

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.

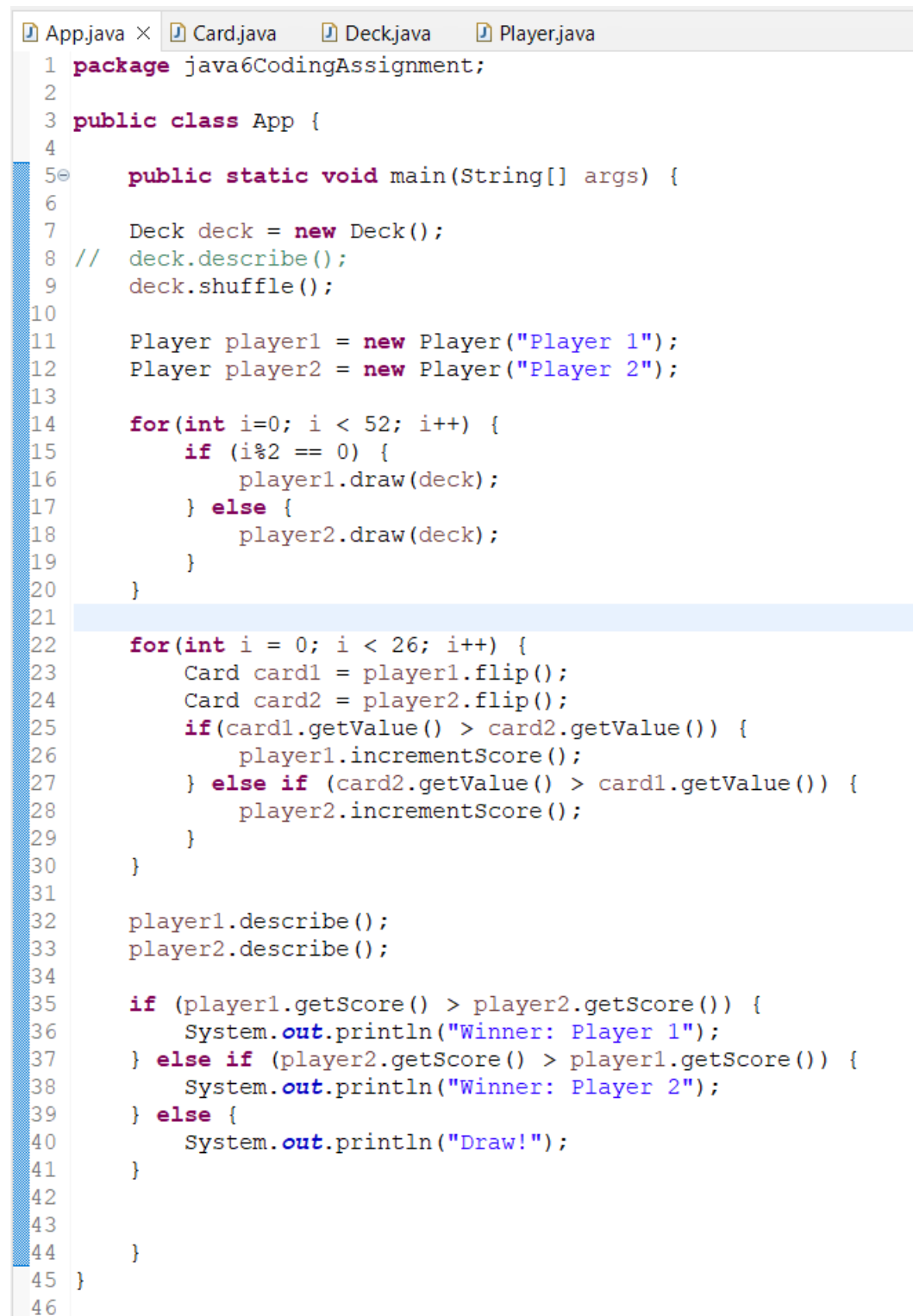
Coding Steps:

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:



```
App.java × Card.java Deck.java Player.java
1 package java6CodingAssignment;
2
3 public class App {
4
5     public static void main(String[] args) {
6
7         Deck deck = new Deck();
8         // deck.describe();
9         deck.shuffle();
10
11         Player player1 = new Player("Player 1");
12         Player player2 = new Player("Player 2");
13
14         for(int i=0; i < 52; i++) {
15             if (i%2 == 0) {
16                 player1.draw(deck);
17             } else {
18                 player2.draw(deck);
19             }
20         }
21
22         for(int i = 0; i < 26; i++) {
23             Card card1 = player1.flip();
24             Card card2 = player2.flip();
25             if(card1.getValue() > card2.getValue()) {
26                 player1.incrementScore();
27             } else if (card2.getValue() > card1.getValue()) {
28                 player2.incrementScore();
29             }
30         }
31
32         player1.describe();
33         player2.describe();
34
35         if (player1.getScore() > player2.getScore()) {
36             System.out.println("Winner: Player 1");
37         } else if (player2.getScore() > player1.getScore()) {
38             System.out.println("Winner: Player 2");
39         } else {
40             System.out.println("Draw!");
41         }
42
43     }
44 }
45
46
```

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

```
1 package java6CodingAssignment;
2
3 public class Card {
4
5     private int value;
6
7     private String name;
8
9     void describe() {
10         System.out.println(value + ", " + name);
11     }
12
13     public int getValue() {
14         return value;
15     }
16
17
18
19     public void setValue(int value) {
20         this.value = value;
21     }
22
23
24
25     public String getName() {
26         return name;
27     }
28
29
30
31     public void setName(String name) {
32         this.name = name;
33     }
34
35     //Card constructor???
36     public Card(int value, String name) {
37         this.value = value;
38         this.name = name;
39     }
40 }
41
```

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

```
1 package java6CodingAssignment;
2
3 import java.util.ArrayList;
4
5
6
7 public class Deck {
8
9     public List<Card> cards = new ArrayList<Card>();
10
11     public Deck() {
12
13         for(int i =2; i <=14; i++) {
14             String cardFullName = i + " of Hearts";
15             if(i == 11) {
16                 cardFullName = "Jack of Hearts";
17             } else if (i == 12){
18                 cardFullName = "Queen of Hearts";
19             } else if (i == 13){
20                 cardFullName = "King of Hearts";
21             } else if (i == 14){
22                 cardFullName = "Ace of Hearts";
23             }
24             Card card = new Card(i, cardFullName);
25             cards.add(card);
26         }
27         for(int i =2; i <=14; i++) {
28             String cardFullName = i + " of Diamonds";
29             if(i == 11) {
30                 cardFullName = "Jack of Diamonds";
31             } else if (i == 12){
32                 cardFullName = "Queen of Diamonds";
33             } else if (i == 13){
34                 cardFullName = "King of Diamonds";
35             } else if (i == 14){
36                 cardFullName = "Ace of Diamonds";
37             }
38             Card card = new Card(i, cardFullName);
39             cards.add(card);
40         }
41     }
42 }
```

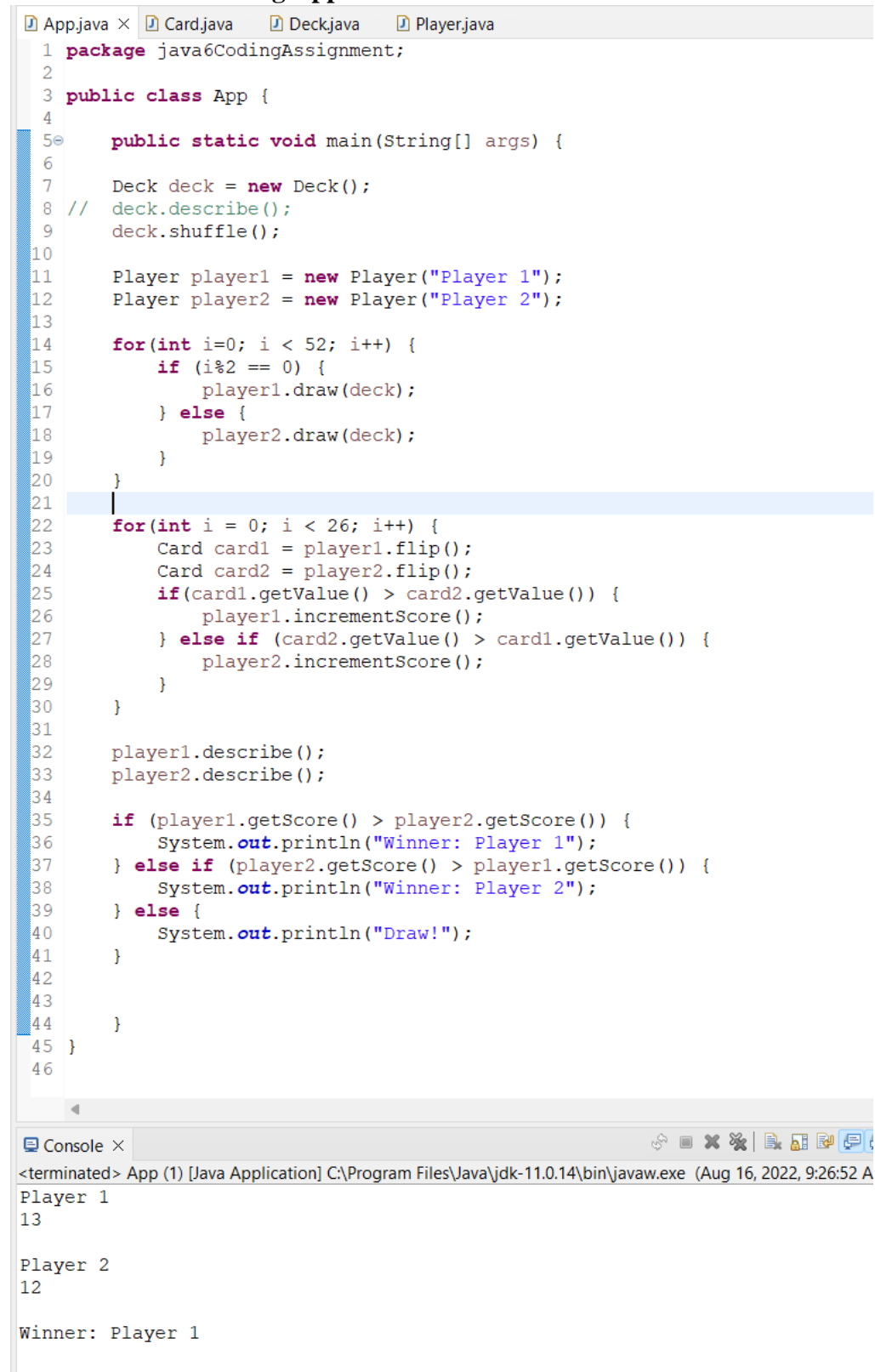
```

App.java  Card.java  Deck.java ×  Player.java
41         for(int i =2; i <=14; i++) {
42             String cardFullName = i + " of Spades";
43             if(i == 11) {
44                 cardFullName = "Jack of Spades";
45             } else if (i == 12){
46                 cardFullName = "Queen of Spades";
47             } else if (i == 13){
48                 cardFullName = "King of Spades";
49             } else if (i == 14){
50                 cardFullName = "Ace of Spades";
51             }
52             Card card = new Card(i, cardFullName);
53             cards.add(card);
54         }
55         for(int i =2; i <=14; i++) {
56             String cardFullName = i + " of Clubs";
57             if(i == 11) {
58                 cardFullName = "Jack of Clubs";
59             } else if (i == 12){
60                 cardFullName = "Queen of Clubs";
61             } else if (i == 13){
62                 cardFullName = "King of Clubs";
63             } else if (i == 14){
64                 cardFullName = "Ace of Clubs";
65             }
66             Card card = new Card(i, cardFullName);
67             cards.add(card);
68         }
69     }
70 }
71
72
73 public void shuffle() {
74     Collections.shuffle(cards);
75 }
76
77 public Card draw() {
78     Card drawnCard = cards.get(0);
79     cards.remove(0);
80     return drawnCard;
81 }
82
83 public void describe() {
84     for (Card card:cards) {
85         card.describe();
86     }
87 }
88 }
89

```

```
App.java Card.java Deck.java Player.java ×
2
3 import java.util.ArrayList;
4
5
6 public class Player {
7
8     List<Card> hand = new ArrayList<Card>();
9
10    private String playerName;
11
12    private int score;
13
14    public Player(String name) {
15        this.playerName = name;
16        this.score = 0;
17    }
18
19    public void describe() {
20        System.out.println(this.getPlayerName());
21        System.out.println(this.getScore() + "\n");
22        for (Card card:hand) {
23            card.describe();
24        }
25    }
26
27    public void draw(Deck deck) {
28        hand.add(deck.draw());
29    }
30
31    public Card flip() {
32        return hand.remove(0);
33    }
34
35    public void incrementScore() {
36        score = score + 1;
37    }
38
39    public String getPlayerName() {
40        return playerName;
41    }
42
43    public void setPlayerName(String playerName) {
44        this.playerName = playerName;
45    }
46
47    public int getScore() {
48        return score;
49    }
50
51    public void setScore(int score) {
52        this.score = score;
53    }
54
55    public List<Card> getHand() {
56        return hand;
57    }
58
59    public void setHand(List<Card> hand) {
60        this.hand = hand;
61    }
62
63 }
```

Screenshots of Running Application:



The screenshot displays a Java IDE with four tabs: App.java, Card.java, Deck.java, and Player.java. The App.java tab is active, showing the following code:

```
1 package java6CodingAssignment;
2
3 public class App {
4
5     public static void main(String[] args) {
6
7         Deck deck = new Deck();
8         // deck.describe();
9         deck.shuffle();
10
11         Player player1 = new Player("Player 1");
12         Player player2 = new Player("Player 2");
13
14         for(int i=0; i < 52; i++) {
15             if (i%2 == 0) {
16                 player1.draw(deck);
17             } else {
18                 player2.draw(deck);
19             }
20         }
21
22         for(int i = 0; i < 26; i++) {
23             Card card1 = player1.flip();
24             Card card2 = player2.flip();
25             if(card1.getValue() > card2.getValue()) {
26                 player1.incrementScore();
27             } else if (card2.getValue() > card1.getValue()) {
28                 player2.incrementScore();
29             }
30         }
31
32         player1.describe();
33         player2.describe();
34
35         if (player1.getScore() > player2.getScore()) {
36             System.out.println("Winner: Player 1");
37         } else if (player2.getScore() > player1.getScore()) {
38             System.out.println("Winner: Player 2");
39         } else {
40             System.out.println("Draw!");
41         }
42
43
44     }
45 }
46
```

The console window at the bottom shows the output of the application:

```
<terminated> App (1) [Java Application] C:\Program Files\Java\jdk-11.0.14\bin\javaw.exe (Aug 16, 2022, 9:26:52 A
Player 1
13

Player 2
12

Winner: Player 1
```


URL to GitHub Repository:

<https://github.com/CoconutMacaron/JavaWarGameProject.git>