# 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:**

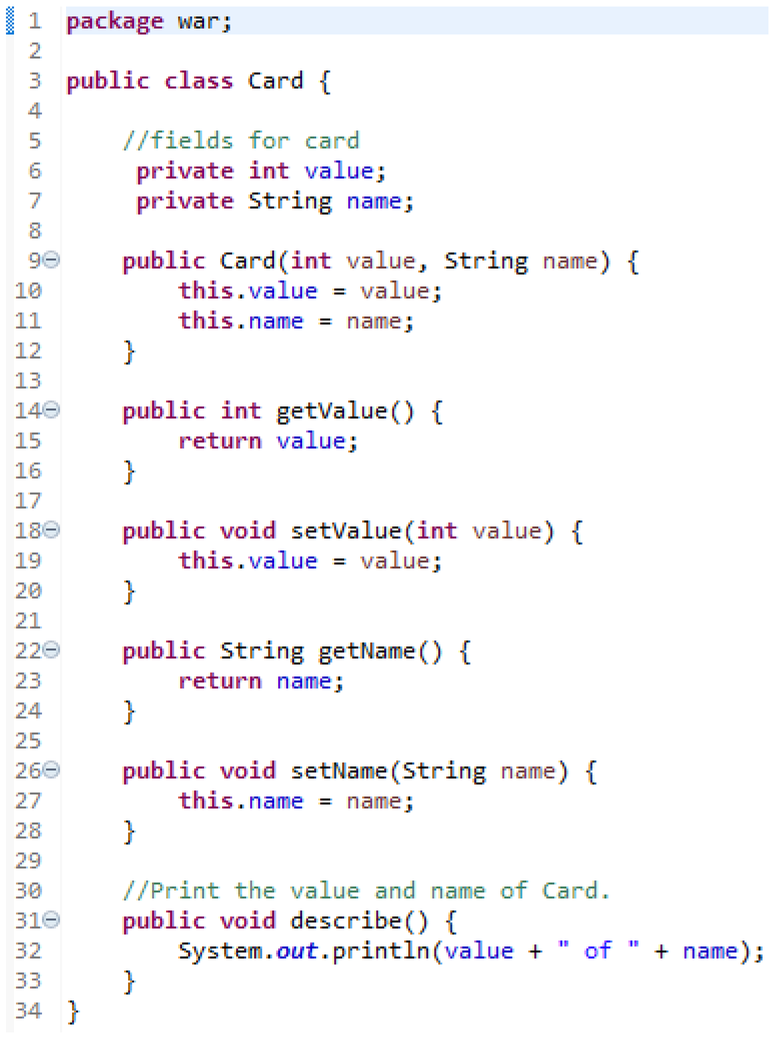
For the final project you will be creating an automated version of the classic card game *WAR.*

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)

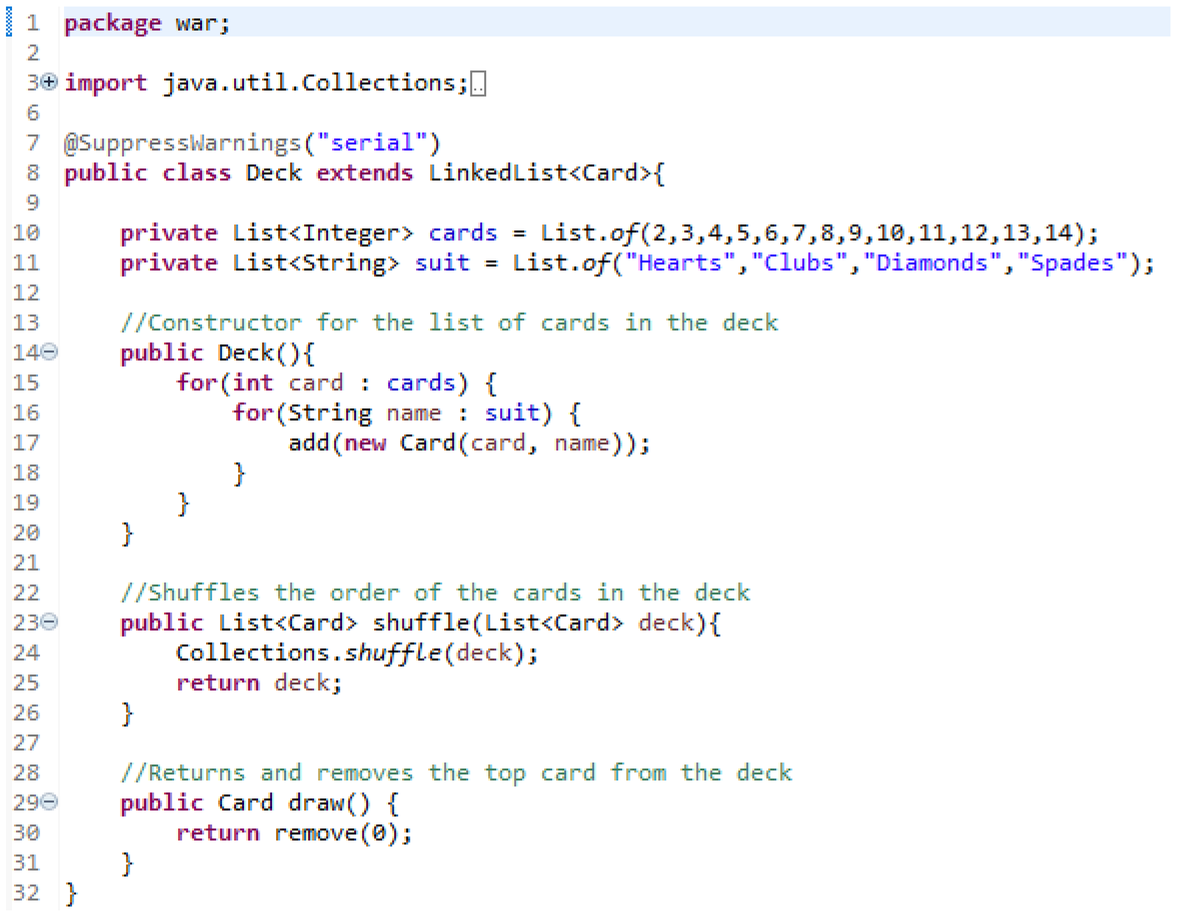
- - 3. In the constructor, when a new Deck is instantiated, the Cards field should be populated with the standard 52 cards.
  - c. Player
    - i. Fields
      1. **hand** (List of Card)
      2. **score** (set to 0 in the constructor)
      3. **name**
    - ii. Methods
      1. **describe** (prints 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.

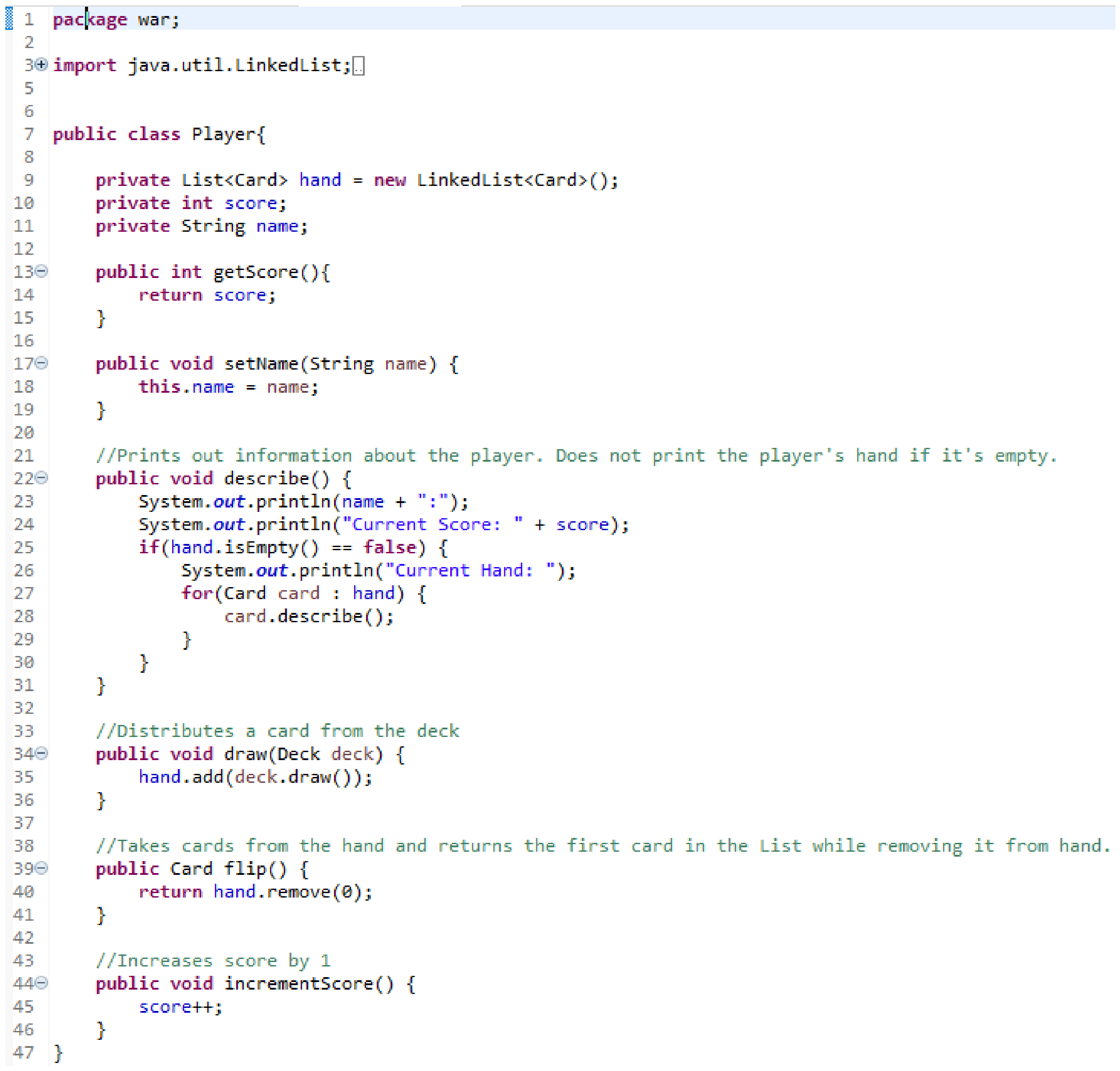
**Screenshots of Code:**



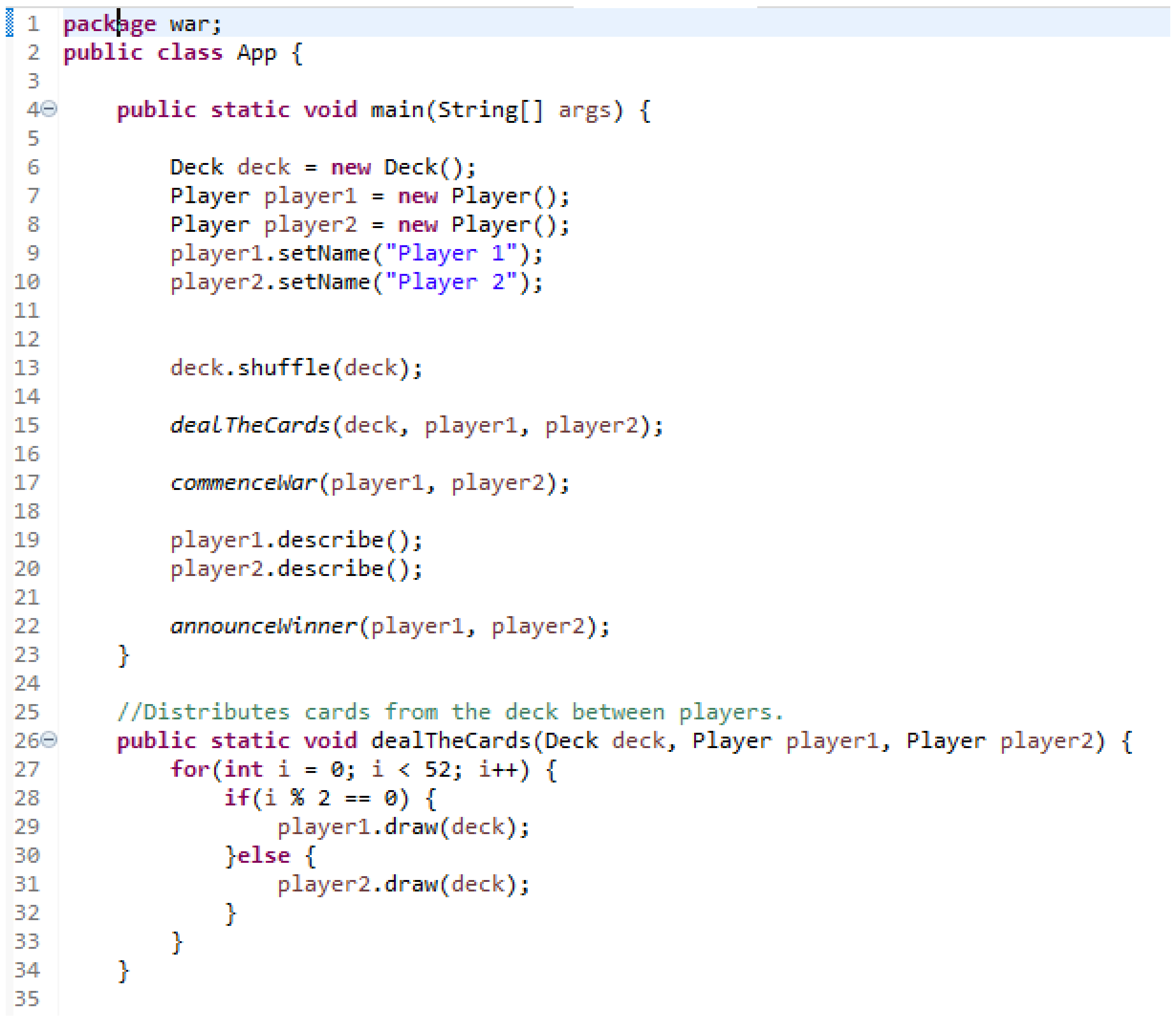
```java
package war;

public class Card {

    //fields for card
     private int value;
     private String name;

    public Card(int value, String name) {
        this.value = value;
        this.name = name;
    }

    public int getValue() {
        return value;
    }

    public void setValue(int value) {
        this.value = value;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    //Print the value and name of Card.
    public void describe() {
        System.out.println(value + " of " + name);
    }
}
```



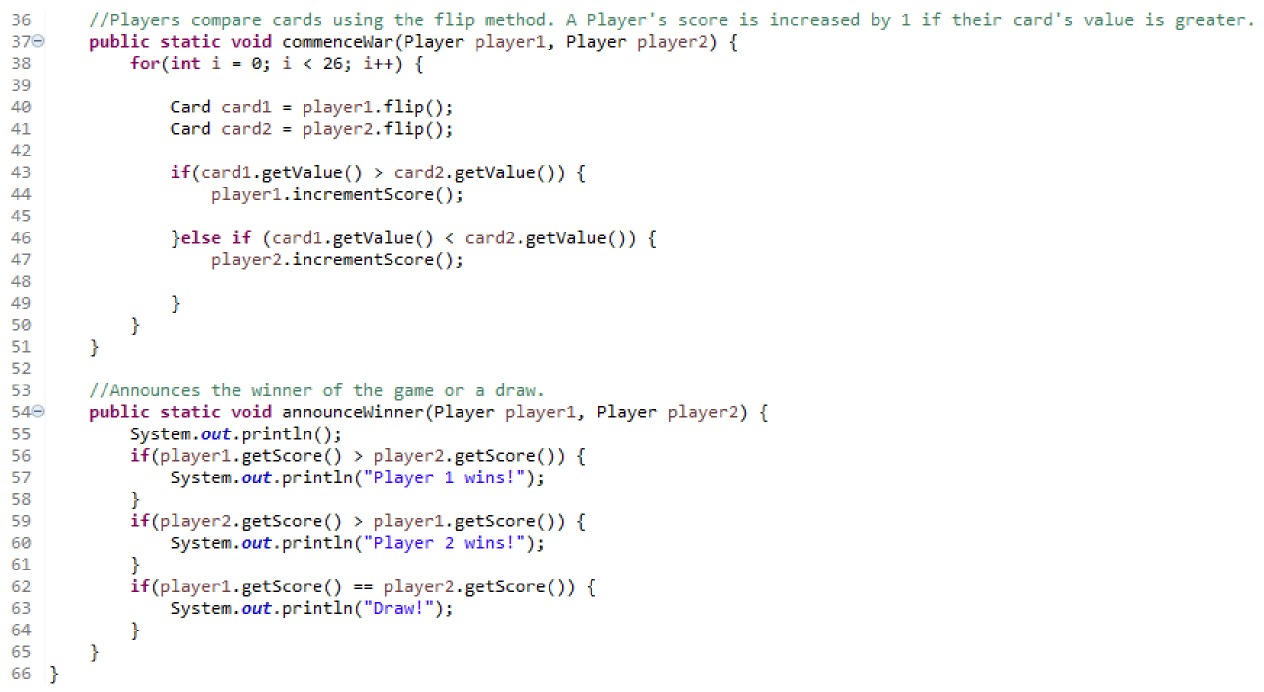
```java
package war;

import java.util.Collections;

@SuppressWarnings("serial")
public class Deck extends LinkedList<Card>{

    private List<Integer> cards = List.of(2,3,4,5,6,7,8,9,10,11,12,13,14);
    private List<String> suit = List.of("Hearts","Clubs","Diamonds","Spades");

    //Constructor for the list of cards in the deck
    public Deck(){
        for(int card : cards) {
            for(String name : suit) {
                add(new Card(card, name));
            }
        }
    }

    //Shuffles the order of the cards in the deck
    public List<Card> shuffle(List<Card> deck){
        Collections.shuffle(deck);
        return deck;
    }

    //Returns and removes the top card from the deck
    public Card draw() {
        return remove(0);
    }
}
```



```java
package war;

import java.util.LinkedList;


public class Player{

    private List<Card> hand = new LinkedList<Card>();
    private int score;
    private String name;

    public int getScore(){
        return score;
    }

    public void setName(String name) {
        this.name = name;
    }

    //Prints out information about the player. Does not print the player's hand if it's empty.
    public void describe() {
        System.out.println(name + ":");
        System.out.println("Current Score: " + score);
        if(hand.isEmpty() == false) {
            System.out.println("Current Hand: ");
            for(Card card : hand) {
                card.describe();
            }
        }
    }

    //Distributes a card from the deck
    public void draw(Deck deck) {
        hand.add(deck.draw());
    }

    //Takes cards from the hand and returns the first card in the List while removing it from hand.
    public Card flip() {
        return hand.remove(0);
    }

    //Increases score by 1
    public void incrementScore() {
        score++;
    }
}
```

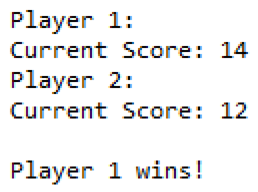


```java
package war;
public class App {

    public static void main(String[] args) {

        Deck deck = new Deck();
        Player player1 = new Player();
        Player player2 = new Player();
        player1.setName("Player 1");
        player2.setName("Player 2");


        deck.shuffle(deck);

        dealTheCards(deck, player1, player2);

        commenceWar(player1, player2);

        player1.describe();
        player2.describe();

        announceWinner(player1, player2);
    }

    //Distributes cards from the deck between players.
    public static void dealTheCards(Deck deck, Player player1, Player player2) {
        for(int i = 0; i < 52; i++) {
            if(i % 2 == 0) {
                player1.draw(deck);
            }else {
                player2.draw(deck);
            }
        }
    }

```



```java
    //Players compare cards using the flip method. A Player's score is increased by 1 if their card's value is greater.
    public static void commenceWar(Player player1, Player player2) {
        for(int i = 0; i < 26; i++) {

            Card card1 = player1.flip();
            Card card2 = player2.flip();

            if(card1.getValue() > card2.getValue()) {
                player1.incrementScore();

            }else if (card1.getValue() < card2.getValue()) {
                player2.incrementScore();

            }
        }
    }

    //Announces the winner of the game or a draw.
    public static void announceWinner(Player player1, Player player2) {
        System.out.println();
        if(player1.getScore() > player2.getScore()) {
            System.out.println("Player 1 wins!");
        }
        if(player2.getScore() > player1.getScore()) {
            System.out.println("Player 2 wins!");
        }
        if(player1.getScore() == player2.getScore()) {
            System.out.println("Draw!");
        }
    }
}
```

## Screenshots of Running Application:



```
Player 1:
Current Score: 14
Player 2:
Current Score: 12

Player 1 wins!
```

## URL to GitHub Repository:

DaltonCash/PT-WK6: Promineo Tech week 6 assignment. This code is War, a game where two players compare a card from the top of their decks. The player that has the highest card the most times wins. (github.com)