

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:

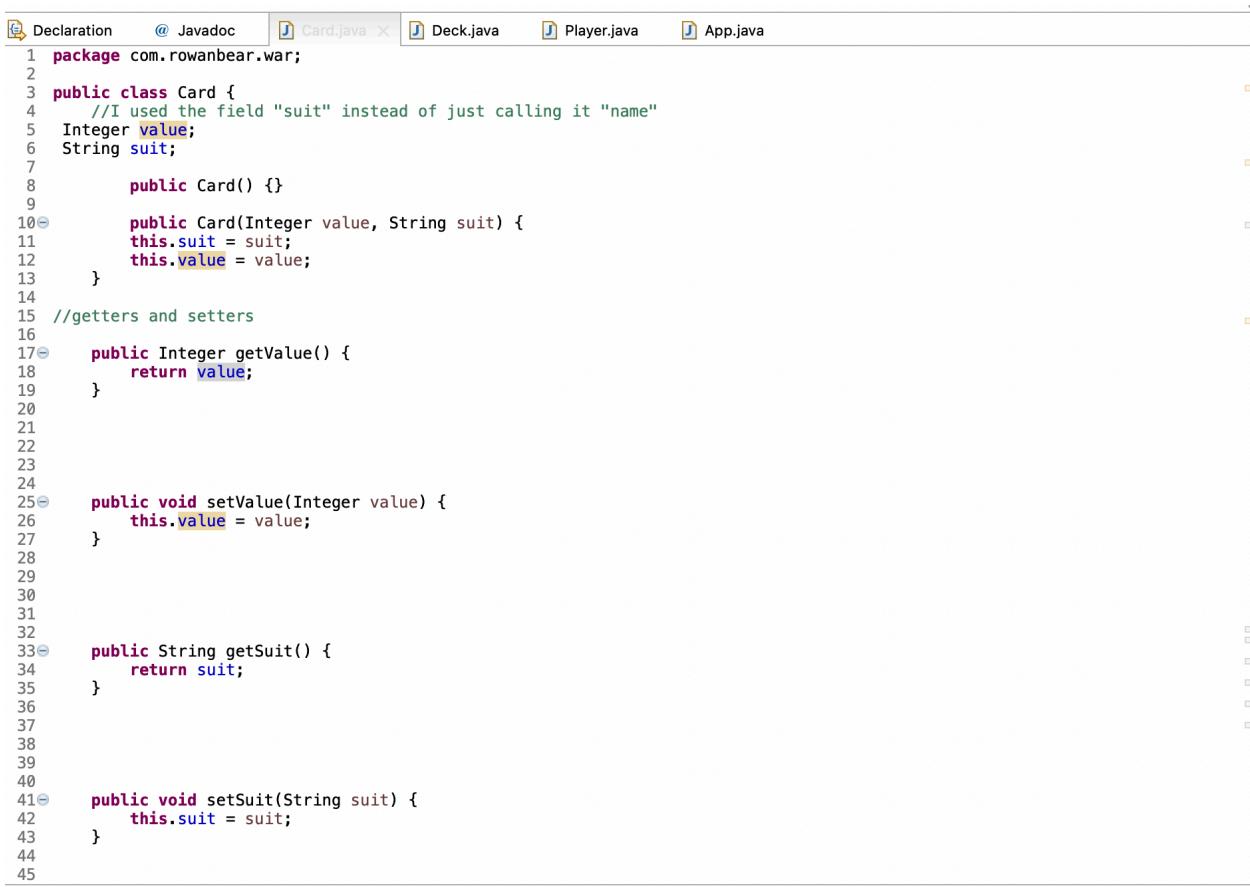
For the final project you will be creating an automated version of the classic card game *WAR*.

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:



The screenshot shows a Java code editor with the file `Card.java` open. The code defines a class `Card` with fields `value` (an `Integer`) and `suit` (a `String`). It includes a constructor that takes `value` and `suit`, and setter/getter methods for both fields. The code editor interface at the top shows tabs for Declaration, Javadoc, Card.java, Deck.java, Player.java, and App.java.

```

Declaration @ Javadoc Card.java X Deck.java Player.java App.java
1 package com.rowanbear.war;
2
3 public class Card {
4     //I used the field "suit" instead of just calling it "name"
5     Integer value;
6     String suit;
7
8     public Card() {}
9
10    public Card(Integer value, String suit) {
11        this.suit = suit;
12        this.value = value;
13    }
14
15    //getters and setters
16
17    public Integer getValue() {
18        return value;
19    }
20
21
22
23
24
25    public void setValue(Integer value) {
26        this.value = value;
27    }
28
29
30
31
32    public String getSuit() {
33        return suit;
34    }
35
36
37
38
39
40    public void setSuit(String suit) {
41        this.suit = suit;
42    }
43
44
45

```

```
35     }
36
37
38
39
40     public void setSuit(String suit) {
41         this.suit = suit;
42     }
43
44
45 //methods
46
47 /**
48 * describe (prints out information about a card)
49 */
50
51
52     public void describe() {
53
54         if (value < 11) {
55             System.out.println(value + suit);
56             // if the card is referred to with a name instead of a value
57         } else if (value == 11) {
58             System.out.println("Jack" + suit);
59         } else if (value == 12) {
60             System.out.println("Queen" + suit);
61         } else if (value == 13) {
62             System.out.println("King" + suit);
63         } else if (value == 14) {
64             System.out.println("Ace" + suit);
65         }
66
67
68
69
70
71
72
73
74     }
75
76 }
77
78
79
```

Console <terminated> app [Java Application] /Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java (Dec 4, 2021, 1:49:34 PM – 1:49:34 PM)
is played by Tom

```
35     }
36
37
38
39
40
41     public void setSuit(String suit) {
42         this.suit = suit;
43     }
44
45
46
47 //methods
48
49 /**
50 * describe (prints out information about a card)
51 */
52
53     public void describe() {
54
55         if (value < 11) {
56             System.out.println(value + suit);
57             // if the card is referred to with a name instead of a value
58         } else if (value == 11) {
59             System.out.println("Jack" + suit);
60         } else if (value == 12) {
61             System.out.println("Queen" + suit);
62         } else if (value == 13) {
63             System.out.println("King" + suit);
64         } else if (value == 14) {
65             System.out.println("Ace" + suit);
66         }
67
68
69
70
71
72
73
74     }
75
76 }
77
78
79
```

```
Declaration @ Javadoc Card.java Deck.java Player.java App.java
1 package com.rowanbear.war;
2
3 import java.util.ArrayList;
4
5 import java.util.Collections;
6
7 public class Deck {
8
9     ArrayList<Card> cards;
10
11    public Deck() {
12
13        this.cards = new ArrayList<Card>();
14
15
16        //creates ArrayList [2, ... 15]
17        ArrayList<Integer> cardValue = new ArrayList<Integer>();
18        for (int i = 2; i < 15; i++) {
19            cardValue.add(i);
20        }
21
22        //In the constructor, when a new Deck is instantiated, the Cards field should be populated with the
23        //standard 52 cards.
24
25        //to make hearts cards
26        for (Integer i : cardValue) {
27            Card card = new Card(i, " Of Hearts" );
28            cards.add(card);
29        }
30
31        //to make spade cards
32        for (Integer i : cardValue) {
33            Card card = new Card(i, " Of Spades" );
34            cards.add(card);
35        }
36
37        //to make diamond cards
38        for (Integer i : cardValue) {
39            Card card = new Card(i, " Of Diamonds" );
40            cards.add(card);
41        }
42
43        //to make clubs cards
44        for (Integer i : cardValue) {
```

```
Declaration @ Javadoc Card.java Deck.java Player.java App.java
38            for (Integer i : cardValue) {
39                Card card = new Card(i, " Of Diamonds" );
40                cards.add(card);
41            }
42
43            //to make clubs cards
44            for (Integer i : cardValue) {
45                Card card = new Card(i, " Of Clubs" );
46                cards.add(card);
47            }
48
49
50
51    }
52
53
54    //methods
55
56    /*
57     * shuffle (randomizes the order of the cards)
58     *
59     * draw (removes and returns the top card of the Cards field)
60     */
61
62
63    public void shuffle() {
64        Collections.shuffle(cards);
65    }
66
67
68    public Card draw() {
69
70        return cards.remove(0);
71    }
72
73
74    public void describe() {
75        for (Card card : cards) {
76            card.describe();
77        }
78    }
79
80
81 }
```

Declaration @ Javadoc Card.java Deck.java Player.java App.java

```
1 package com.rowanbear.war;
2
3 import java.util.ArrayList;
4
5 public class Player {
6
7     ArrayList<Card> hand;
8
9     int score;
10
11    String name;
12
13    public Player () {
14        this.hand = new ArrayList<Card>();
15        this.score = 0;
16        this.name = "Default Name";
17    }
18
19    //getters and setters
20
21    public ArrayList<Card> getHand() {
22        return hand;
23    }
24
25    public void setHand(ArrayList<Card> hand) {
26        this.hand = hand;
27    }
28
29    public int getScore() {
30        return score;
31    }
32
33    public void setScore(int score) {
34        this.score = score;
35    }
36
37    public String getName() {
38        return name;
39    }
40
41    public void setName(String name) {
42        this.name = name;
43    }
44
45    //methods
```

Declaration @ Javadoc Card.java Deck.java Player.java App.java

```
49 * describe - prints out info about player and calls describe method for each card in Hand List
50 *
51 * flip - removes and returns the top card of the Hand
52 *
53 * draw - takes a Deck as an argument and calls the draw method on the deck, adding the returned Card to the hand field.
54 *
55 * incrementScore - adds a point to player's score field
56 *
57 */
58
59
60    public void describe() {
61        System.out.println("This is " + this.name + ". \nThey have these cards in their hand: ");
62        for (Card card : this.hand) {
63            card.describe();
64        }
65
66        System.out.println(name + "'s current score is: " + score);
67
68    }
69
70    //flip method also prints out what is happening to the console to make it more exciting, like a real game!
71
72    public Card flip() {
73        Card card = hand.get(0);
74        hand.remove(hand.get(0));
75        System.out.println("-----");
76        card.describe();
77        System.out.println(" is played by " + this.name + "\n-----");
78
79        return card;
80    }
81
82
83    public void draw(Deck deck) {
84        this.hand.add(deck.draw());
85    }
86
87    public void incrementScore() {
88        this.score += 1;
89    }
90
91
92 }
```

Declaration @ Javadoc Card.java Deck.java Player.java App.java

```

1 package com.rowanbear.war;
2
3 public class App {
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7
8         //Instantiate a Deck and two Players, call the shuffle method on the deck.
9
10        Deck exampleDeck = new Deck();
11
12        exampleDeck.shuffle();
13
14        Player playerOne = new Player();
15        playerOne.name = "Lucy";
16        playerOne.describe();
17
18
19        Player playerTwo = new Player();
20        playerTwo.name = "Tom";
21        playerTwo.describe();
22
23        // Using a traditional for loop, iterate 52 times calling the Draw method on the other player each iteration using
24        // the Deck you instantiated.
24
25        for (int i = 2; i < 54; i++) {
26            if (i%2 == 0) {
27                playerOne.draw(exampleDeck);
28            } else {
29                playerTwo.draw(exampleDeck);
30            }
31        }
32
33        //Using a traditional for loop, iterate 26 times and call the flip method for each player.
34
35        //Compare the value of each card returned by the two player's flip methods. Call the incrementScore method on the
36        //player whose card has the higher value.
36
37
38        for (int i = 2; i < 28; i++) {
39
40            Card card1 = playerOne.flip();
41            Card card2 = playerTwo.flip();
43

```

Declaration @ Javadoc Card.java Deck.java Player.java App.java

```

35        //Compare the value of each card returned by the two player's flip methods. Call the incrementScore method on the
36        //player whose card has the higher value.
36
37
38        for (int i = 2; i < 28; i++) {
39
40            Card card1 = playerOne.flip();
41            Card card2 = playerTwo.flip();
43
44            if (card1.value > card2.value) {
45                System.out.println(playerOne.name + " wins this round");
46                playerOne.incrementScore();
47            } else if (card1.value < card2.value) {
48                System.out.println(playerTwo.name + " wins this round");
49                playerTwo.incrementScore();
50            } else if (card1.value == card2.value){
51                System.out.println("Draw: Neither player wins this round.");
52            }
53        }
54
55    }
56
57    //After the loop, compare the final score from each player.
58    //Print the final score of each player and either "Player 1", "Player 2", or "Draw" depending on which score is
59    //higher or if they are both the same.
59
60    System.out.println(playerOne.name + "'s score is: " + playerOne.score);
61
62    System.out.println(playerTwo.name + "'s score is: " + playerTwo.score);
63
64
65    if (playerOne.score > playerTwo.score) {
66        System.out.println(playerOne.name + " Wins");
67    } else if (playerOne.score < playerTwo.score) {
68        System.out.println(playerTwo.name + " Wins");
69    } else if (playerOne.score == playerTwo.score) {
70        System.out.println("Draw");
71    }
72
73
74
75
76
77

```

Screenshots of Running Application:

Console X |

<terminated> app [Java Application] /Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java (Dec 4, 2021, 1:51:21 PM – 1:51:21 PM)

This is Lucy.
 They have these cards in their hand:
 Lucy's current score is: 0
 This is Tom.
 They have these cards in their hand:
 Tom's current score is: 0

Queen Of Clubs
 is played by Lucy

2 Of Clubs
 is played by Tom

Lucy wins this round

Ace Of Diamonds
 is played by Lucy

4 Of Clubs
 is played by Tom

Lucy wins this round

7 Of Clubs
 is played by Lucy

5 Of Clubs
 is played by Tom

Lucy wins this round

4 Of Diamonds
 is played by Lucy

King Of Diamonds
 is played by Tom

Tom wins this round

King Of Spades

Console X |

<terminated> app [Java Application] /Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java (Dec 4, 2021, 1:51:21 PM – 1:51:21 PM)

6 Of Clubs
 is played by Tom

Lucy wins this round

2 Of Spades
 is played by Lucy

8 Of Clubs
 is played by Tom

Tom wins this round

Ace Of Hearts
 is played by Lucy

3 Of Clubs
 is played by Tom

Lucy wins this round

Jack Of Spades
 is played by Lucy

9 Of Diamonds
 is played by Tom

Lucy wins this round

10 Of Hearts
 is played by Lucy

9 Of Hearts
 is played by Tom

Lucy wins this round
 Lucy's score is: 13
 Tom's score is: 10
 Lucy Wins

URL to GitHub Repository: https://github.com/Rowan-Bear/Java-Week6_Final-Project.git