

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 week6CodingAssignment;
2
3 //create a class App with a main method
4
5 public class App {
6
7     public static void main(String[] args) {
8
9         //3- instantiate a deck and two players, call the shuffle method on the deck
10
11         Deck deck = new Deck();
12         deck.shuffle();
13
14         Player player1 = new Player("Nova");
15         Player player2 = new Player("Silas");
16
17         //4- using a traditional for loop, iterate 52 times calling the draw method on the other player using the deck instantiated
18
19         for (int draw = 0; draw < 26; draw++) {
20             player1.draw(deck);
21             player2.draw(deck);
22         }
23
24
25         //5- using a traditional for loop, iterate 26 times and call the flip method for each player
26         //compare the value of each card returned and call the incrementScore method on the player with the higher card value
27
28         for (int i = 0; i < 26; i++) {
29             Card player1FlipCard = player1.flip();
30             Card player2FlipCard = player2.flip();
31
32             System.out.println(player1.getName() + " drew " + player1FlipCard.getName());
33             System.out.println(player2.getName() + " drew " + player2FlipCard.getName());
34
35             if (player1FlipCard.getValue() > player2FlipCard.getValue()) {
36                 player1.incrementScore();
37             } else if (player2FlipCard.getValue() > player1FlipCard.getValue()) {
38                 player2.incrementScore();
39             }
40         }
41
42         //6- after the loop, compare the final score from each player
43
44         System.out.println(player1.getName() + " scored: " + player1.getScore());
45     }
46 }
```

```
App.java × Card.java Deck.java Player.java
16
17 //4- using a traditional for loop, iterate 52 times calling the draw method on the other player using the deck instantiated
18
19 for (int draw = 0; draw < 26; draw++) {
20     player1.draw(deck);
21     player2.draw(deck);
22 }
23
24
25 //5- using a traditional for loop, iterate 26 times and call the flip method for each player
26 //compare the value of each card returned and call the incrementScore method on the player with the higher card value
27
28 for (int i = 0; i < 26; i++) {
29     Card player1FlipCard = player1.flip();
30     Card player2FlipCard = player2.flip();
31
32     System.out.println(player1.getName() + " drew " + player1FlipCard.getName());
33     System.out.println(player2.getName() + " drew " + player2FlipCard.getName());
34
35     if (player1FlipCard.getValue() > player2FlipCard.getValue()) {
36         player1.incrementScore();
37     } else if (player2FlipCard.getValue() > player1FlipCard.getValue()) {
38         player2.incrementScore();
39     }
40 }
41
42 //6- after the loop, compare the final score from each player
43
44 System.out.println(player1.getName() + " scored: " + player1.getScore());
45 System.out.println(player2.getName() + " scored: " + player2.getScore());
46
47 //7- print the final score of each player and either player1, player2, or draw depending on which score is higher or if they are the same
48
49 if (player1.getScore() > player2.getScore()) {
50     System.out.println(player1.getName() + " is the winner!");
51 } else if (player2.getScore() > player1.getScore()) {
52     System.out.println(player2.getName() + " is the winner!");
53 } else {
54     System.out.println("It is a draw!");
55 }
56
57 }
58
59 }
60 }
```

```
App.java Card.java X Deck.java Player.java
1 package week6CodingAssignment;
2
3 //a- Create class Card
4
5 public class Card {
6
7     //fields of value and name
8
9     private int value;
10    private String name;
11
12    //Constructor
13
14    public Card(int value, String name) {
15        this.value = value;
16        this.name = name;
17    }
18
19    //getters and setters
20
21    public int getValue() {
22        return value;
23    }
24
25    public void setValue(int value) {
26        this.value = value;
27    }
28
29    public String getName() {
30        return name;
31    }
32
33    public void setName(String name) {
34        this.name = name;
35    }
36
37    //describe method for a card
38    public String describe() {
39        return name;
40    }
41 }
42
```

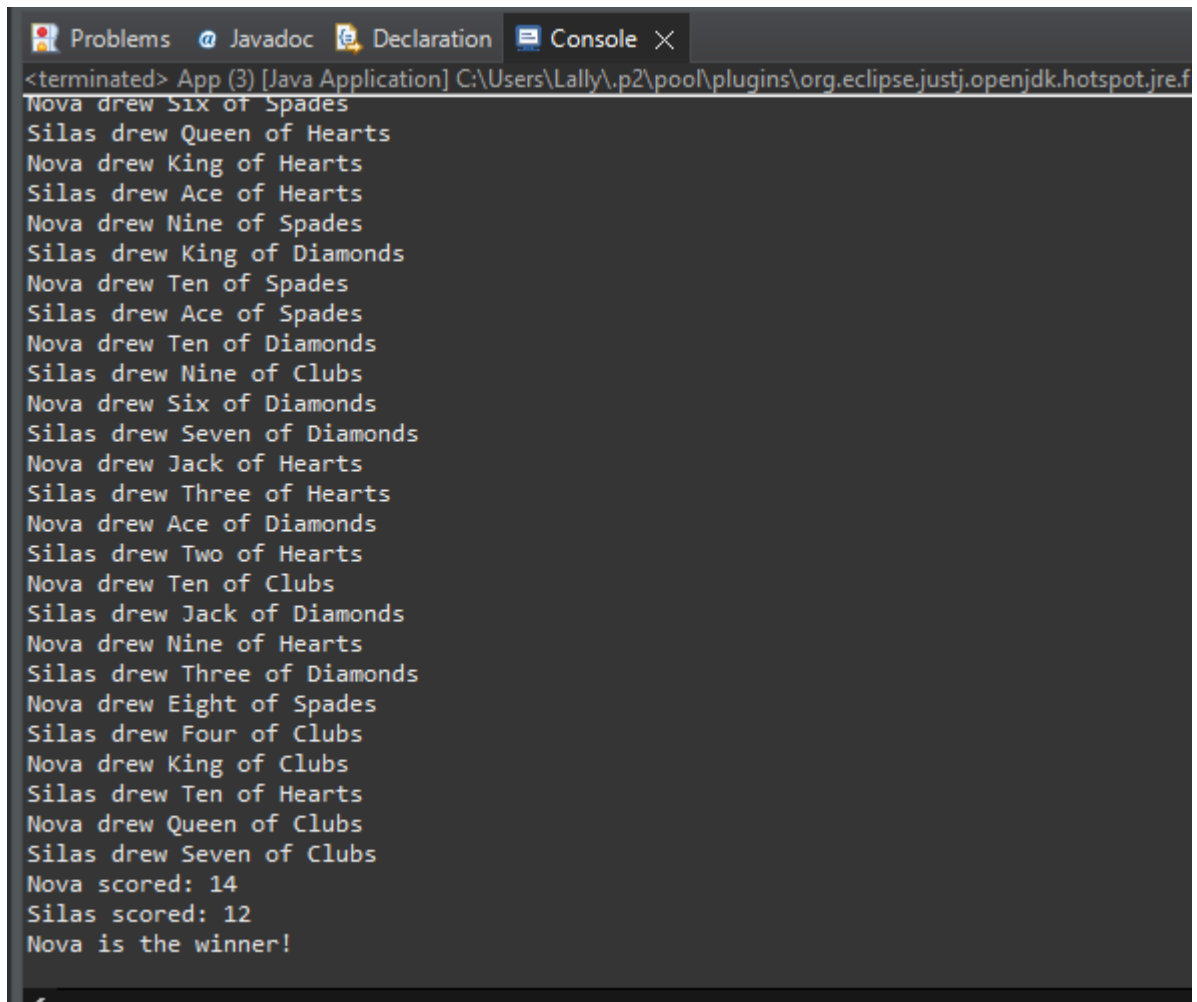
```
App.java Card.java Deck.java X Player.java
1 package week6CodingAssignment;
2
3 import java.util.ArrayList;
4
5 //b- Create class Deck
6
7 public class Deck {
8
9     //field cards(list of cards)
10
11     List<Card> cards = new ArrayList<Card>();
12
13     Deck() {
14         String[] suit = {"Diamonds", "Hearts", "Spades", "Clubs"};
15         String[] value = {"Two", "Three", "Four", "Five", "Six", "Seven", "Eight", "Nine", "Ten", "Jack", "Queen", "King", "Ace"};
16         for (int i = 0; i < 4; i++) {
17             for (int x = 2; x < 15; x++) {
18                 cards.add(new Card(x, value[x - 2] + " of " + suit[i]));
19             }
20         }
21
22         //shuffle(randomizes the order of the cards)
23
24         public void shuffle() {
25             Collections.shuffle(cards);
26         }
27
28         //draw(removes and returns the top card of the Cards field)
29
30         public Card draw() {
31             Card drawCard = cards.get(0);
32             cards.remove(0);
33             return drawCard;
34         }
35     }
36 }
37
38
39
40
41
42
```

```
App.java Card.java Deck.java Player.java X
1 package week6CodingAssignment;
2
3 import java.util.ArrayList;
4
5 //c- Create class Player
6
7
8 public class Player {
9
10     //fields of hand(list of Card), score(set to 0 in constructor), and name
11
12     private List<Card> hand = new ArrayList<Card>();
13     private String name;
14     private int score;
15
16     //constructor
17
18     public Player(String name) {
19         this.name = name;
20         score = 0;
21     }
22
23
24     //describe(prints out info about the player and calls describe method for cards in hand list)
25     public void describe() {
26         System.out.println(name + " : " + hand);
27         for (int i = 0; i < hand.size(); i++) {
28             hand.get(i).describe();
29         }
30     }
31
32     //flip(removes and returns the top card of the hand)
33
34     public Card flip() {
35         Card drawCard = hand.get(0);
36         hand.remove(0);
37         return drawCard;
38     }
39
40     //draw(takes a deck and calls the draw method on the deck, adding returned card to the hand field)
41
42     public void draw(Deck deck) {
43         hand.add(deck.draw());
44     }
45
46     //increment score(adds one to the players score field)
```

```
App.java Card.java Deck.java Player.java X
28
29     }
30 }
31
32 //flip(removes and returns the top card of the hand)
33
34 public Card flip() {
35     Card drawCard = hand.get(0);
36     hand.remove(0);
37     return drawCard;
38 }
39
40 //draw(takes a deck and calls the draw method on the deck, adding returned card to the hand field)
41
42 public void draw(Deck deck) {
43     hand.add(deck.draw());
44 }
45
46 //increment score(adds one to the players score field)
47
48 public void incrementScore() {
49     setScore(getScore() + 1);
50 }
51
52 //getters and setters
53
54 public int getScore() {
55     return score;
56 }
57
58 public void setScore(int score) {
59     this.score = score;
60 }
61
62 public String getName() {
63     return name;
64 }
65
66 public void setName(String name) {
67     this.name = name;
68 }
69
70 }
71
72 }
```

Screenshots of Running Application:

```
Problems Javadoc Declaration Console X
<terminated> App (3) [Java Application] C:\Users\Lally\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre
Nova drew Seven of Hearts
Silas drew Five of Diamonds
Nova drew Four of Spades
Silas drew Eight of Clubs
Nova drew Three of Spades
Silas drew Five of Spades
Nova drew Five of Clubs
Silas drew Ace of Clubs
Nova drew Queen of Spades
Silas drew Eight of Hearts
Nova drew Eight of Diamonds
Silas drew Six of Hearts
Nova drew Five of Hearts
Silas drew Two of Spades
Nova drew Nine of Diamonds
Silas drew King of Spades
Nova drew Seven of Spades
Silas drew Three of Clubs
Nova drew Four of Diamonds
Silas drew Six of Clubs
Nova drew Four of Hearts
Silas drew Jack of Clubs
Nova drew Jack of Spades
Silas drew Two of Diamonds
Nova drew Queen of Diamonds
Silas drew Two of Clubs
Nova drew Six of Spades
Silas drew Queen of Hearts
Nova drew King of Hearts
Silas drew Ace of Hearts
```



The screenshot shows the Eclipse IDE's console window. The title bar includes tabs for 'Problems', 'Javadoc', 'Declaration', and 'Console'. The console output displays the following text:

```
<terminated> App (3) [Java Application] C:\Users\Lally\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.f
Nova drew Six of Spades
Silas drew Queen of Hearts
Nova drew King of Hearts
Silas drew Ace of Hearts
Nova drew Nine of Spades
Silas drew King of Diamonds
Nova drew Ten of Spades
Silas drew Ace of Spades
Nova drew Ten of Diamonds
Silas drew Nine of Clubs
Nova drew Six of Diamonds
Silas drew Seven of Diamonds
Nova drew Jack of Hearts
Silas drew Three of Hearts
Nova drew Ace of Diamonds
Silas drew Two of Hearts
Nova drew Ten of Clubs
Silas drew Jack of Diamonds
Nova drew Nine of Hearts
Silas drew Three of Diamonds
Nova drew Eight of Spades
Silas drew Four of Clubs
Nova drew King of Clubs
Silas drew Ten of Hearts
Nova drew Queen of Clubs
Silas drew Seven of Clubs
Nova scored: 14
Silas scored: 12
Nova is the winner!
```

URL to GitHub Repository:

<https://github.com/aduran92/Week6-Coding-Assignment>