

# 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:

```

Deck.java  Card.java  Player.java  App.java
1 import java.util.HashMap;
2
3
4 public class Card {
5     private static StringBuilder builder = new StringBuilder(); // string builder shared amongst all card objects
6     private int value; // 2-14
7     private int name; // 0-3
8
9     // this map maps an integer to 1 out of 4 shapes and is constant (final)
10    // and is shared between all card objects
11    private static final Map<Integer, String> possibleShapes = new HashMap<>() {
12        /**
13         *
14         */
15        private static final long serialVersionUID = 1L; // eclipse suggested adding this but i didn't understand it
16
17        {
18            put((Integer) 0, "Diamond");
19            put((Integer) 1, "Heart");
20            put((Integer) 2, "Spade");
21            put((Integer) 3, "Club");
22        }
23    };
24
25    // this map maps an integer to the 13 values a card could have
26    // does not change (final) and is shared between all card objects
27    private static final Map<Integer, String> possibleValues = new HashMap<>() {
28        /**
29         *
30         */
31        private static final long serialVersionUID = 2393305926036422177L; // eclipse suggested adding this but don't
32        // know its function
33
34        {
35            put((Integer) 2, "Two");
36            put((Integer) 3, "Three");
37            put((Integer) 4, "Four");
38            put((Integer) 5, "Five");
39            put((Integer) 6, "Six");
40            put((Integer) 7, "Seven");
41            put((Integer) 8, "Eight");
42            put((Integer) 9, "Nine");
43            put((Integer) 10, "Ten");
44            put((Integer) 11, "Jack");
45            put((Integer) 12, "Queen");
46            put((Integer) 13, "King");
47            put((Integer) 14, "Ace");
48        }
49    };
50
51 }
52

```

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

```
52
53 // Default constructor with random values for name and value
54 Card() {
55     Random rand = new Random();
56     this.value = rand.nextInt(13) + 2;
57     this.name = rand.nextInt(4);
58 }
59
60 // Overloaded constructor with explicitly declared values for name and value
61 Card(int value, int name) {
62     this.value = value;
63     this.name = name;
64 }
65
66 // getters and setters
67 public int getValue() {
68     return value;
69 }
70
71 public void setValue(int value) {
72     this.value = value;
73 }
74
75 public int getName() {
76     return name;
77 }
78
79 public void setName(int name) {
80     this.name = name;
81 }
82
83 // size of the names map
84 public static int possibleNamesSize() {
85     return possibleShapes.size();
86 }
87
88 // size of the values map
89 public static int possibleValuesSize() {
90     return possibleValues.size();
91 }
```

```

92
93 // Describe returns a string of the verbally spoken full name of 1/52 cards
94 // uses the string builder to build the string and then clears it before
95 // returning it
96 public String describe() {
97     builder.append(possibleValues.get((Integer) value));
98     builder.append(" of ");
99     builder.append(possibleShapes.get((Integer) name));
100     builder.append("s");
101     String result = builder.toString();
102     builder.setLength(0);
103     return result;
104 }
105
106 // Same as describe. Used to automatically print out the
107 // spoken value of a card whenever this object needs to be converted to a string
108 public String toString() {
109     builder.append(possibleValues.get((Integer) value));
110     builder.append(" of ");
111     builder.append(possibleShapes.get((Integer) name));
112     builder.append("s");
113     String result = builder.toString();
114     builder.setLength(0);
115     return result;
116 }
117 }
118

```

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

```
1 import java.util.ArrayList;
2
3
4
5 public class Deck {
6     // not static, every deck is unique
7     private List<Card> cards = new ArrayList<>();
8
9     // constructor adds all possible cards in the deck
10    // designed for expansion of the type and value of the cards
11    // but may break due to the way 'value' works for a standard deck
12    Deck() {
13        for (int i = 0; i < Card.possibleNamesSize(); i++) {
14            for (int j = 0; j < Card.possibleValuesSize(); j++) {
15                // uses card's overloaded constructor
16                cards.add(new Card(j + 2, i));
17            }
18        }
19    }
20
21    // reorganizes the order of the cards in the deck
22    public void shuffle() {
23        // part of the standard Collections methods
24        Collections.shuffle(this.cards);
25    }
26
27    // pulls out the very last card in the list and returns it
28    public Card draw() {
29        Card selectedCard = this.cards.get(cards.size() - 1);
30        this.cards.remove(this.cards.size() - 1);
31        return selectedCard;
32    }
33
34    // calls toString method for a list (used the toString method in the Card object
35    public String toString() {
36        return this.cards.toString();
37    }
38
39    // returns an int for the current size of the deck
40    public int size() {
41        return this.cards.size();
42    }
43 }
44
```

Deck.java Card.java Player.java X App.java

```
1 import java.util.ArrayList;
2
3
4 public class Player {
5     // player's information
6     private List<Card> hand = new ArrayList<>();
7     private int score;
8     private String name;
9
10    // no default constructor, must name the player
11    Player(String name) {
12        this.name = name;
13        this.score = 0;
14    }
15
16    // returns int of the player's current score
17    public int getScore() {
18        return this.score;
19    }
20
21    // prints out information of the player to the console
22    // hand, name, score
23    public void describe() {
24        System.out.println("Player: " + this.name);
25        System.out.println("Score: " + this.score);
26        System.out.println("has hand:");
27        System.out.println(hand.toString());
28    }
29
30    // takes a card out of the player's hand and returns it. also deletes card from
31    // the arraylist
32    public Card flip() {
33        Card selectedCard = this.hand.get(this.hand.size() - 1);
34        this.hand.remove(this.hand.size() - 1);
35        return selectedCard;
36    }
37
38    // takes a card from a given deck and adds it to the players hand
39    public void draw(Deck d) {
40        this.hand.add(d.draw());
41    }
42
43    // increases player's score by 1
44    public void incrementScore() {
45        this.score++;
46    }
47
48    // returns int for the size of player's hand
49    public int handSize() {
50        return this.hand.size();
51    }
52}
```

Deck.java Card.java Player.java X App.java

```
6     private List<Card> hand = new ArrayList<>();
7     private int score;
8     private String name;
9
10    // no default constructor, must name the player
11    Player(String name) {
12        this.name = name;
13        this.score = 0;
14    }
15
16    // returns int of the player's current score
17    public int getScore() {
18        return this.score;
19    }
20
21    // prints out information of the player to the console
22    // hand, name, score
23    public void describe() {
24        System.out.println("Player: " + this.name);
25        System.out.println("Score: " + this.score);
26        System.out.println("has hand:");
27        System.out.println(hand.toString());
28    }
29
30    // takes a card out of the player's hand and returns it. also deletes card from
31    // the arraylist
32    public Card flip() {
33        Card selectedCard = this.hand.get(this.hand.size() - 1);
34        this.hand.remove(this.hand.size() - 1);
35        return selectedCard;
36    }
37
38    // takes a card from a given deck and adds it to the players hand
39    public void draw(Deck d) {
40        this.hand.add(d.draw());
41    }
42
43    // increases player's score by 1
44    public void incrementScore() {
45        this.score++;
46    }
47
48    // returns int for the size of player's hand
49    public int handSize() {
50        return this.hand.size();
51    }
52
53    // returns string of player's name
54    public String getName() {
55        return this.name;
56    }
57 }
58
```



```
Deck.java  Card.java  Player.java  App.java X
1
2 public class App {
3
4     public static void main(String[] args) {
5
6         // creates a new deck
7         Deck d = new Deck();
8         System.out.println("Deck created of size: " + d.size());
9         System.out.println(d + "\n");
10
11        // shuffles the order of the deck
12        d.shuffle();
13        System.out.println("Deck shuffled: ");
14        System.out.println(d + "\n");
15
16        // creates 2 players
17        Player colby = new Player("Colby");
18        Player jack = new Player("Jack");
19
20        // for loop gives every other player a card from the deck
21        // while loop might be better but assignment calls for a for loop
22        int initialDeckSize = d.size();
23        for (int i = 0; i < initialDeckSize; i++) {
24            if (i % 2 == 0) {
25                colby.draw(d);
26            } else {
27                jack.draw(d);
28            }
29        }
30
31        // temporary card value storage
32        int p1Val = 0;
33        int p2Val = 0;
34        int initialHandSize = colby.handSize();
35
36        // test functionality of describe (could put it in the loop for each round
37        // but I didn't want to clutter the terminal more than i already have
38        colby.describe();
39        jack.describe();
```



```
31 // temporary card value storage
32 int p1Val = 0;
33 int p2Val = 0;
34 int initialHandSize = colby.handSize();
35
36 // test functionality of describe (could put it in the loop for each round
37 // but I didn't want to clutter the terminal more than i already have
38 colby.describe();
39 jack.describe();
40
41 // flips through each player's hand to compare values, adding score to the
42 // winning player
43 // while loop would still be better here
44
45 for (int i = 0; i < initialHandSize; i++) {
46
47     p1Val = colby.flip().getValue();
48     p2Val = jack.flip().getValue();
49
50     if (p1Val == p2Val) {
51         // do nothing
52         continue;
53     } else if (p1Val > p2Val) {
54         colby.incrementScore();
55     } else if (p2Val > p1Val) {
56         jack.incrementScore();
57     }
58     // System.out.println("\n\n\n");
59     // System.out.println("----- Round " + (i+1) + " -----");
60     // colby.describe();
61     // jack.describe();
62
63 }
64
65 // using getters
66 int finalP1Score = colby.getScore();
67 int finalP2Score = jack.getScore();
68
69 System.out.println("\n\n\n");
70
71 // prints out the winner!
72 if (finalP1Score == finalP2Score) {
73     System.out.println("It's a draw!!");
74 } else if (finalP1Score > finalP2Score) {
75     System.out.println(colby.getName() + " won!");
76 } else if (finalP2Score > finalP1Score) {
77     System.out.println(jack.getName() + " won!");
78 }
79
80 }
81
82 }
83
```

**Screenshots of Running Application:**

```
Problems Javadoc Declaration Console X Terminal
<terminated> App (1) [Java Application] C:\Program Files\Java\jdk-11.0.15\bin\java.exe (Jul 23, 2022, 4:04:08 AM - 4:04:09 AM) [pid: 15224]
Deck created:
[Two of Diamonds, Three of Diamonds, Four of Diamonds, Five of Diamonds, Six of Diamonds, Seven of Diamonds, Eight of Diamonds, Nine of Diamonds, Ten of Diamonds, Jack of Diamonds, Queen of Diamonds, King of Diamonds, Ace of Diamonds,
Deck shuffled:
[Eight of Spades, Four of Hearts, Jack of Hearts, Six of Diamonds, Queen of Clubs, Ace of Diamonds, Ace of Clubs, Ten of Clubs, Nine of Spades, Eight of Clubs, Queen of Spades, Six of Spades, Four of Clubs, Eight of Diamonds, Three of
Player: Colby
Score: 0
has hand:
[Ace of Spades, Jack of Diamonds, Five of Diamonds, Three of Diamonds, Two of Hearts, Two of Clubs, Eight of Hearts, Five of Spades, Queen of Hearts, Ten of Spades, King of Spades, Queen of Diamonds, Jack of Clubs, Three of Spades, Se
Player: Jack
Score: 0
has hand:
[Ace of Hearts, Seven of Diamonds, King of Diamonds, Four of Diamonds, Three of Hearts, Five of Hearts, Nine of Diamonds, Ten of Hearts, Two of Spades, Two of Diamonds, King of Hearts, Jack of Spades, Nine of Clubs, King of Clubs, Six
Jack won!
```

```
Problems Javadoc Declaration Console X Terminal
<terminated> App (1) [Java Application] C:\Program Files\Java\jdk-11.0.15\bin\java.exe (Jul 23, 2022, 4:15:24 AM - 4:15:24 AM) [pid: 5124]
Deck created of size: 52
[Two of Diamonds, Three of Diamonds, Four of Diamonds, Five of Diamonds, Six of Diamonds, Seven of Diamonds, Eight of Diamonds, Nine of Diamonds, Ten of Diamonds, Jack of Diamonds, Queen of Diamonds, King of Diamonds, Ace of Diamonds, I
Deck shuffled:
[Eight of Clubs, Ace of Clubs, Two of Diamonds, Jack of Hearts, Two of Hearts, Seven of Diamonds, Six of Spades, Two of Clubs, King of Spades, Three of Spades, Five of Hearts, Ace of Diamonds, Queen of Clubs, Seven of Spades, Four of Sp
Player: Colby
Score: 0
has hand:
[Seven of Hearts, Nine of Clubs, Queen of Hearts, King of Clubs, Ace of Hearts, Five of Diamonds, Ten of Hearts, Four of Diamonds, Four of Hearts, Three of Hearts, Ten of Diamonds, King of Hearts, Six of Diamonds, Queen of Diamonds, Que
Player: Jack
Score: 0
has hand:
[Ten of Clubs, Three of Clubs, Nine of Hearts, Eight of Diamonds, Four of Clubs, Nine of Spades, Three of Diamonds, Eight of Spades, Five of Clubs, Nine of Diamonds, Jack of Clubs, Eight of Hearts, Six of Clubs, Two of Spades, Jack of S
Colby won!
```

```
Problems Javadoc Declaration Console X Terminal
<terminated> App (1) [Java Application] C:\Program Files\Java\jdk-11.0.15\bin\java.exe (Jul 23, 2022, 4:15:45 AM - 4:15:46 AM) [pid: 17540]
Deck created of size: 52
[Two of Diamonds, Three of Diamonds, Four of Diamonds, Five of Diamonds, Six of Diamonds, Seven of Diamonds, Eight of Diamonds, Nine of Diamonds, Ten of Diamonds, Jack of Diamonds, Queen of Diamonds, King of Diamonds, Ace of Diamonds, T
Deck shuffled:
[Jack of Clubs, Seven of Clubs, Four of Spades, King of Clubs, Seven of Spades, Six of Clubs, Five of Spades, Five of Clubs, Nine of Hearts, Four of Hearts, Four of Clubs, Ten of Hearts, Six of Spades, Five of Diamonds, Three of Clubs,
Player: Colby
Score: 0
has hand:
[King of Diamonds, Two of Spades, Queen of Hearts, King of Hearts, Ace of Hearts, Three of Spades, Queen of Diamonds, Four of Diamonds, Queen of Clubs, Eight of Diamonds, Two of Clubs, Eight of Hearts, Jack of Diamonds, Two of Diamonds,
Player: Jack
Score: 0
has hand:
[Seven of Hearts, Nine of Spades, Ten of Diamonds, Six of Hearts, Ten of Clubs, Three of Diamonds, Nine of Diamonds, Ace of Spades, Five of Hearts, Seven of Diamonds, Eight of Clubs, Ace of Diamonds, Queen of Spades, Ten of Spades, Nine
It's a draw!!
```

URL to GitHub Repository:

<https://github.com/PCarmona5745/PromineoJavaFinalProject>