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

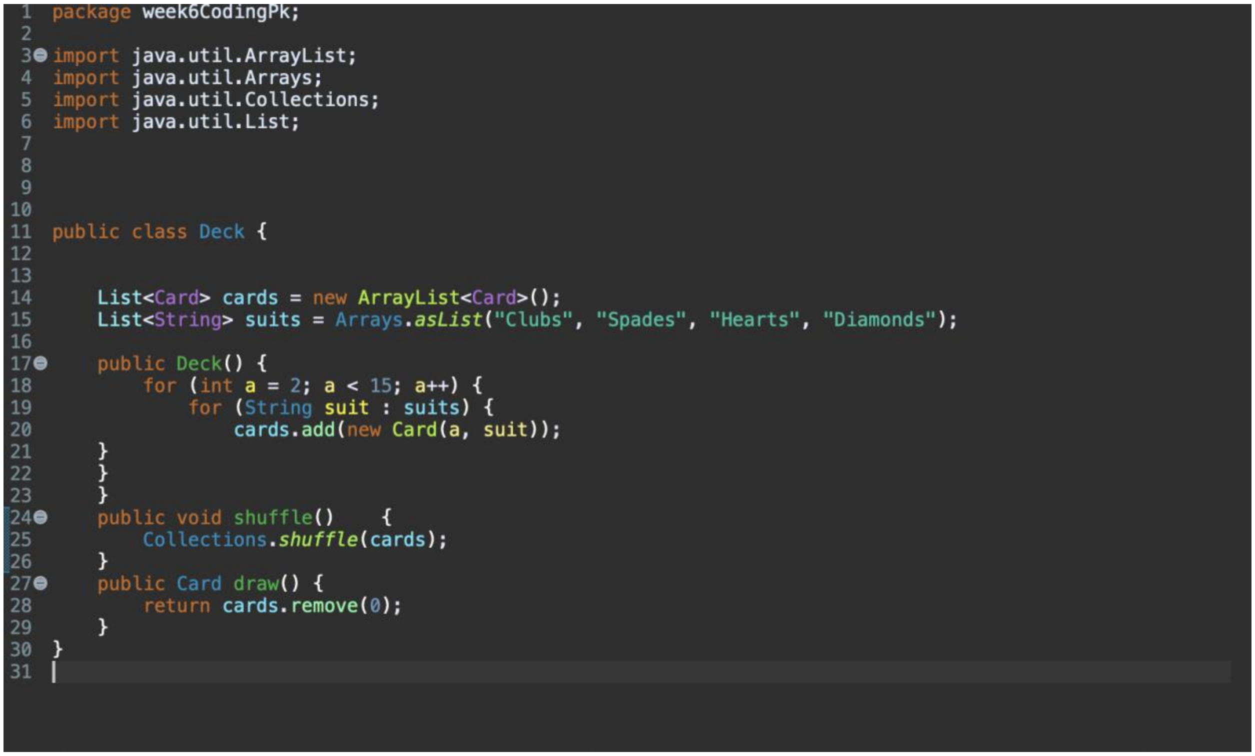
            1. **describe** (prints out information about the player and calls the describe method for each card in the Hand List)
            2. **flip** (removes and returns the top card of the Hand)
            3. **draw** (takes a Deck as an argument and calls the draw method on the deck, adding the returned 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.

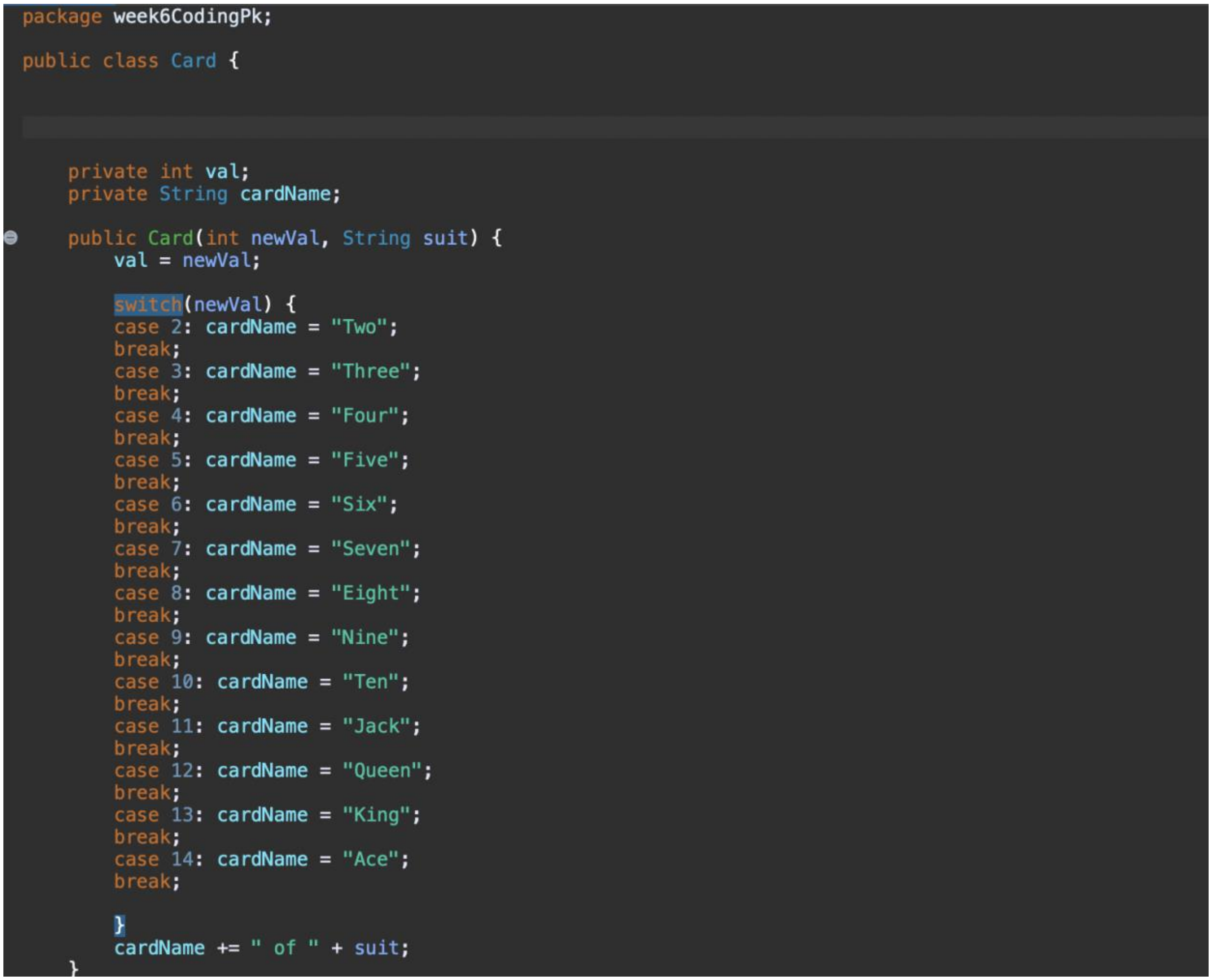
    a. Compare the value of each card returned by the two player's flip methods. Call the incrementScore method on the player whose card has the higher value.

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.

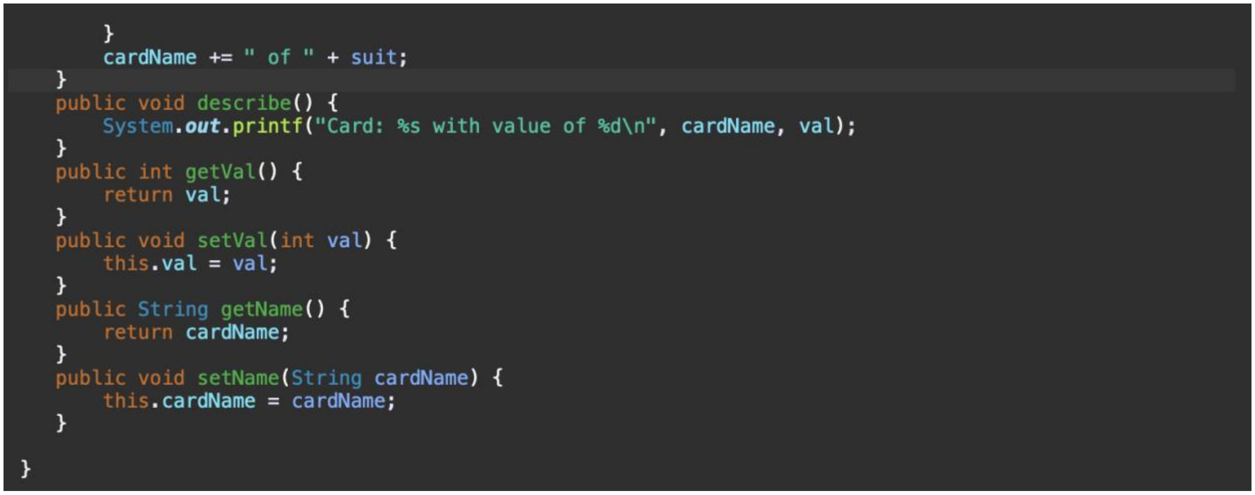
Screenshots of Code:



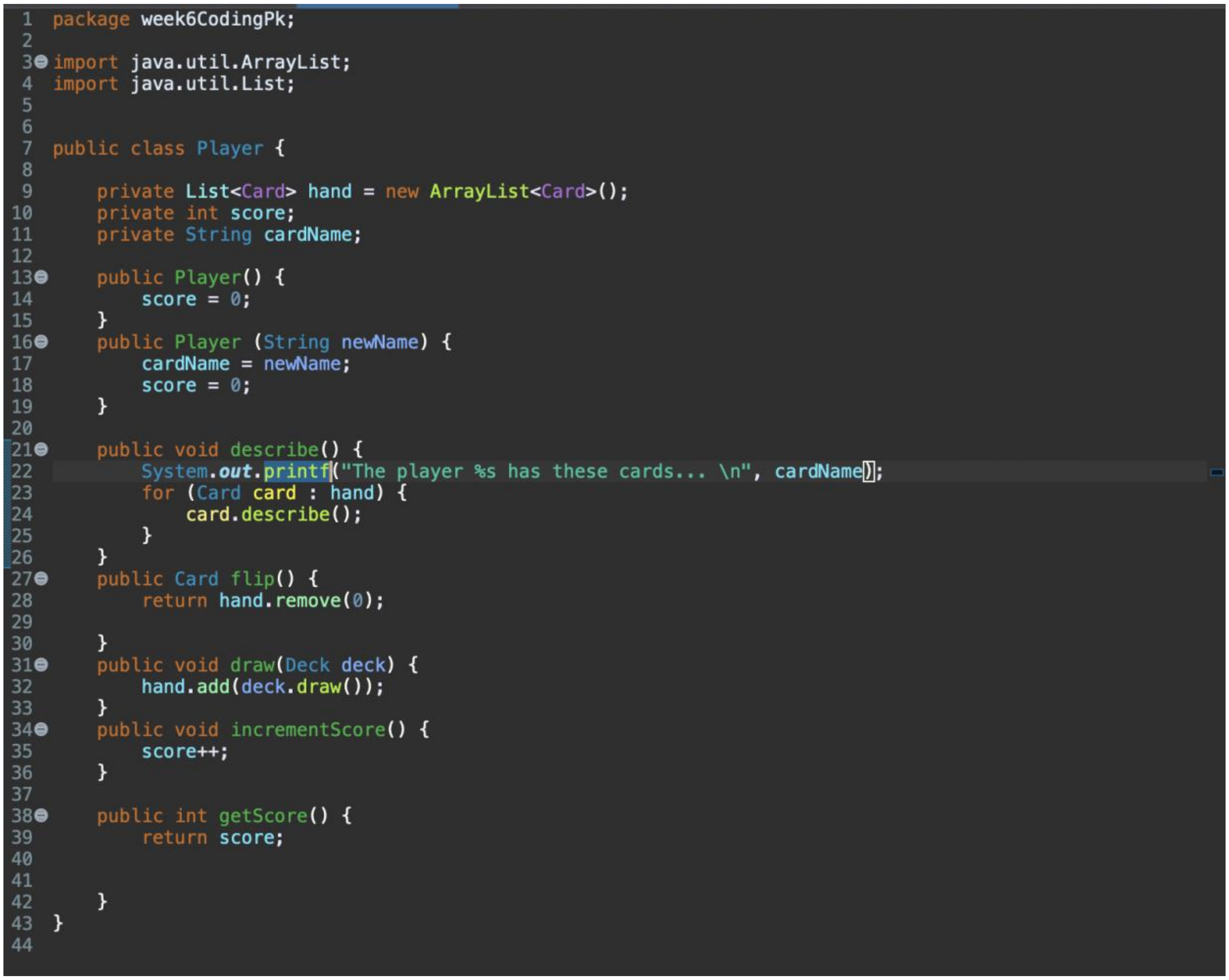
```java
package week6CodingPk;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.List;




public class Deck {


    List<Card> cards = new ArrayList<Card>();
    List<String> suits = Arrays.asList("Clubs", "Spades", "Hearts", "Diamonds");

    public Deck() {
        for (int a = 2; a < 15; a++) {
            for (String suit : suits) {
                cards.add(new Card(a, suit));
    }
    }
    }
    public void shuffle()    {
        Collections.shuffle(cards);
    }
    public Card draw() {
        return cards.remove(0);
    }
}

```



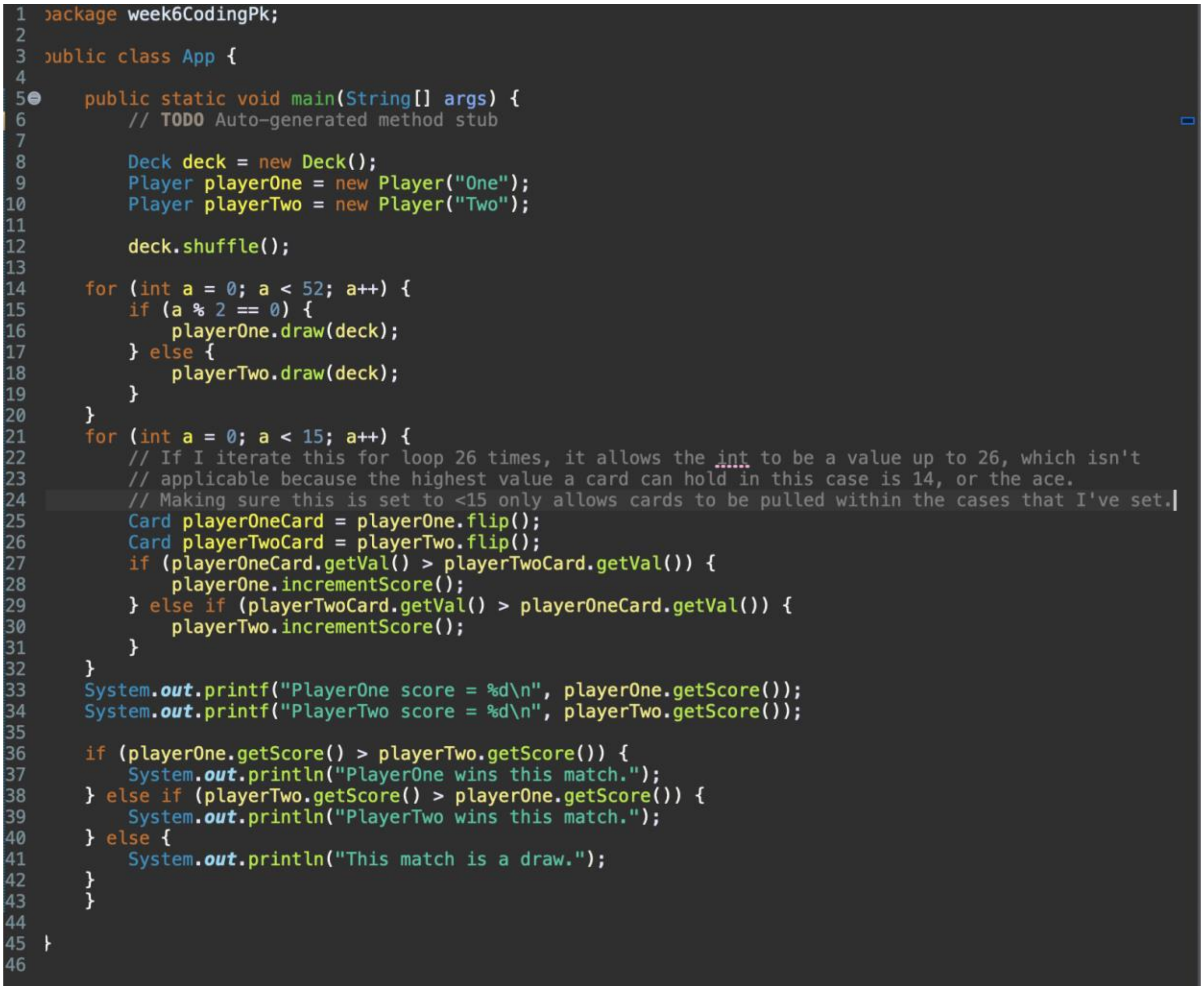
```java
package week6CodingPk;

public class Card {



    private int val;
    private String cardName;

    public Card(int newVal, String suit) {
        val = newVal;

        switch(newVal) {
        case 2: cardName = "Two";
        break;
        case 3: cardName = "Three";
        break;
        case 4: cardName = "Four";
        break;
        case 5: cardName = "Five";
        break;
        case 6: cardName = "Six";
        break;
        case 7: cardName = "Seven";
        break;
        case 8: cardName = "Eight";
        break;
        case 9: cardName = "Nine";
        break;
        case 10: cardName = "Ten";
        break;
        case 11: cardName = "Jack";
        break;
        case 12: cardName = "Queen";
        break;
        case 13: cardName = "King";
        break;
        case 14: cardName = "Ace";
        break;

        }
        cardName += " of " + suit;
    }
```



```java
        }
        cardName += " of " + suit;
    }
    public void describe() {
        System.out.printf("Card: %s with value of %d\n", cardName, val);
    }
    public int getVal() {
        return val;
    }
    public void setVal(int val) {
        this.val = val;
    }
    public String getName() {
        return cardName;
    }
    public void setName(String cardName) {
        this.cardName = cardName;
    }

}
```

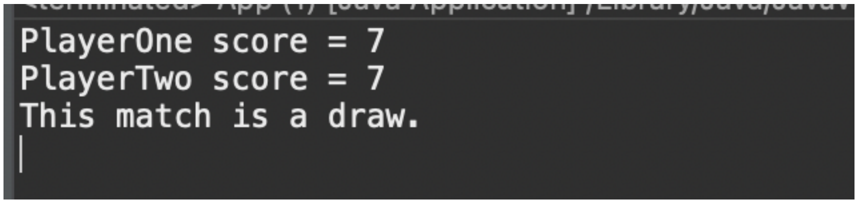


```java
package week6CodingPk;

import java.util.ArrayList;
import java.util.List;


public class Player {

    private List<Card> hand = new ArrayList<Card>();
    private int score;
    private String cardName;

    public Player() {
        score = 0;
    }
    public Player (String newName) {
        cardName = newName;
        score = 0;
    }

    public void describe() {
        System.out.printf("The player %s has these cards... \n", cardName);
        for (Card card : hand) {
            card.describe();
        }
    }
    public Card flip() {
        return hand.remove(0);

    }
    public void draw(Deck deck) {
        hand.add(deck.draw());
    }
    public void incrementScore() {
        score++;
    }

    public int getScore() {
        return score;


    }
}

```

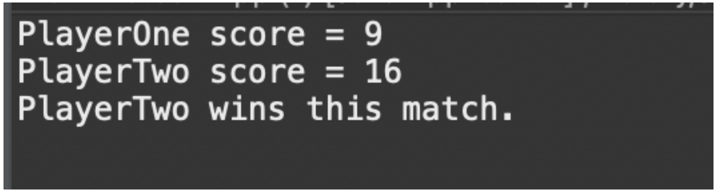


```java
package week6CodingPk;

public class App {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        Deck deck = new Deck();
        Player playerOne = new Player("One");
        Player playerTwo = new Player("Two");

        deck.shuffle();

    for (int a = 0; a < 52; a++) {
        if (a % 2 == 0) {
            playerOne.draw(deck);
        } else {
            playerTwo.draw(deck);
        }
    }
    for (int a = 0; a < 15; a++) {
        // If I iterate this for loop 26 times, it allows the int to be a value up to 26, which isn't
        // applicable because the highest value a card can hold in this case is 14, or the ace.
        // Making sure this is set to <15 only allows cards to be pulled within the cases that I've set.
        Card playerOneCard = playerOne.flip();
        Card playerTwoCard = playerTwo.flip();
        if (playerOneCard.getVal() > playerTwoCard.getVal()) {
            playerOne.incrementScore();
        } else if (playerTwoCard.getVal() > playerOneCard.getVal()) {
            playerTwo.incrementScore();
        }
    }
    System.out.printf("PlayerOne score = %d\n", playerOne.getScore());
    System.out.printf("PlayerTwo score = %d\n", playerTwo.getScore());

    if (playerOne.getScore() > playerTwo.getScore()) {
        System.out.println("PlayerOne wins this match.");
    } else if (playerTwo.getScore() > playerOne.getScore()) {
        System.out.println("PlayerTwo wins this match.");
    } else {
        System.out.println("This match is a draw.");
    }
    }

}
```

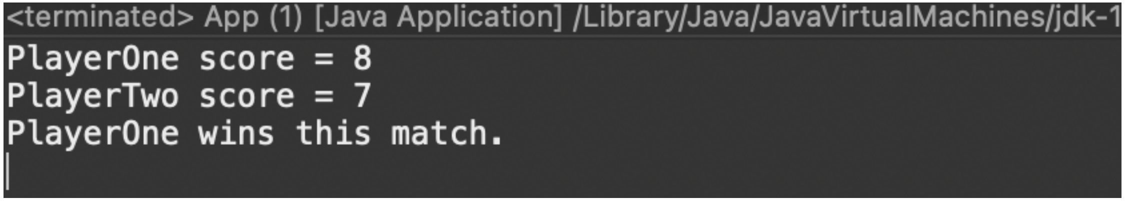
Screenshots of Running Application:



```
PlayerOne score = 7
PlayerTwo score = 7
This match is a draw.
```



```
PlayerOne score = 9
PlayerTwo score = 16
PlayerTwo wins this match.
```



```
<terminated> App (1) [Java Application] /Library/Java/JavaVirtualMachines/jdk-1
PlayerOne score = 8
PlayerTwo score = 7
PlayerOne wins this match.
```

URL to GitHub Repository:

https://github.com/ssbudz123/Week6