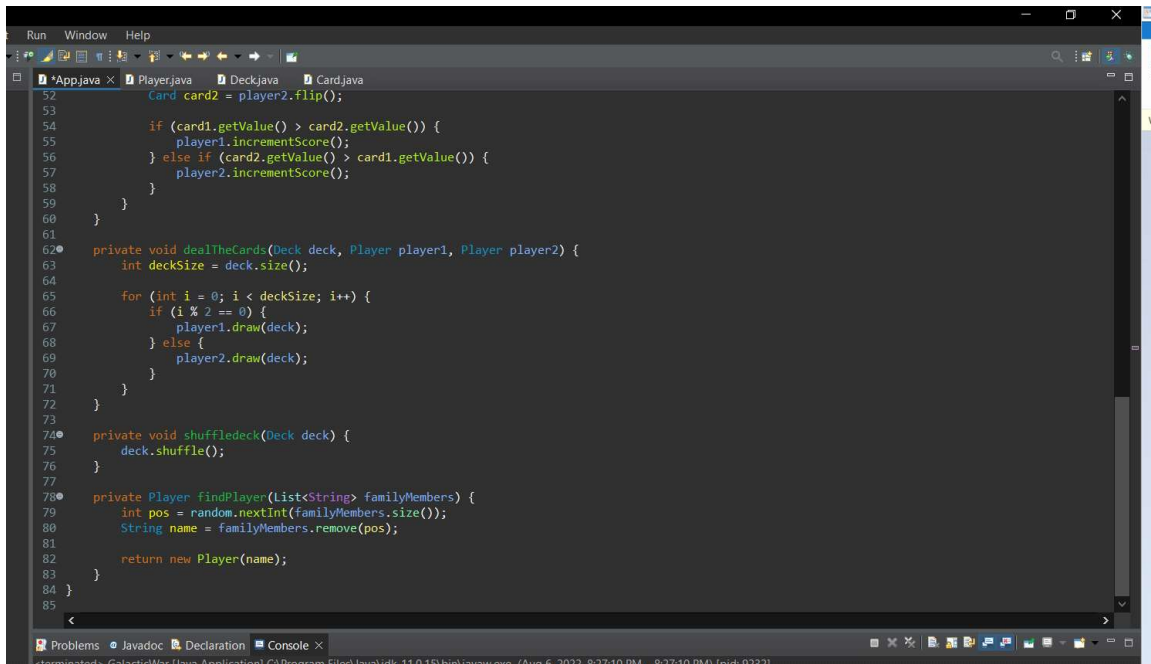
1 package war;
2
3 import java.util.LinkedList;
4
5
6
7 public class App {
8     List<String> familyMembers = List.of("Nick", "Caroline", "Eva", "Loriann", "Ed", "Ann", "Steve", "Doug");
9     Random random = new Random();
10
11     public static void main(String[] args) {
12
13         new App().playGame();
14     }
15
16     private void playGame() {
17         Deck deck = new Deck();
18         List<String> familyNames = new LinkedList<>(familyMembers);
19
20         Player player1 = findPlayer(familyNames);
21         Player player2 = findPlayer(familyNames);
22
23         System.out.println(player1 + " and " + player2 + " are playing War.");
24
25         shuffleDeck(deck);
26
27         dealTheCards(deck, player1, player2);
28
29         playWar(player1, player2);
30         findTheWinner(player1, player2);
31     }
32
33     private void findTheWinner(Player player1, Player player2) {
34         if (player1.getScore() > player2.getScore()) {
35             System.out.println("Player 1: " + player1 + " is the winner with a score of " + player1.getScore());
36         }
37     }
38 }

```

```

33
34     private void findTheWinner(Player player1, Player player2) {
35         if (player1.getScore() > player2.getScore()) {
36             System.out.println("Player 1: " + player1 + " is the winner with a score of " + player1.getScore());
37             System.out.println("Player 2: " + player2 + " is the loser with a score of " + player2.getScore());
38         } else if (player2.getScore() > player1.getScore()) {
39             System.out.println("Player 2: " + player2 + " is the winner with a score of " + player2.getScore());
40             System.out.println("Player 1: " + player1 + " is the loser with a score of " + player1.getScore());
41         } else {
42             System.out.println("There was a Draw, both " + player1 + " and " + player2 + " have a score of " + player1.getScore());
43         }
44     }
45
46     private void playWar(Player player1, Player player2) {
47         int numCards = player1.getHand().size();
48
49         for (int turn = 0; turn < numCards; turn++) {
50             Card card1 = player1.flip();
51             Card card2 = player2.flip();
52
53             if (card1.getValue() > card2.getValue()) {
54                 player1.incrementScore();
55             } else if (card2.getValue() > card1.getValue()) {
56                 player2.incrementScore();
57             }
58         }
59     }
60
61     private void dealTheCards(Deck deck, Player player1, Player player2) {
62         int deckSize = deck.size();
63
64         for (int i = 0; i < deckSize; i++) {
65             if (i % 2 == 0) {
66

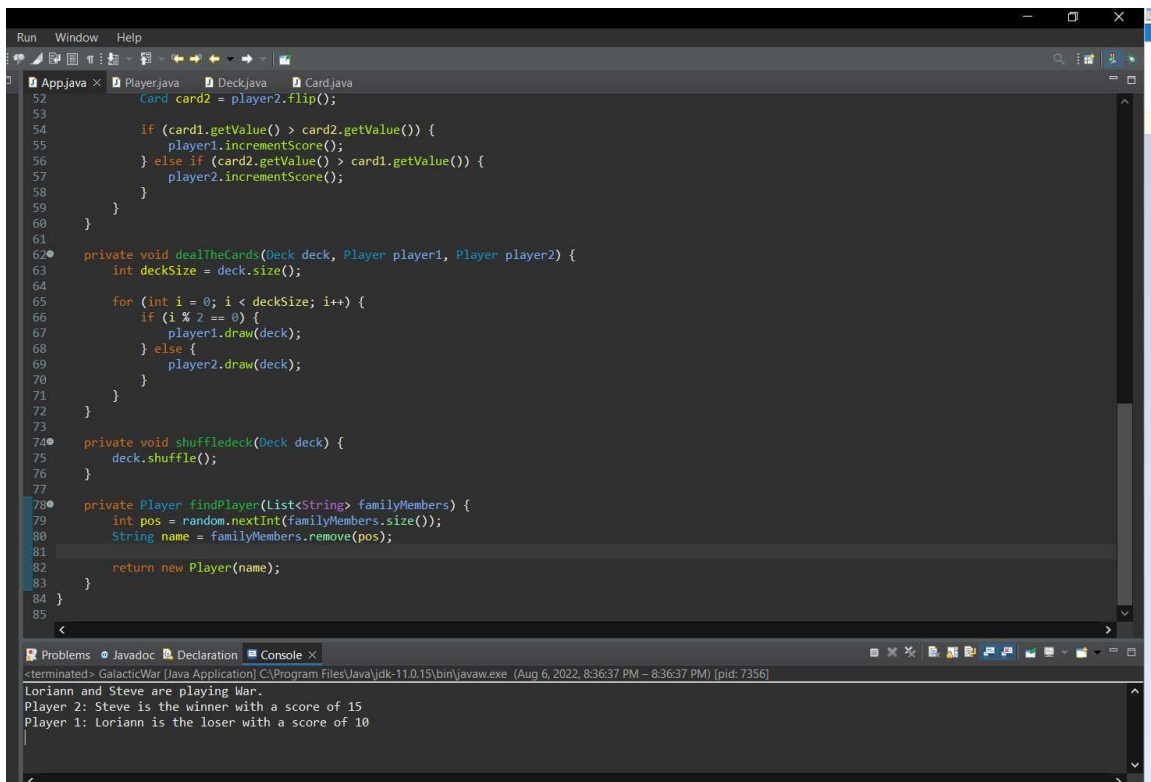
```



This screenshot shows an IDE window with a Java file named 'App.java'. The code implements a card game logic. It includes methods for flipping a card, comparing card values to determine the winner, dealing cards to two players in an alternating fashion, shuffling a deck, and finding a player by name from a list. The code is as follows:

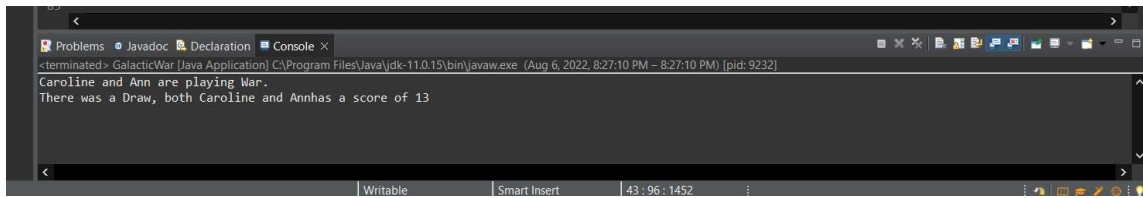
```
52 Card card2 = player2.flip();
53
54 if (card1.getValue() > card2.getValue()) {
55     player1.incrementScore();
56 } else if (card2.getValue() > card1.getValue()) {
57     player2.incrementScore();
58 }
59 }
60
61
62 private void dealTheCards(Deck deck, Player player1, Player player2) {
63     int deckSize = deck.size();
64
65     for (int i = 0; i < deckSize; i++) {
66         if (i % 2 == 0) {
67             player1.draw(deck);
68         } else {
69             player2.draw(deck);
70         }
71     }
72 }
73
74 private void shuffleDeck(Deck deck) {
75     deck.shuffle();
76 }
77
78 private Player findPlayer(List<String> familyMembers) {
79     int pos = random.nextInt(familyMembers.size());
80     String name = familyMembers.remove(pos);
81
82     return new Player(name);
83 }
84 }
85
```

Screenshots of Running Application:



This screenshot shows the same IDE window as above, but with the 'Console' tab selected at the bottom. The console displays the output of the application, which is a text-based card game simulation. The output shows the game progress, including card deals and the final scores for the two players, Loriann and Steve.

```
<terminated> GalacticWar [Java Application] C:\Program Files\Java\jdk-11.0.15\bin\javaw.exe (Aug 6, 2022, 8:36:37 PM - 8:36:37 PM) [pid: 7356]
Loriann and Steve are playing War.
Player 2: Steve is the winner with a score of 15
Player 1: Loriann is the loser with a score of 10
|
```



URL to GitHub Repository:

<https://github.com/NckDeSimone/Week-6-Final-Project>