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:

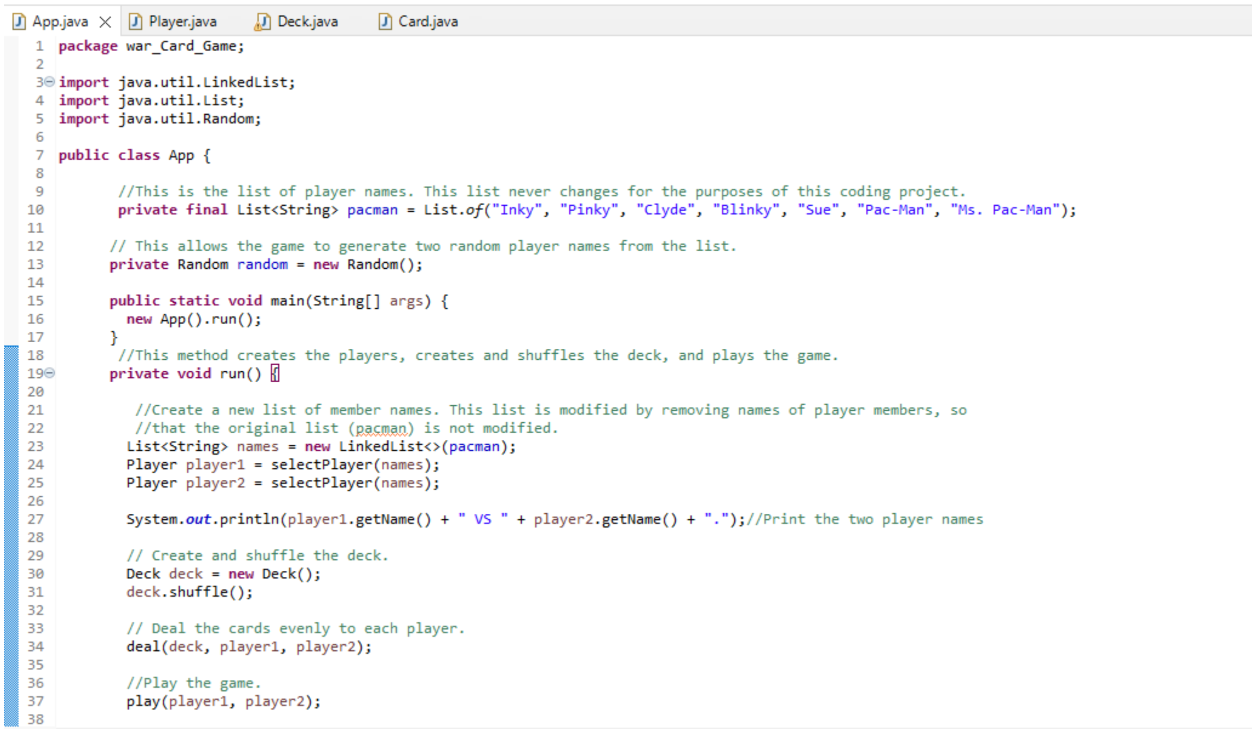
For the final project you will be creating an automated version of the classic card game *WAR.*

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)

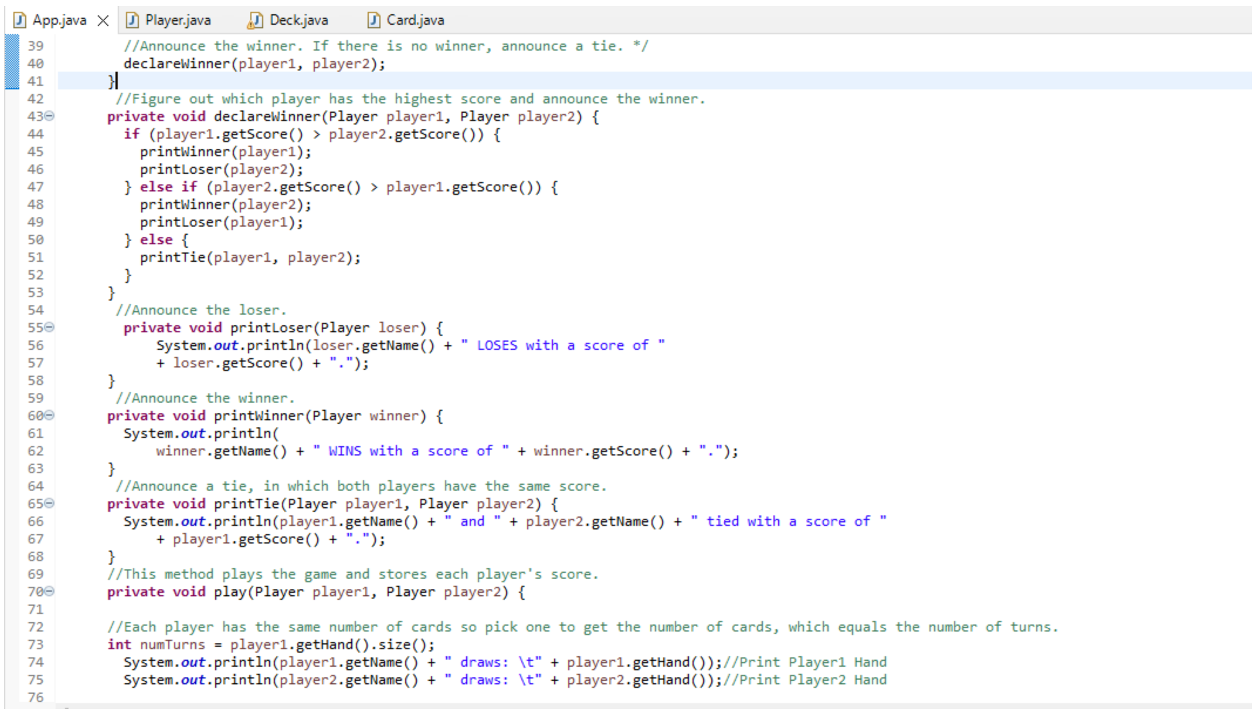
            3. In the constructor, when a new Deck is instantiated, the Cards field should be populated with the standard 52 cards.
    c. Player
        i. Fields
            1. **hand** (List of Card)
            2. **score** (set to 0 in the constructor)
            3. **name**
        ii. Methods
          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

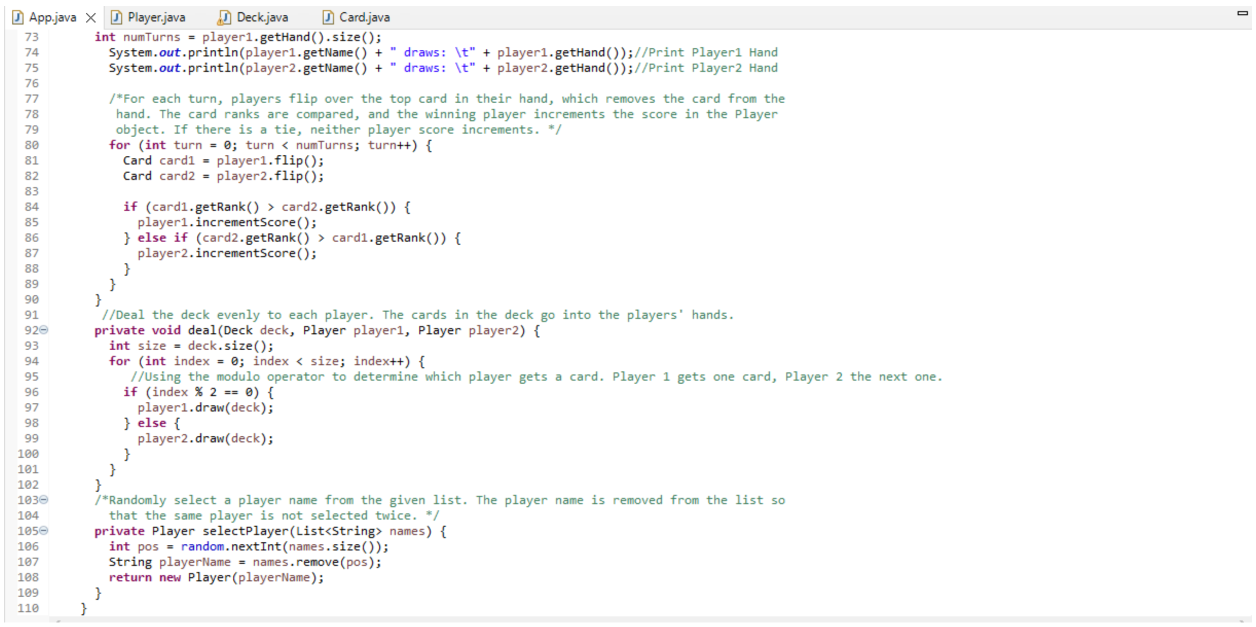
Screenshots of Code:



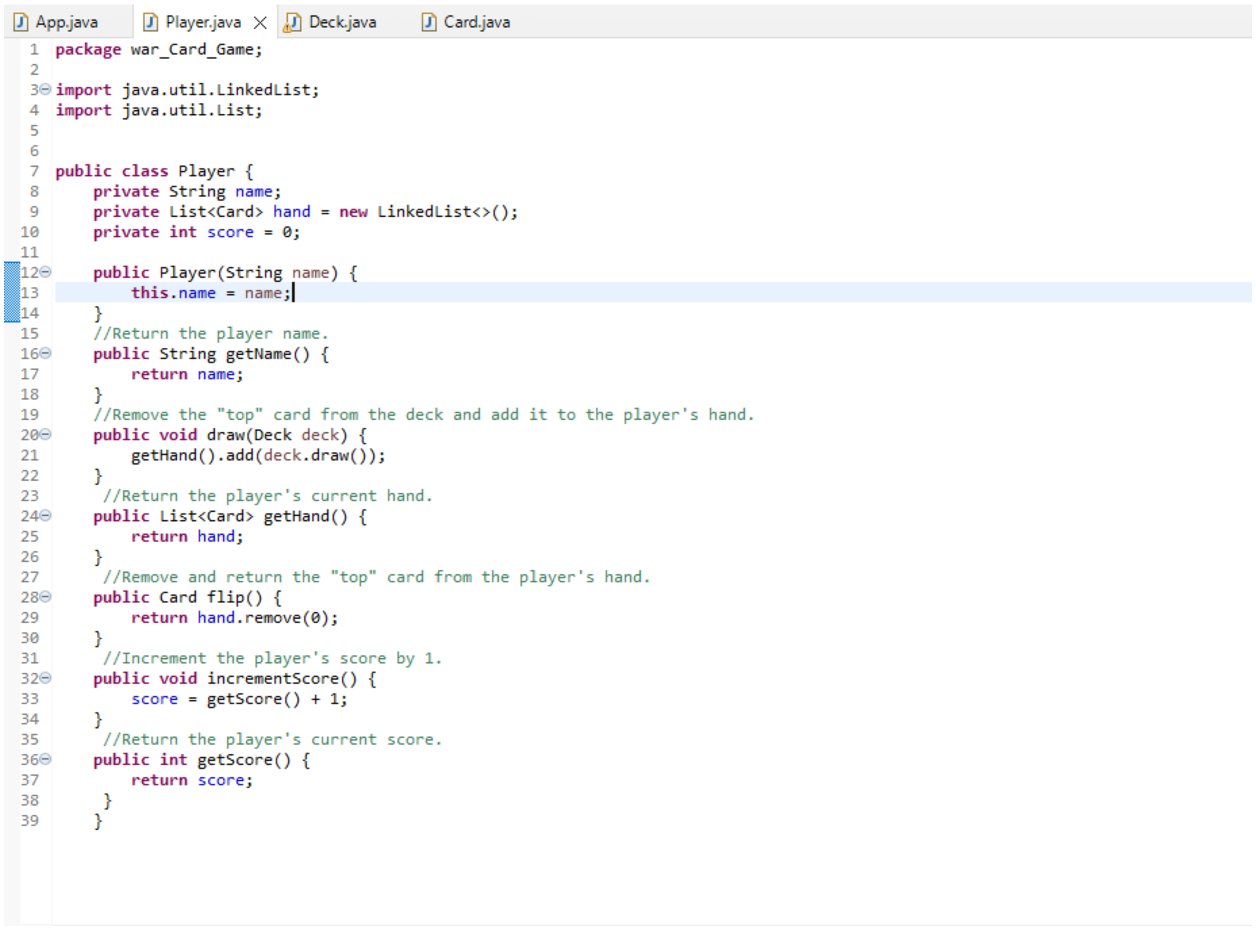
```java
package war_Card_Game;

import java.util.LinkedList;
import java.util.List;
import java.util.Random;

public class App {

	//This is the list of player names. This list never changes for the purposes of this coding project.
	private final List<String> pacman = List.of("Inky", "Pinky", "Clyde", "Blinky", "Sue", "Pac-Man", "Ms. Pac-Man");

	// This allows the game to generate two random player names from the list.
	private Random random = new Random();

	public static void main(String[] args) {
	  new App().run();
	}
	 //This method creates the players, creates and shuffles the deck, and plays the game.
	private void run() {

	   //Create a new list of member names. This list is modified by removing names of player members, so
	   //that the original list (pacman) is not modified.
	  List<String> names = new LinkedList<>(pacman);
	  Player player1 = selectPlayer(names);
	  Player player2 = selectPlayer(names);

	  System.out.println(player1.getName() + " VS " + player2.getName() + ".");//Print the two player names

	  // Create and shuffle the deck.
	  Deck deck = new Deck();
	  deck.shuffle();

	  // Deal the cards evenly to each player.
	  deal(deck, player1, player2);

	  //Play the game.
	  play(player1, player2);

```



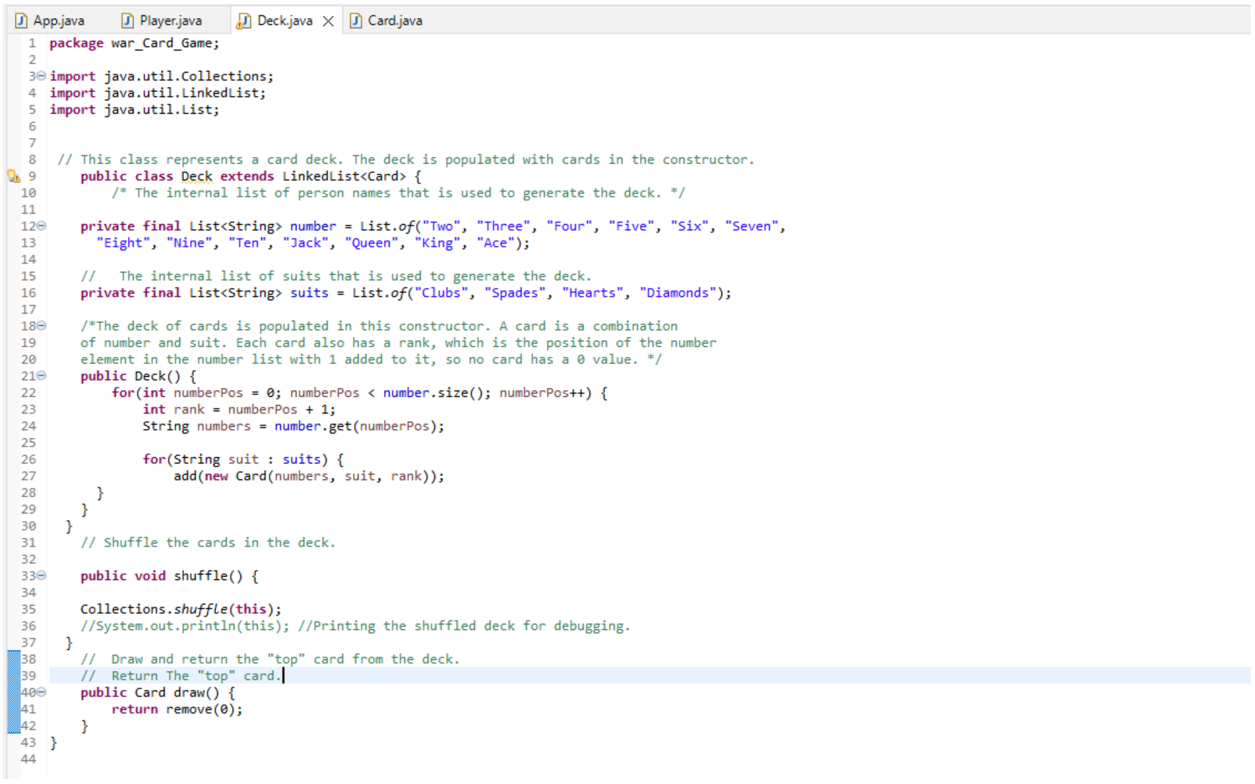
```java
	  //Announce the winner. If there is no winner, announce a tie. */
	  declareWinner(player1, player2);
	}
	 //Figure out which player has the highest score and announce the winner.
	private void declareWinner(Player player1, Player player2) {
	  if (player1.getScore() > player2.getScore()) {
	    printWinner(player1);
	    printLoser(player2);
	  } else if (player2.getScore() > player1.getScore()) {
	    printWinner(player2);
	    printLoser(player1);
	  } else {
	    printTie(player1, player2);
	  }
	}
	 //Announce the loser.
	  private void printLoser(Player loser) {
	      System.out.println(loser.getName() + " LOSES with a score of "
	      + loser.getScore() + ".");
	}
	 //Announce the winner.
	private void printWinner(Player winner) {
	  System.out.println(
	      winner.getName() + " WINS with a score of " + winner.getScore() + ".");
	}
	 //Announce a tie, in which both players have the same score.
	private void printTie(Player player1, Player player2) {
	  System.out.println(player1.getName() + " and " + player2.getName() + " tied with a score of "
	      + player1.getScore() + ".");
	}
	//This method plays the game and stores each player's score.
	private void play(Player player1, Player player2) {

	//Each player has the same number of cards so pick one to get the number of cards, which equals the number of turns.
	int numTurns = player1.getHand().size();
	  System.out.println(player1.getName() + " draws: \t" + player1.getHand());//Print Player1 Hand
	  System.out.println(player2.getName() + " draws: \t" + player2.getHand());//Print Player2 Hand

```



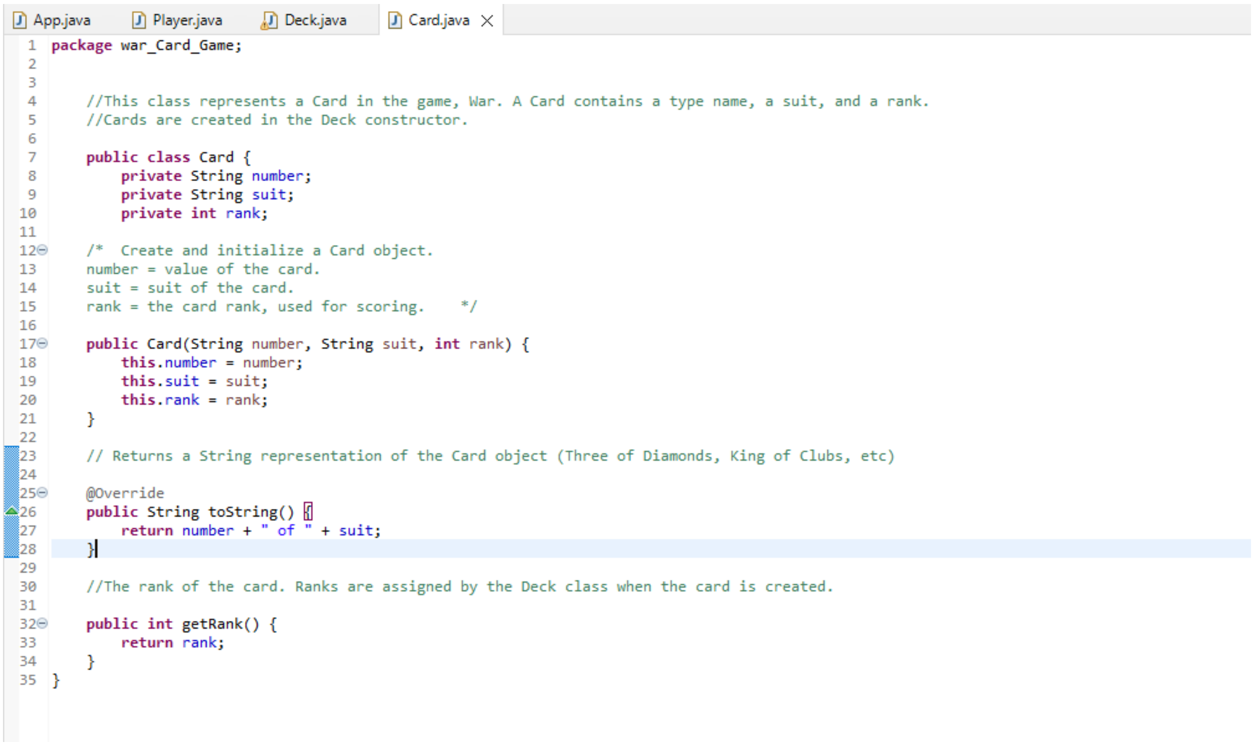
```java
      int numTurns = player1.getHand().size();
        System.out.println(player1.getName() + " draws: \t" + player1.getHand());//Print Player1 Hand
        System.out.println(player2.getName() + " draws: \t" + player2.getHand());//Print Player2 Hand

        /*For each turn, players flip over the top card in their hand, which removes the card from the
         hand. The card ranks are compared, and the winning player increments the score in the Player
         object. If there is a tie, neither player score increments. */
        for (int turn = 0; turn < numTurns; turn++) {
          Card card1 = player1.flip();
          Card card2 = player2.flip();

          if (card1.getRank() > card2.getRank()) {
            player1.incrementScore();
          } else if (card2.getRank() > card1.getRank()) {
            player2.incrementScore();
          }
        }
      }
       //Deal the deck evenly to each player. The cards in the deck go into the players' hands.
      private void deal(Deck deck, Player player1, Player player2) {
        int size = deck.size();
        for (int index = 0; index < size; index++) {
           //Using the modulo operator to determine which player gets a card. Player 1 gets one card, Player 2 the next one.
          if (index % 2 == 0) {
            player1.draw(deck);
          } else {
            player2.draw(deck);
          }
        }
      }
      /*Randomly select a player name from the given list. The player name is removed from the list so
        that the same player is not selected twice. */
      private Player selectPlayer(List<String> names) {
        int pos = random.nextInt(names.size());
        String playerName = names.remove(pos);
        return new Player(playerName);
      }
    }
```



```java
package war_Card_Game;

import java.util.LinkedList;
import java.util.List;


public class Player {
    private String name;
    private List<Card> hand = new LinkedList<>();
    private int score = 0;

    public Player(String name) {
        this.name = name;
    }
    //Return the player name.
    public String getName() {
        return name;
    }
    //Remove the "top" card from the deck and add it to the player's hand.
    public void draw(Deck deck) {
        getHand().add(deck.draw());
    }
     //Return the player's current hand.
    public List<Card> getHand() {
        return hand;
    }
     //Remove and return the "top" card from the player's hand.
    public Card flip() {
        return hand.remove(0);
    }
     //Increment the player's score by 1.
    public void incrementScore() {
        score = getScore() + 1;
    }
     //Return the player's current score.
    public int getScore() {
        return score;
     }
    }
```

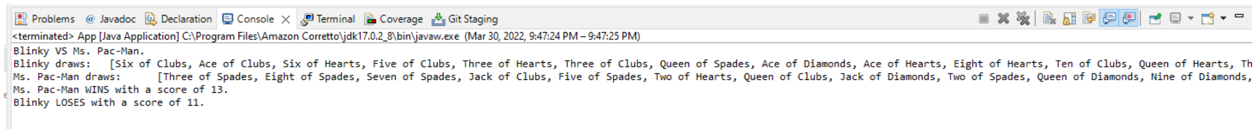


```java
package war_Card_Game;

import java.util.Collections;
import java.util.LinkedList;
import java.util.List;


 // This class represents a card deck. The deck is populated with cards in the constructor.
    public class Deck extends LinkedList<Card> {
        /* The internal list of person names that is used to generate the deck. */

    private final List<String> number = List.of("Two", "Three", "Four", "Five", "Six", "Seven",
      "Eight", "Nine", "Ten", "Jack", "Queen", "King", "Ace");

    //  The internal list of suits that is used to generate the deck.
    private final List<String> suits = List.of("Clubs", "Spades", "Hearts", "Diamonds");

    /*The deck of cards is populated in this constructor. A card is a combination
    of number and suit. Each card also has a rank, which is the position of the number
    element in the number list with 1 added to it, so no card has a 0 value. */
    public Deck() {
        for(int numberPos = 0; numberPos < number.size(); numberPos++) {
            int rank = numberPos + 1;
            String numbers = number.get(numberPos);

            for(String suit : suits) {
                add(new Card(numbers, suit, rank));
      }
    }
  }
    // Shuffle the cards in the deck.

    public void shuffle() {

    Collections.shuffle(this);
    //System.out.println(this); //Printing the shuffled deck for debugging.
  }
    //  Draw and return the "top" card from the deck.
    //  Return The "top" card.
    public Card draw() {
        return remove(0);
    }
}
```



```java
package war_Card_Game;


    //This class represents a Card in the game, War. A Card contains a type name, a suit, and a rank.
    //Cards are created in the Deck constructor.

    public class Card {
        private String number;
        private String suit;
        private int rank;

    /*  Create and initialize a Card object.
    number = value of the card.
    suit = suit of the card.
    rank = the card rank, used for scoring.    */

    public Card(String number, String suit, int rank) {
        this.number = number;
        this.suit = suit;
        this.rank = rank;
    }

    // Returns a String representation of the Card object (Three of Diamonds, King of Clubs, etc)

    @Override
    public String toString() {
        return number + " of " + suit;
    }

    //The rank of the card. Ranks are assigned by the Deck class when the card is created.

    public int getRank() {
        return rank;
    }
}
```

Screenshots of Running Application:



```
<terminated> App [Java Application] C:\Program Files\Amazon Corretto\jdk17.0.2_8\bin\javaw.exe  (Mar 30, 2022, 9:47:24 PM – 9:47:25 PM)
Blinky VS Ms. Pac-Man.
Blinky draws:   [Six of Clubs, Ace of Clubs, Six of Hearts, Five of Clubs, Three of Hearts, Three of Clubs, Queen of Spades, Ace of Diamonds, Ace of Hearts, Eight of Hearts, Ten of Clubs, Queen of Hearts, Th
Ms. Pac-Man draws:      [Three of Spades, Eight of Spades, Seven of Spades, Jack of Clubs, Five of Spades, Two of Hearts, Queen of Clubs, Jack of Diamonds, Two of Spades, Queen of Diamonds, Nine of Diamonds,
Ms. Pac-Man WINS with a score of 13.
Blinky LOSES with a score of 11.
```

URL to GitHub Repository:

https://github.com/dfleeman/Week6CodingAssignment