

# 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. Lastly, in the Learning Management System, click the "Add Submission" button and paste the URL to your GitHub repository.

## 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 Deck.java Player.java War.java
1 package warHomeworkWeek6;
2
3 public class Card {
4
5     private int value;
6     private String name;
7
8     public Card(String name, int value) {
9         this.name = name;
10        this.value = value;
11    }
12
13
14    public String describe() {
15        return toString();
16    }
17
18    public int getValue() {
19        return value;
20    }
21    public void setValue(int value) {
22        this.value = value;
23    }
24    public String getName() {
25        return name;
26    }
27    public void setName(String name) {
28        this.name = name;
29    }
30    @Override
31    public String toString() {
32        return "Card [value=" + value + ", name=" + name + "]";
33    }
34 }
35
36
37
38
39
40
```

```
Card.java *Deck.java Player.java War.java
1 package warHomeworkWeek6;
2
3 import java.util.ArrayList;
4 import java.util.Collections;
5 import java.util.List;
6
7 public class Deck {
8     private List<Card> cards = new ArrayList<>();
9     private List<Integer> numbers = List.of(2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14);
10    private List<String> suits = List.of("Diamonds", "Clubs", "Hearts", "Spades");
11
12    public Deck() {
13        for(Integer number : numbers) {
14            for(int index=0; index < suits.size(); index++) {
15                String name = number + " of " + suits.get(index);
16                int value = number;
17                cards.add(new Card(name, value));
18            }
19        }
20    }
21
22    public void shuffle() {
23        Collections.shuffle(cards);
24    }
25
26    public Card draw() {
27        return cards.remove(0);
28    }
29
30    @Override
31    public String toString() {
32        return "Deck card=" + cards;
33    }
34
35    public int size() {
36        return cards.size();
37    }
38
39
40
41
42
43
44
45
```

```
1 package warHomeworkWeek6;
2
3 import java.util.ArrayList;
4
5 public class Player {
6     private String name;
7     private int score;
8     private List<Card> hand = new ArrayList<>();
9
10
11     public Player(String name) {
12         this.name = name;
13     }
14
15     public int getScore() {
16         return score;
17     }
18
19     public String getName() {
20         return name;
21     }
22
23     public String describe() {
24         return toString();
25     }
26
27     public Card flip() {
28         if(!hand.isEmpty()) {
29             return hand.remove(0);
30         }
31         throw new IllegalStateException("Out of Cards!");
32     }
33
34     public void draw(Deck deck) {
35         hand.add(deck.draw());
36     }
37
38     public void incrementScore() {
39         score += 1;
40     }
41
42
43     public int size() {
44         return hand.size();
45     }
46     @Override
47     public String toString() {
48         return "Player [name=" + name + ", score=" + score + ", hand=" + hand + "];"
49     }
50
51 }
52
```

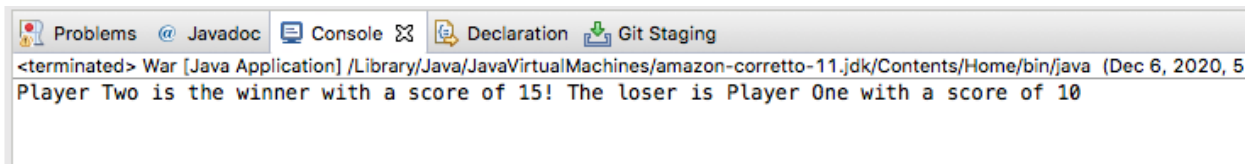
```

Card.java Deck.java Player.java War.java
1 package warHomeworkWeek6;
2
3 public class War {
4
5     public static void main(String[] args) {
6         Player p1 = new Player("Player One");
7         Player p2 = new Player("Player Two");
8
9         Deck deck = new Deck();
10        deck.shuffle();
11
12
13        dealCards(p1, p2, deck);
14
15        gamePlay(p1, p2);
16
17        printScore(p1, p2);
18
19    }
20
21
22    private static void printScore(Player p1, Player p2) {
23        if(p1.getScore() > p2.getScore()) {
24            declareWinner (p1, p2) ;|
25        }
26        else if (p2.getScore() > p1.getScore()) {
27            declareWinner (p2, p1);
28        } else {
29            System.out.println("Player " + p1.getName() + " and player " + p2.getName() + " tie");
30        }
31    }
32
33    private static void gamePlay(Player p1, Player p2) {
34        for(int turn = 0; turn < 26; turn++) {
35            Card c1 = p1.flip();
36            Card c2 = p2.flip();
37
38
39            if (c1.getValue()>c2.getValue()) {
40                p1.incrementScore();
41            } else if (c2.getValue() > c1.getValue()){
42                p2.incrementScore();
43            }
44        }
45    }
46
47    private static void dealCards(Player p1, Player p2, Deck deck) {
48        int deckSize = deck.size();
49
50        for(int index=0; index< deckSize; index++) {
51            if(index % 2 ==0) {
52                p1.draw(deck);
53            }
54            else {
55                p2.draw(deck);
56            }
57        }
58    }
59
60    private static void declareWinner (Player winner, Player loser) {
61        System.out.println(winner.getName() + " is the winner with a score of " + winner.getScore() + "!" + " The loser is " +
62        loser.getName() + " with a score of " + loser.getScore());
63    }
64
65
66
67
68
69 }

```

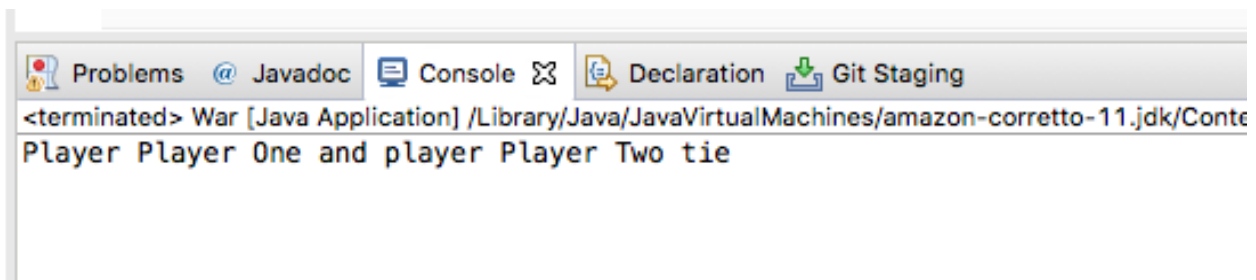
## Screenshots of Running Application:

### Game 1



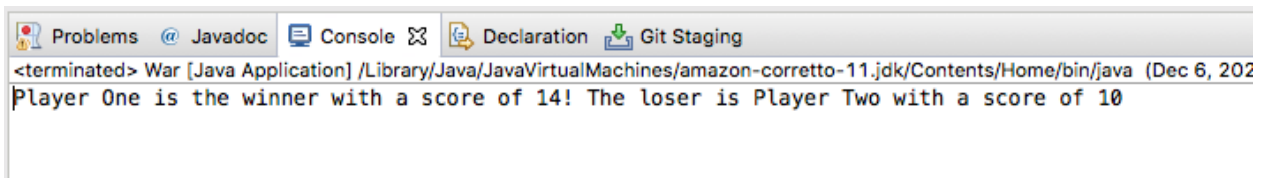
The screenshot shows the IDE's console window with the following text: `<terminated> War [Java Application] /Library/Java/JavaVirtualMachines/amazon-corretto-11.jdk/Contents/Home/bin/java (Dec 6, 2020, 5` followed by a new line `Player Two is the winner with a score of 15! The loser is Player One with a score of 10`. The IDE interface includes tabs for Problems, Javadoc, Console, Declaration, and Git Staging.

### Game 2



The screenshot shows the IDE's console window with the following text: `<terminated> War [Java Application] /Library/Java/JavaVirtualMachines/amazon-corretto-11.jdk/Conte` followed by a new line `Player Player One and player Player Two tie`. The IDE interface includes tabs for Problems, Javadoc, Console, Declaration, and Git Staging.

### Game 3



The screenshot shows the IDE's console window with the following text: `<terminated> War [Java Application] /Library/Java/JavaVirtualMachines/amazon-corretto-11.jdk/Contents/Home/bin/java (Dec 6, 202` followed by a new line `Player One is the winner with a score of 14! The loser is Player Two with a score of 10`. The IDE interface includes tabs for Problems, Javadoc, Console, Declaration, and Git Staging.

## URL to GitHub Repository:

<https://github.com/Christinalytle/week6War.git>