

# 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:

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

```
1 package war_game;
2
3 //Question 1a. Create Card class.
4 public class Card {
5
6     //i. Define value and name fields for Card.
7     private int value;
8     private String name;
9
10    //ii. Define card methods.
11    //1. getters and setters
12    public int getValue() {
13        return value;
14    }
15    public void setValue(int value) {
16        this.value = value;
17    }
18    public String getName() {
19        return name;
20    }
21    public void setName(String name) {
22        this.name = name;
23    }
24
25    //2. describe method for printing card info
26    public String describe() {
27        return name;
28    }
29
30    //Constructor used to build Deck.
31    public Card(int value, int suit) {
32        this.value = value;
33        switch(value) {
34            case 2:
35                switch(suit) {
36                    case 1:
37                        name = "Two of Spades";
38                        break;
```

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

```
38         break;
39     case 2:
40         name = "Two of Clubs";
41         break;
42     case 3:
43         name = "Two of Diamonds";
44         break;
45     case 4:
46         name = "Two of Hearts";
47         break;
48     }
49     break;
50
51     case 3:
52         switch(suit) {
53             case 1:
54                 name = "Three of Spades";
55                 break;
56             case 2:
57                 name = "Three of Clubs";
58                 break;
59             case 3:
60                 name = "Three of Diamonds";
61                 break;
62             case 4:
63                 name = "Three of Hearts";
64                 break;
65             }
66         break;
67
68     case 4:
69         switch(suit) {
70             case 1:
71                 name = "Four of Spades";
72                 break;
73             case 2:
74                 name = "Four of Clubs";
75                 break;
```

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

```
75         break;
76     case 3:
77         name = "Four of Diamonds";
78         break;
79     case 4:
80         name = "Four of Hearts";
81         break;
82     }
83     break;
84
85 case 5:
86     switch(suit) {
87     case 1:
88         name = "Five of Spades";
89         break;
90     case 2:
91         name = "Five of Clubs";
92         break;
93     case 3:
94         name = "Five of Diamonds";
95         break;
96     case 4:
97         name = "Five of Hearts";
98         break;
99     }
100     break;
101
102 case 6:
103     switch(suit) {
104     case 1:
105         name = "Six of Spades";
106         break;
107     case 2:
108         name = "Six of Clubs";
109         break;
110     case 3:
111         name = "Six of Diamonds";
112         break;
```

```
Card.java × Deck.java Player.java App.java
112         break;
113     case 4:
114         name = "Six of Hearts";
115         break;
116     }
117     break;
118
119     case 7:
120         switch(suit) {
121             case 1:
122                 name = "Seven of Spades";
123                 break;
124             case 2:
125                 name = "Seven of Clubs";
126                 break;
127             case 3:
128                 name = "Seven of Diamonds";
129                 break;
130             case 4:
131                 name = "Seven of Hearts";
132                 break;
133         }
134         break;
135
136     case 8:
137         switch(suit) {
138             case 1:
139                 name = "Eight of Spades";
140                 break;
141             case 2:
142                 name = "Eight of Clubs";
143                 break;
144             case 3:
145                 name = "Eight of Diamonds";
146                 break;
147             case 4:
148                 name = "Eight of Hearts";
149                 break;
```

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

```
149         break;
150     }
151     break;
152
153     case 9:
154         switch(suit) {
155             case 1:
156                 name = "Nine of Spades";
157                 break;
158             case 2:
159                 name = "Nine of Clubs";
160                 break;
161             case 3:
162                 name = "Nine of Diamonds";
163                 break;
164             case 4:
165                 name = "Nine of Hearts";
166                 break;
167         }
168         break;
169
170     case 10:
171         switch(suit) {
172             case 1:
173                 name = "Ten of Spades";
174                 break;
175             case 2:
176                 name = "Ten of Clubs";
177                 break;
178             case 3:
179                 name = "Ten of Diamonds";
180                 break;
181             case 4:
182                 name = "Ten of Hearts";
183                 break;
184         }
185         break;
```

```
Card.java × Deck.java Player.java App.java
184     },
185     break;
186
187     case 11:
188         switch(suit) {
189             case 1:
190                 name = "Jack of Spades";
191                 break;
192             case 2:
193                 name = "Jack of Clubs";
194                 break;
195             case 3:
196                 name = "Jack of Diamonds";
197                 break;
198             case 4:
199                 name = "Jack of Hearts";
200                 break;
201         }
202         break;
203
204     case 12:
205         switch(suit) {
206             case 1:
207                 name = "Queen of Spades";
208                 break;
209             case 2:
210                 name = "Queen of Clubs";
211                 break;
212             case 3:
213                 name = "Queen of Diamonds";
214                 break;
215             case 4:
216                 name = "Queen of Hearts";
217                 break;
218         }
219         break;
220
221     case 13:
```



```
Card.java × Deck.java Player.java App.java
220
221     case 13:
222         switch(suit) {
223             case 1:
224                 name = "King of Spades";
225                 break;
226             case 2:
227                 name = "King of Clubs";
228                 break;
229             case 3:
230                 name = "King of Diamonds";
231                 break;
232             case 4:
233                 name = "King of Hearts";
234                 break;
235             }
236         break;
237
238     case 14:
239         switch(suit) {
240             case 1:
241                 name = "Ace of Spades";
242                 break;
243             case 2:
244                 name = "Ace of Clubs";
245                 break;
246             case 3:
247                 name = "Ace of Diamonds";
248                 break;
249             case 4:
250                 name = "Ace of Hearts";
251                 break;
252             }
253         break;
254     }
255
256
257 }
```

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

```
1 package war_game;
2
3 import java.util.*;
4
5 //Question 1b. Create Deck class.
6 public class Deck {
7
8     //i. Declare list of Card named cards
9     private List<Card> cards = new ArrayList<Card>();
10
11     //ii. Declare Deck methods.
12
13     //1. Method that shuffles the Deck.
14     public void shuffle() {
15         Collections.shuffle(cards);
16     }
17
18     //2. Method that draws and removes top card from Deck.
19     public Card draw() {
20         return cards.remove(0);
21     }
22
23     //3. Constructor that instantiates Deck and fills it with 52 cards.
24     public Deck() {
25         for (int value = 2; value <= 14; value++) {
26             for (int suit = 1; suit <= 4; suit++) {
27                 Card card = new Card(value, suit);
28                 cards.add(card);
29             }
30         }
31     }
32 }
33
```

```
Card.java  Deck.java  Player.java  App.java
1 package war_game;
2
3 import java.util.*;
4
5 //Question 1c. Create the Player class.
6 public class Player {
7
8     //i. Declare variables associated to Player class.
9     private List<Card> hand = new ArrayList<Card>();
10    private String name;
11    private int score;
12
13    //ii. Declare methods associated to Player class.
14    //1. Print out player name and cards in player's hand
15    public void describe() {
16        System.out.println(name + "'s Hand:");
17        for (int i = 0; i < 26; i++) {
18            System.out.println(hand.get(i).describe());
19        }
20    }
21
22    //2. Remove and return the top card in player's hand.
23    public Card flip() {
24        return hand.remove(0);
25    }
26
27    //3. Calls the draw method on the deck and adds Card to hand.
28    public void draw(Deck deck) {
29        hand.add(deck.draw());
30    }
31
32    //4. Add 1 to player score.
33    public void incrementScore() {
34        setScore(getScore() + 1);
35    }
36}
```

```

22 //2. Remove and return the top card in player's hand.
23 public Card flip() {
24     return hand.remove(0);
25 }
26
27 //3. Calls the draw method on the deck and adds Card to hand.
28 public void draw(Deck deck) {
29     hand.add(deck.draw());
30 }
31
32 //4. Add 1 to player score.
33 public void incrementScore() {
34     setScore(getScore() + 1);
35 }
36
37 //Constructor
38 public Player(String name) {
39     this.setScore(0);
40     this.name = name;
41 }
42
43 //getters and setters
44 public int getScore() {
45     return score;
46 }
47
48 public void setScore(int score) {
49     this.score = score;
50 }
51
52 public String getName() {
53     return name;
54 }
55
56 public void setName(String name) {
57     this.name = name;
58 }
59 }
60

```

```

Card.java  Deck.java  Player.java  App.java X
1  package war_game;
2
3  //Question 2. Create class named App
4  public class App {
5
6      public static void main(String[] args) {
7
8          //Question 3. Instantiate a Deck and two Players.
9          Deck deck = new Deck();
10         Player player1 = new Player("Garrett");
11         Player player2 = new Player("War Machine");
12
13         //Shuffle the Deck.
14         deck.shuffle();
15
16         //Question 4 Draw Player hands.
17         for (int i = 0; i < 52; i++) {
18             if (i < 26) {
19                 player1.draw(deck);
20             } else if (i >= 26 && i < 52) {
21                 player2.draw(deck);
22             }
23         }
24
25         //Show player hands
26         System.out.println("*****");
27         System.out.println("*****Player Hands*****");
28         System.out.println("*****");
29         player1.describe();
30         System.out.println("*****");
31         player2.describe();

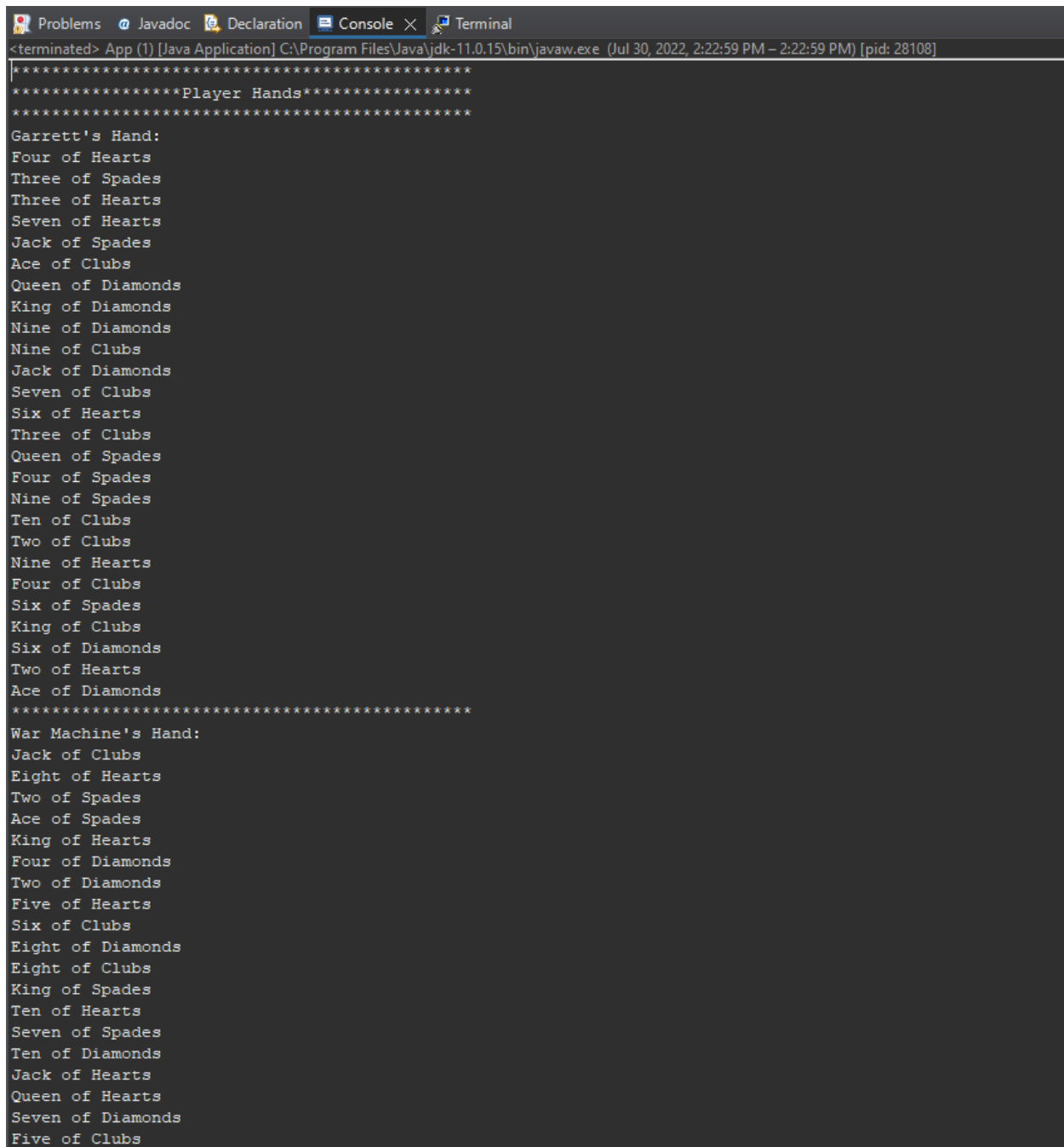
```

```

Card.java Deck.java Player.java App.java X
25 //Show player hands
26 System.out.println("*****");
27 System.out.println("*****Player Hands*****");
28 System.out.println("*****");
29 player1.describe();
30 System.out.println("*****");
31 player2.describe();
32
33 //Question 5. Flip each player's hand.
34 System.out.println("*****");
35 System.out.println("*****Matchups*****");
36 System.out.println("*****");
37 for (int i = 0; i < 26; i++) {
38     Card player1Card = player1.flip();
39     Card player2Card = player2.flip();
40     System.out.println(player1Card.getName() + " vs " + player2Card.getName());
41     if (player1Card.getValue() < player2Card.getValue()) {
42         player2.incrementScore();
43     } else if (player1Card.getValue() > player2Card.getValue()) {
44         player1.incrementScore();
45     }
46 }
47
48 //Question 6 & 7 Compare final scores and print final scores and winner if there is one.
49 System.out.println("*****");
50 System.out.println("*****Final Result*****");
51 System.out.println("*****");
52 System.out.println(player1.getName() + "'s score: " + player1.getScore());
53 System.out.println(player2.getName() + "'s score: " + player2.getScore());
54 if (player1.getScore() > player2.getScore()) {
55     System.out.println(player1.getName() + " Wins!");
56 } else if (player1.getScore() < player2.getScore()) {
57     System.out.println(player2.getName() + " Wins!");
58 } else {
59     System.out.println("It's a draw!");
60 }
61
62

```

## Screenshots of Running Application:



```
<terminated> App (1) [Java Application] C:\Program Files\Java\jdk-11.0.15\bin\javaw.exe (Jul 30, 2022, 2:22:59 PM – 2:22:59 PM) [pid: 28108]
*****
*****Player Hands*****
*****
Garrett's Hand:
Four of Hearts
Three of Spades
Three of Hearts
Seven of Hearts
Jack of Spades
Ace of Clubs
Queen of Diamonds
King of Diamonds
Nine of Diamonds
Nine of Clubs
Jack of Diamonds
Seven of Clubs
Six of Hearts
Three of Clubs
Queen of Spades
Four of Spades
Nine of Spades
Ten of Clubs
Two of Clubs
Nine of Hearts
Four of Clubs
Six of Spades
King of Clubs
Six of Diamonds
Two of Hearts
Ace of Diamonds
*****
War Machine's Hand:
Jack of Clubs
Eight of Hearts
Two of Spades
Ace of Spades
King of Hearts
Four of Diamonds
Two of Diamonds
Five of Hearts
Six of Clubs
Eight of Diamonds
Eight of Clubs
King of Spades
Ten of Hearts
Seven of Spades
Ten of Diamonds
Jack of Hearts
Queen of Hearts
Seven of Diamonds
Five of Clubs
```

```
Problems Javadoc Declaration Console X Terminal
<terminated> App (1) [Java Application] C:\Program Files\Java\jdk-11.0.15\bin\javaw.exe (Jul 30, 2022, 2:22:59 PM – 2:22:59 PM) [pid: 28108]
Eight of Clubs
King of Spades
Ten of Hearts
Seven of Spades
Ten of Diamonds
Jack of Hearts
Queen of Hearts
Seven of Diamonds
Five of Clubs
Three of Diamonds
Ace of Hearts
Five of Spades
Eight of Spades
Five of Diamonds
Queen of Clubs
Ten of Spades
*****
*****Matchups*****
*****
Four of Hearts vs Jack of Clubs
Three of Spades vs Eight of Hearts
Three of Hearts vs Two of Spades
Seven of Hearts vs Ace of Spades
Jack of Spades vs King of Hearts
Ace of Clubs vs Four of Diamonds
Queen of Diamonds vs Two of Diamonds
King of Diamonds vs Five of Hearts
Nine of Diamonds vs Six of Clubs
Nine of Clubs vs Eight of Diamonds
Jack of Diamonds vs Eight of Clubs
Seven of Clubs vs King of Spades
Six of Hearts vs Ten of Hearts
Three of Clubs vs Seven of Spades
Queen of Spades vs Ten of Diamonds
Four of Spades vs Jack of Hearts
Nine of Spades vs Queen of Hearts
Ten of Clubs vs Seven of Diamonds
Two of Clubs vs Five of Clubs
Nine of Hearts vs Three of Diamonds
Four of Clubs vs Ace of Hearts
Six of Spades vs Five of Spades
King of Clubs vs Eight of Spades
Six of Diamonds vs Five of Diamonds
Two of Hearts vs Queen of Clubs
Ace of Diamonds vs Ten of Spades
*****
*****Final Result*****
*****
Garrett's score: 14
War Machine's score: 12
Garrett Wins!
```

URL to GitHub Repository:

<https://github.com/gyarbrough/War-Game>