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.

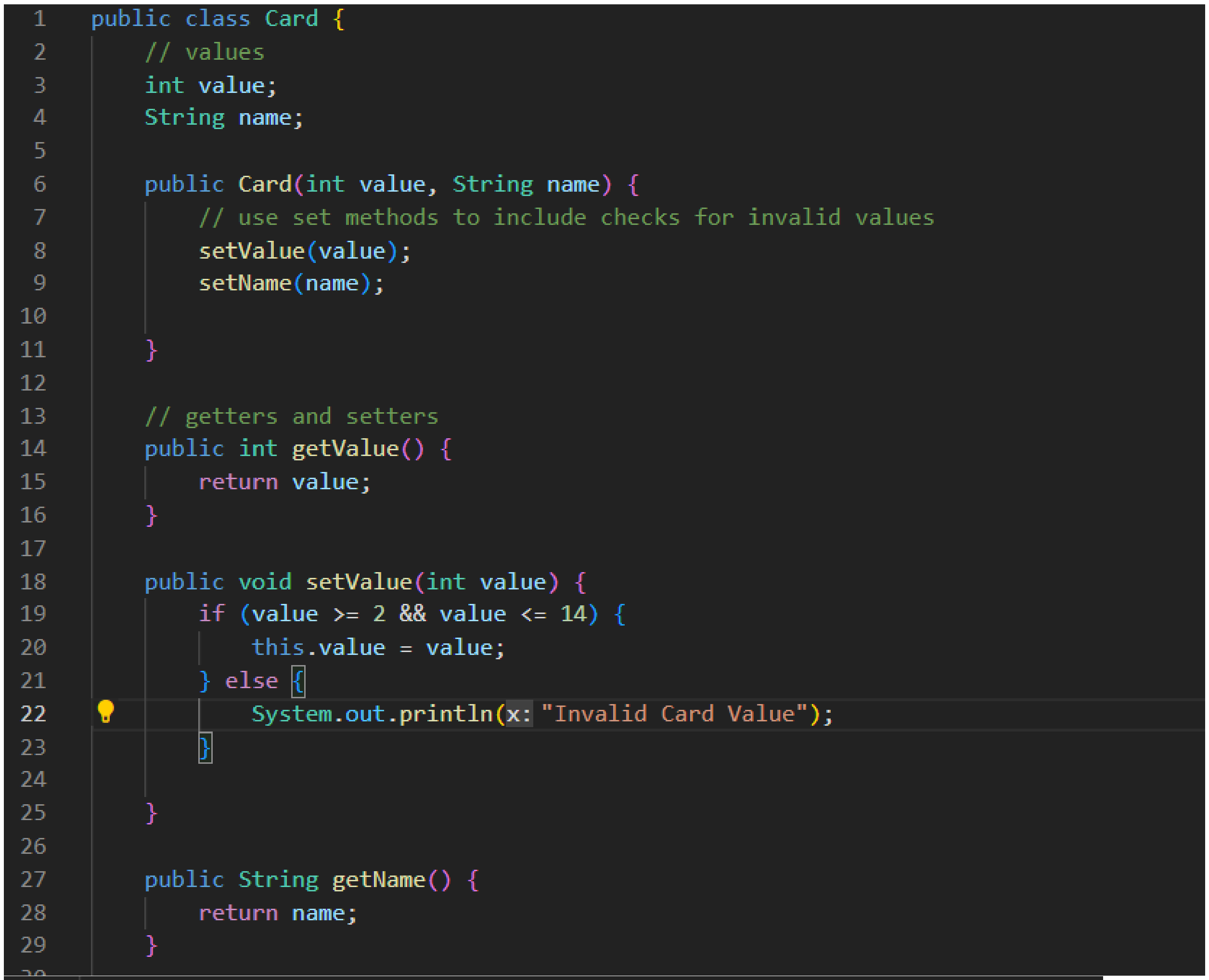
Coding Steps:

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

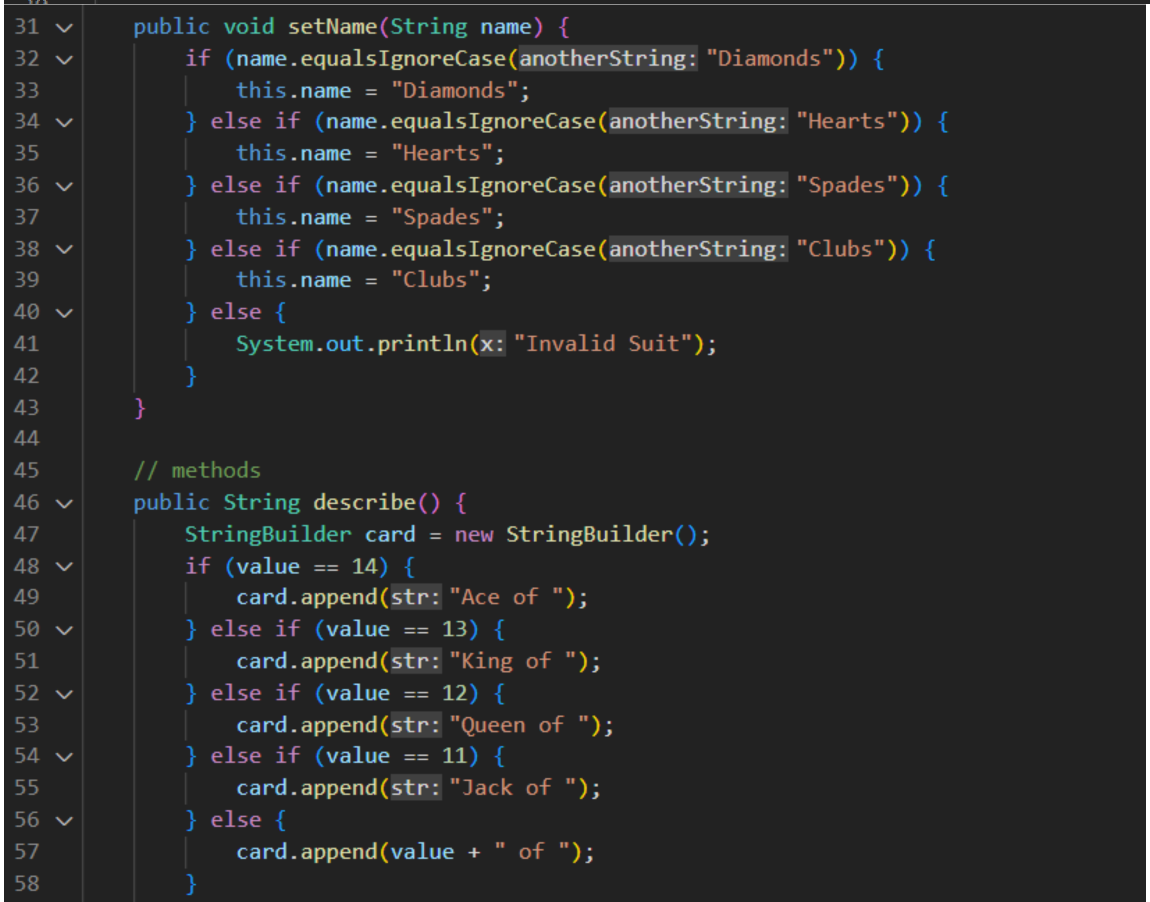
1. Create the following classes.
    a. Card
        i. Fields
            1. **value** (contains a value from 2-14 representing cards 2-Ace)
            2. **name** (e.g. Ace of Diamonds, or Two of Hearts)
        ii. Methods
            1. Getters and Setters
            2. **describe** (prints out 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.

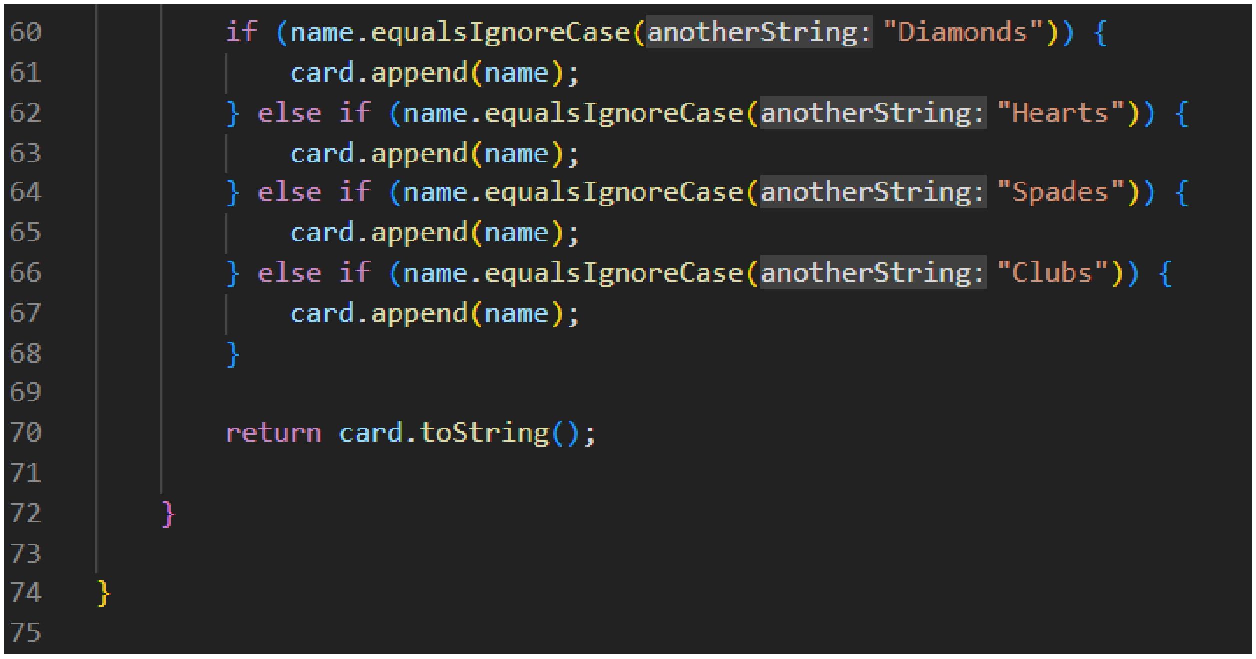
Screenshots of Code:



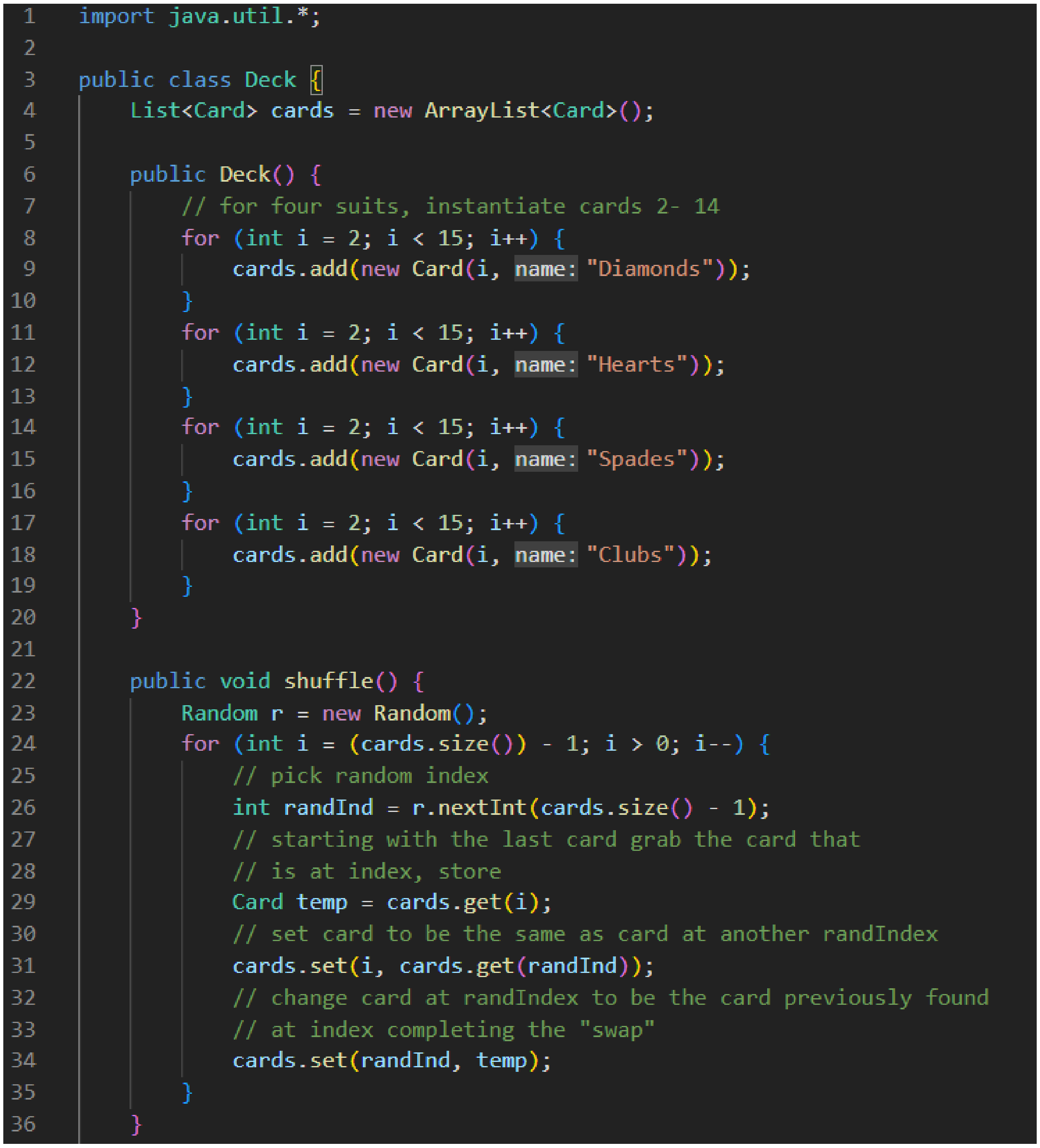
```java
public class Card {
    // values
    int value;
    String name;

    public Card(int value, String name) {
        // use set methods to include checks for invalid values
        setValue(value);
        setName(name);

    }

    // getters and setters
    public int getValue() {
        return value;
    }

    public void setValue(int value) {
        if (value >= 2 && value <= 14) {
            this.value = value;
        } else {
            System.out.println(x: "Invalid Card Value");
        }

    }

    public String getName() {
        return name;
    }
```



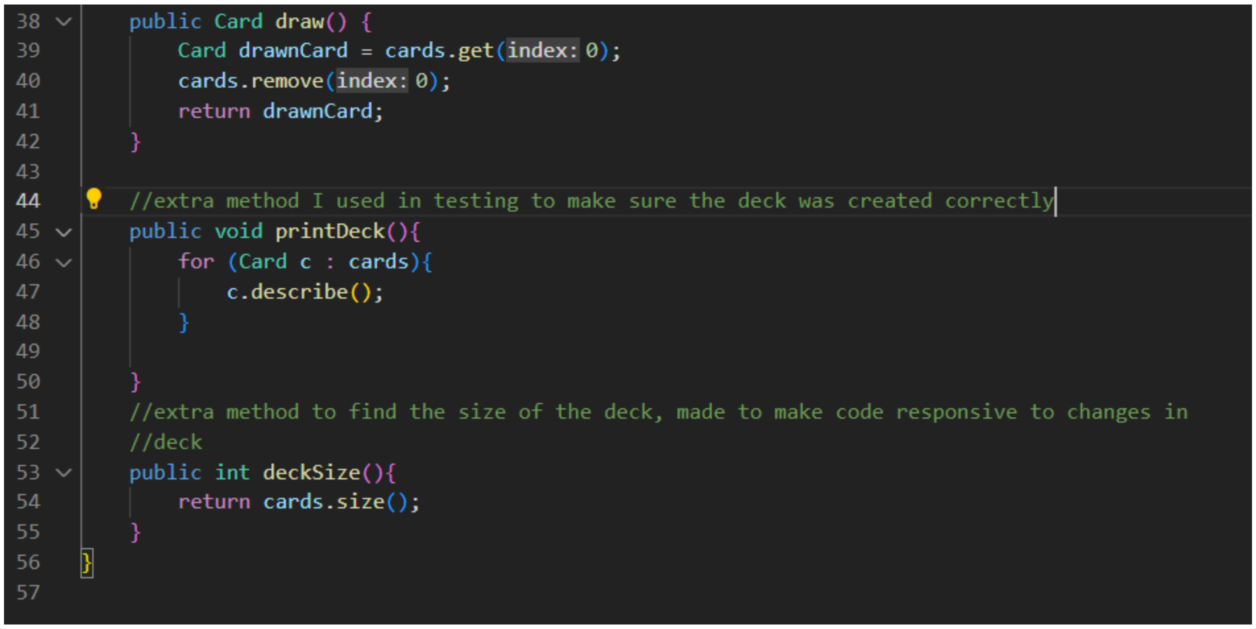
```java
    public void setName(String name) {
        if (name.equalsIgnoreCase(anotherString: "Diamonds")) {
            this.name = "Diamonds";
        } else if (name.equalsIgnoreCase(anotherString: "Hearts")) {
            this.name = "Hearts";
        } else if (name.equalsIgnoreCase(anotherString: "Spades")) {
            this.name = "Spades";
        } else if (name.equalsIgnoreCase(anotherString: "Clubs")) {
            this.name = "Clubs";
        } else {
            System.out.println(x: "Invalid Suit");
        }
    }

    // methods
    public String describe() {
        StringBuilder card = new StringBuilder();
        if (value == 14) {
            card.append(str: "Ace of ");
        } else if (value == 13) {
            card.append(str: "King of ");
        } else if (value == 12) {
            card.append(str: "Queen of ");
        } else if (value == 11) {
            card.append(str: "Jack of ");
        } else {
            card.append(value + " of ");
        }
```



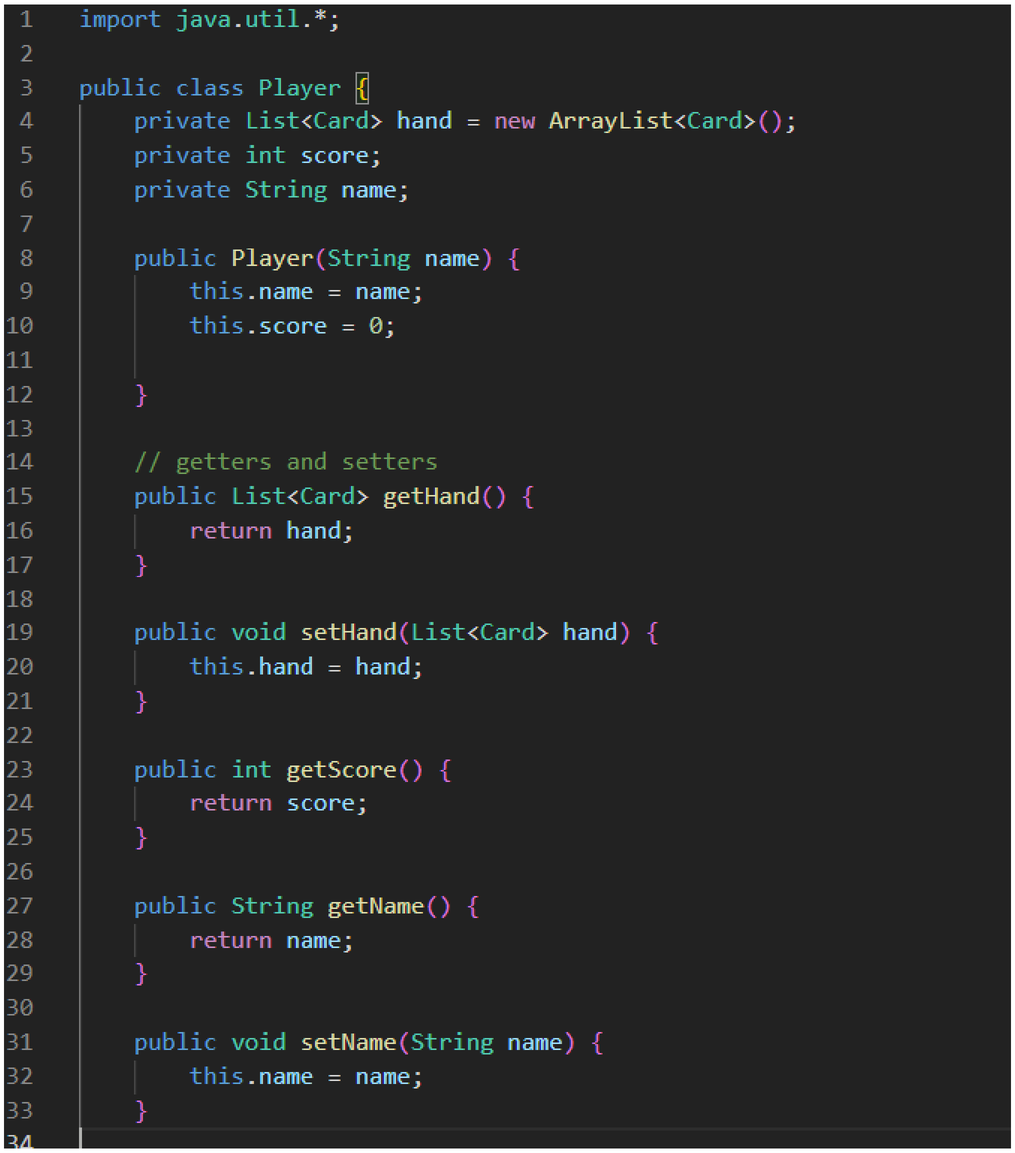
```java
        if (name.equalsIgnoreCase(anotherString: "Diamonds")) {
            card.append(name);
        } else if (name.equalsIgnoreCase(anotherString: "Hearts")) {
            card.append(name);
        } else if (name.equalsIgnoreCase(anotherString: "Spades")) {
            card.append(name);
        } else if (name.equalsIgnoreCase(anotherString: "Clubs")) {
            card.append(name);
        }

        return card.toString();

    }

}

```



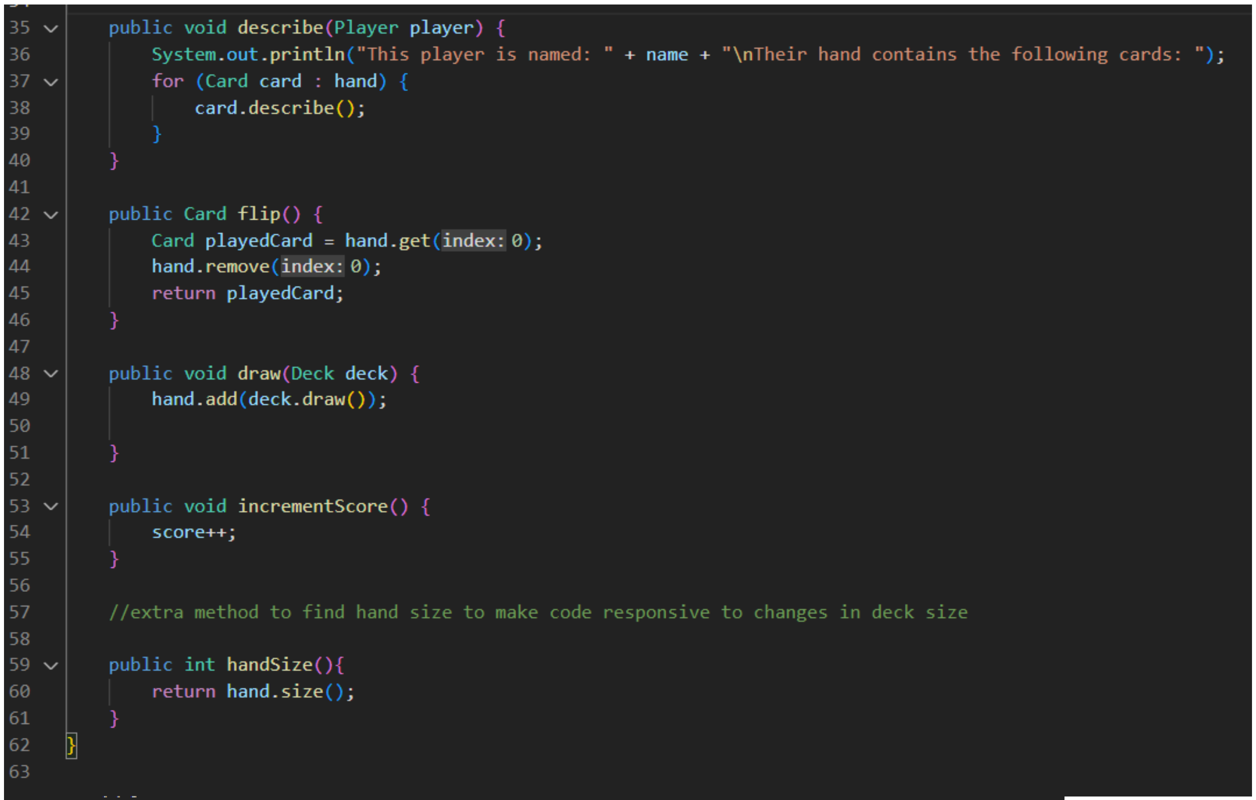
```java
import java.util.*;

public class Deck {
    List<Card> cards = new ArrayList<Card>();

    public Deck() {
        // for four suits, instantiate cards 2- 14
        for (int i = 2; i < 15; i++) {
            cards.add(new Card(i, name: "Diamonds"));
        }
        for (int i = 2; i < 15; i++) {
            cards.add(new Card(i, name: "Hearts"));
        }
        for (int i = 2; i < 15; i++) {
            cards.add(new Card(i, name: "Spades"));
        }
        for (int i = 2; i < 15; i++) {
            cards.add(new Card(i, name: "Clubs"));
        }
    }

    public void shuffle() {
        Random r = new Random();
        for (int i = (cards.size()) - 1; i > 0; i--) {
            // pick random index
            int randInd = r.nextInt(cards.size() - 1);
            // starting with the last card grab the card that
            // is at index, store
            Card temp = cards.get(i);
            // set card to be the same as card at another randIndex
            cards.set(i, cards.get(randInd));
            // change card at randIndex to be the card previously found
            // at index completing the "swap"
            cards.set(randInd, temp);
        }
    }
```



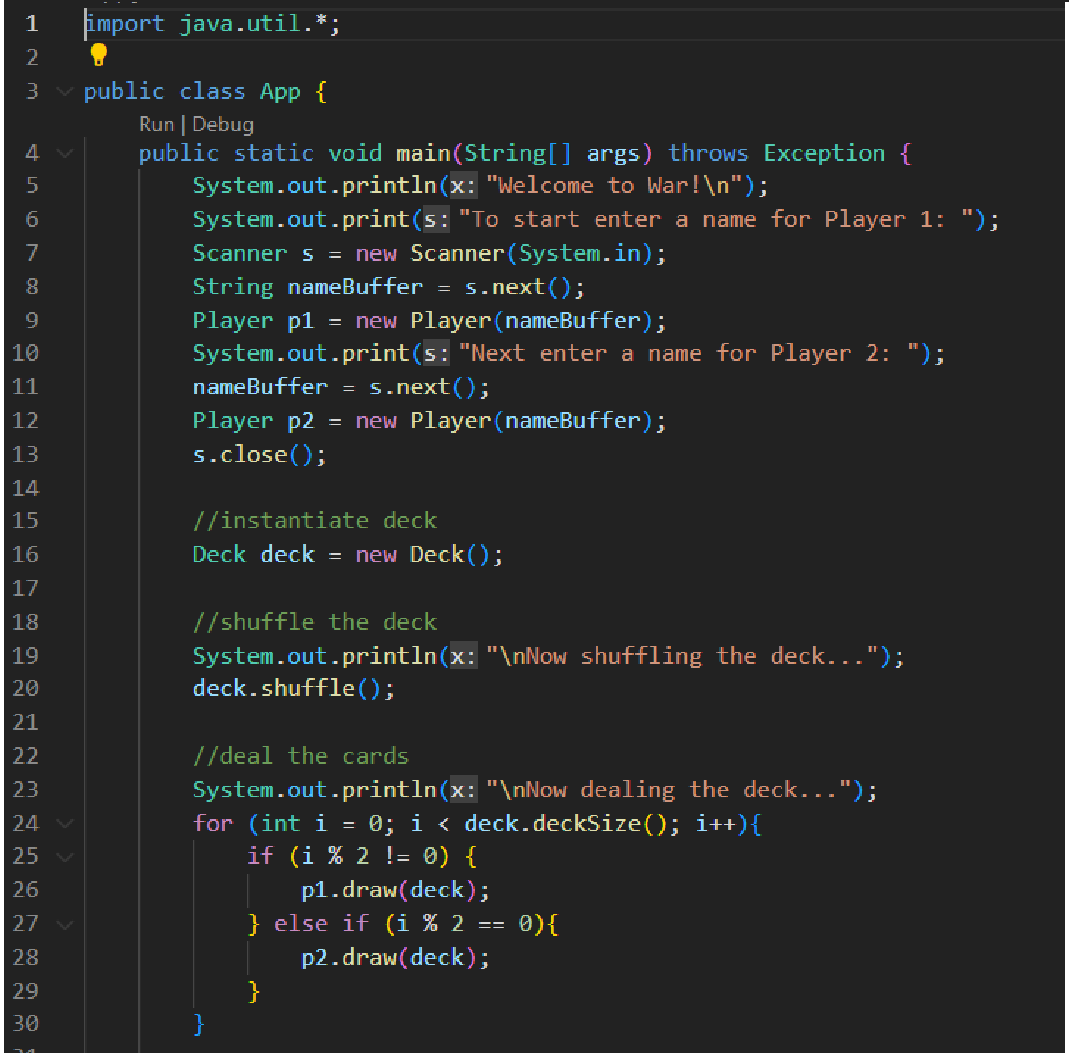
```java
    public Card draw() {
        Card drawnCard = cards.get(index: 0);
        cards.remove(index: 0);
        return drawnCard;
    }

    //extra method I used in testing to make sure the deck was created correctly
    public void printDeck(){
        for (Card c : cards){
            c.describe();
        }

    }
    //extra method to find the size of the deck, made to make code responsive to changes in
    //deck
    public int deckSize(){
        return cards.size();
    }
}

```



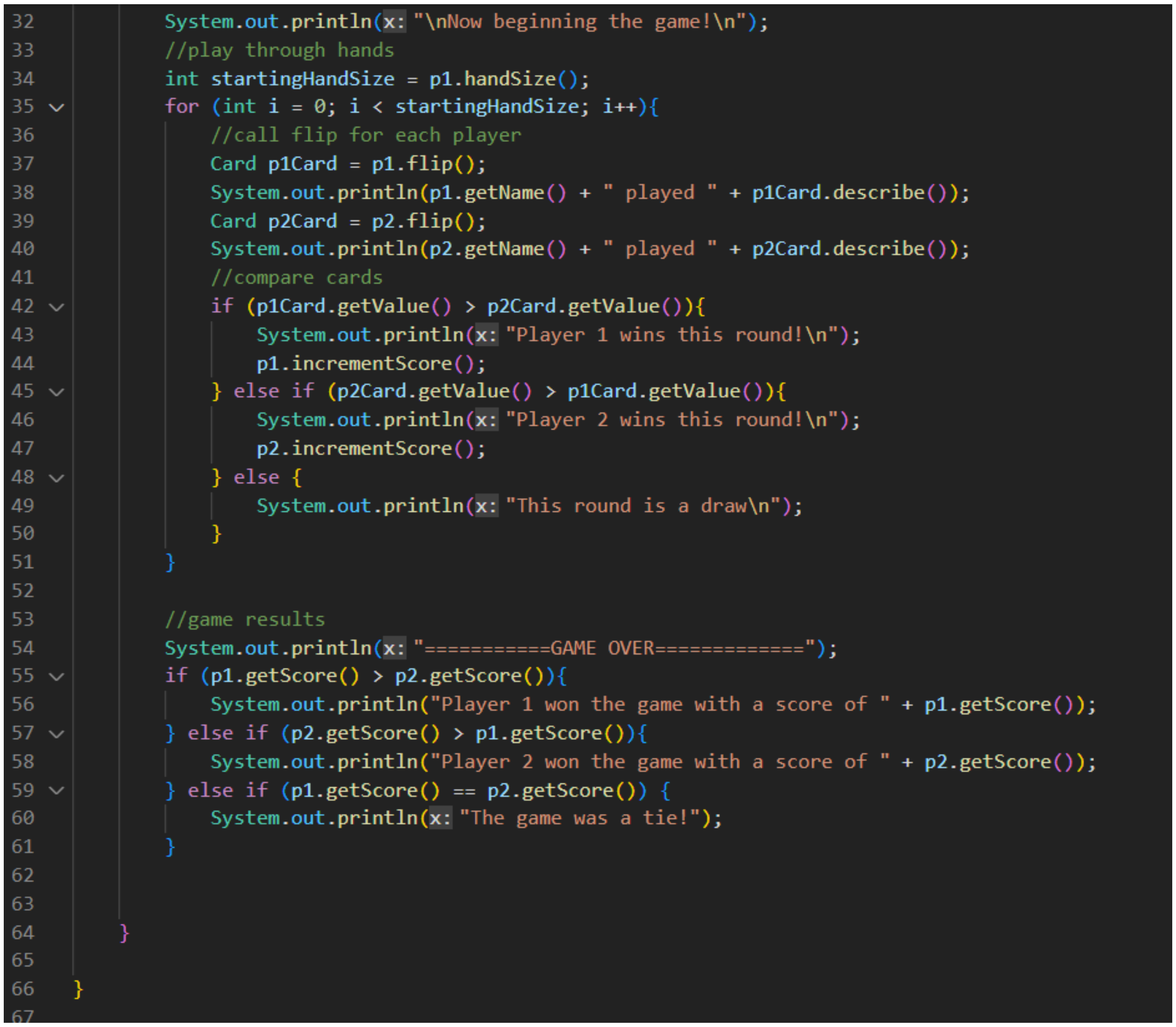
```java
import java.util.*;

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

    public Player(String name) {
        this.name = name;
        this.score = 0;

    }

    // getters and setters
    public List<Card> getHand() {
        return hand;
    }

    public void setHand(List<Card> hand) {
        this.hand = hand;
    }

    public int getScore() {
        return score;
    }

    public String getName() {
        return name;
    }

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

```



```java
    public void describe(Player player) {
        System.out.println("This player is named: " + name + "\nTheir hand contains the following cards: ");
        for (Card card : hand) {
            card.describe();
        }
    }

    public Card flip() {
        Card playedCard = hand.get(index: 0);
        hand.remove(index: 0);
        return playedCard;
    }

    public void draw(Deck deck) {
        hand.add(deck.draw());

    }

    public void incrementScore() {
        score++;
    }

    //extra method to find hand size to make code responsive to changes in deck size

    public int handSize(){
        return hand.size();
    }
}

```

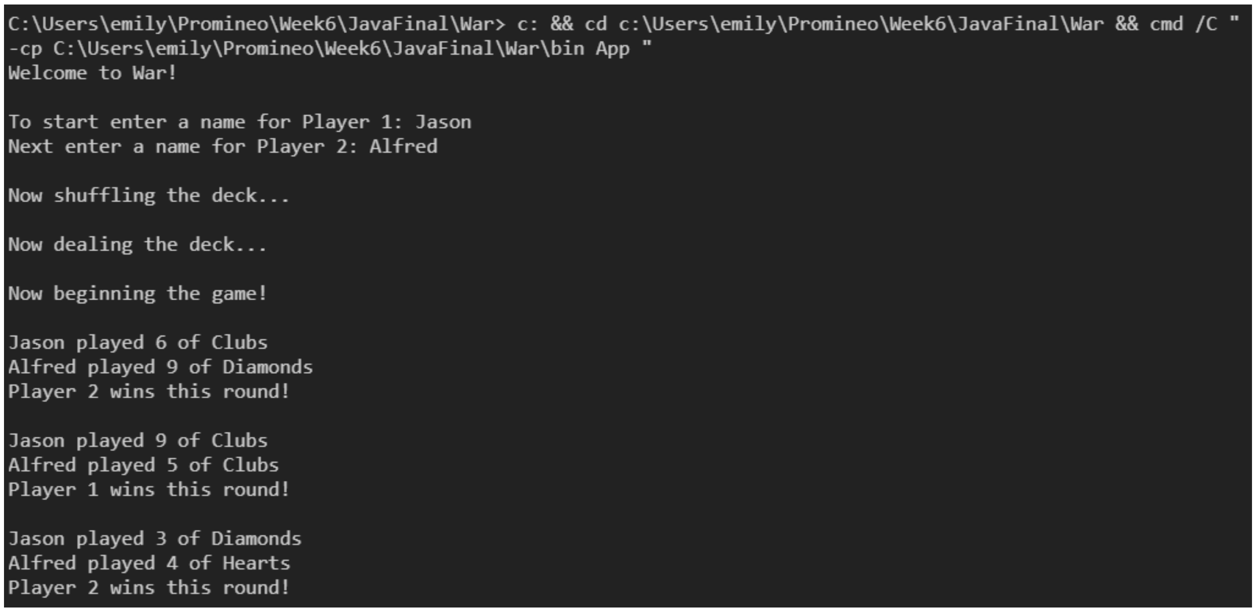


```java
import java.util.*;

public class App {
    Run | Debug
    public static void main(String[] args) throws Exception {
        System.out.println(x: "Welcome to War!\n");
        System.out.print(s: "To start enter a name for Player 1: ");
        Scanner s = new Scanner(System.in);
        String nameBuffer = s.next();
        Player p1 = new Player(nameBuffer);
        System.out.print(s: "Next enter a name for Player 2: ");
        nameBuffer = s.next();
        Player p2 = new Player(nameBuffer);
        s.close();

        //instantiate deck
        Deck deck = new Deck();

        //shuffle the deck
        System.out.println(x: "\nNow shuffling the deck...");
        deck.shuffle();

        //deal the cards
        System.out.println(x: "\nNow dealing the deck...");
        for (int i = 0; i < deck.deckSize(); i++){
            if (i % 2 != 0) {
                p1.draw(deck);
            } else if (i % 2 == 0){
                p2.draw(deck);
            }
        }
```

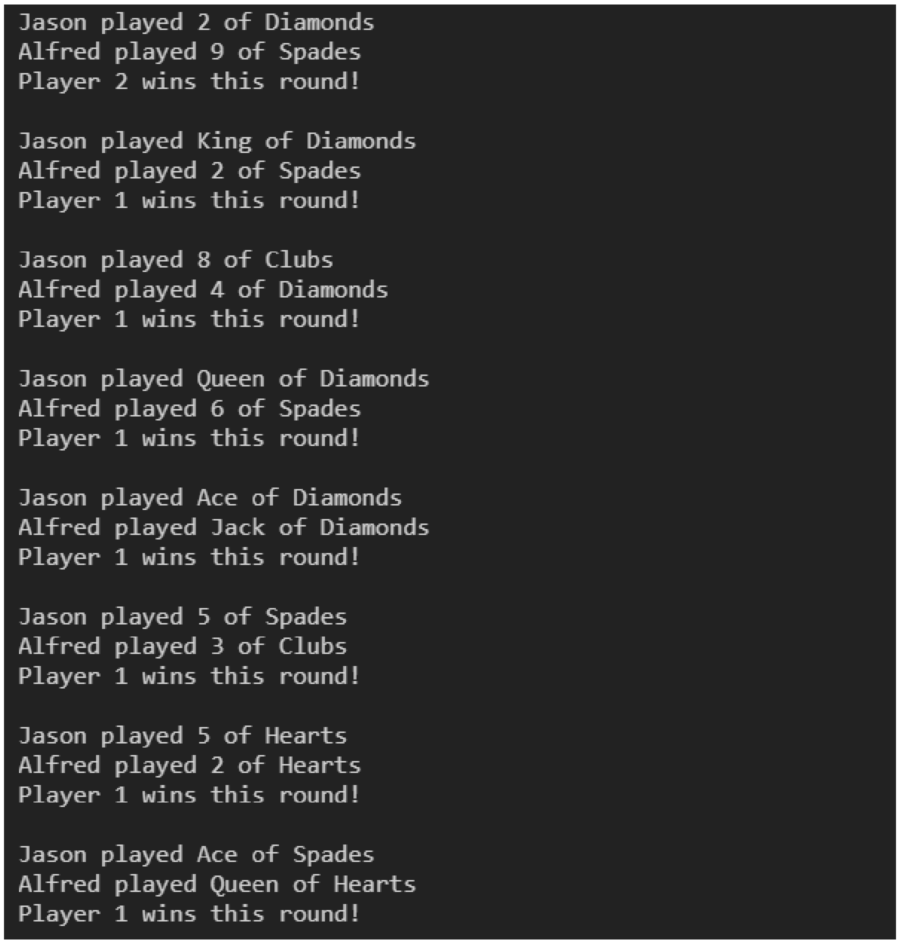


```java
        System.out.println(x: "\nNow beginning the game!\n");
        //play through hands
        int startingHandSize = p1.handSize();
        for (int i = 0; i < startingHandSize; i++){
            //call flip for each player
            Card p1Card = p1.flip();
            System.out.println(p1.getName() + " played " + p1Card.describe());
            Card p2Card = p2.flip();
            System.out.println(p2.getName() + " played " + p2Card.describe());
            //compare cards
            if (p1Card.getValue() > p2Card.getValue()){
                System.out.println(x: "Player 1 wins this round!\n");
                p1.incrementScore();
            } else if (p2Card.getValue() > p1Card.getValue()){
                System.out.println(x: "Player 2 wins this round!\n");
                p2.incrementScore();
            } else {
                System.out.println(x: "This round is a draw\n");
            }
        }

        //game results
        System.out.println(x: "===========GAME OVER=============");
        if (p1.getScore() > p2.getScore()){
            System.out.println("Player 1 won the game with a score of " + p1.getScore());
        } else if (p2.getScore() > p1.getScore()){
            System.out.println("Player 2 won the game with a score of " + p2.getScore());
        } else if (p1.getScore() == p2.getScore()) {
            System.out.println(x: "The game was a tie!");
        }


    }

}
```

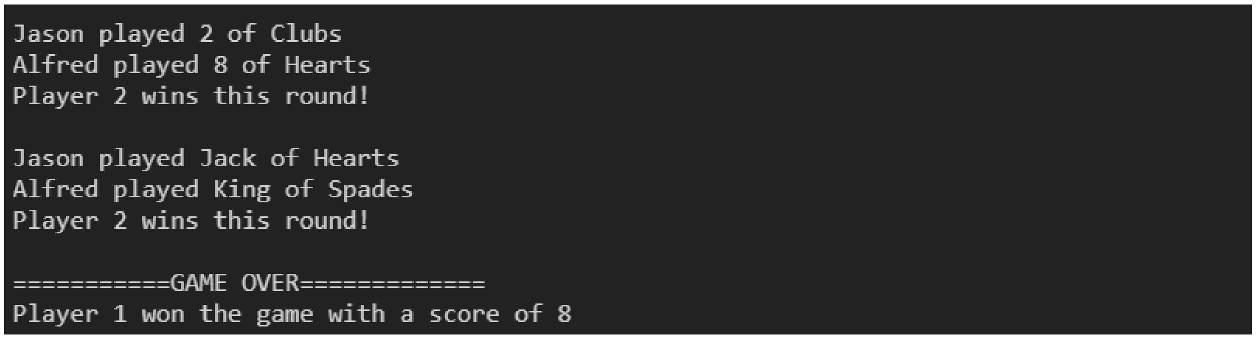
Screenshots of Running Application:



```
C:\Users\emily\Promineo\Week6\JavaFinal\War> c: && cd c:\Users\emily\Promineo\Week6\JavaFinal\War && cmd /C "
-cp C:\Users\emily\Promineo\Week6\JavaFinal\War\bin App "
Welcome to War!

To start enter a name for Player 1: Jason
Next enter a name for Player 2: Alfred

Now shuffling the deck...

Now dealing the deck...

Now beginning the game!

Jason played 6 of Clubs
Alfred played 9 of Diamonds
Player 2 wins this round!

Jason played 9 of Clubs
Alfred played 5 of Clubs
Player 1 wins this round!

Jason played 3 of Diamonds
Alfred played 4 of Hearts
Player 2 wins this round!
```



```
Jason played 2 of Diamonds
Alfred played 9 of Spades
Player 2 wins this round!

Jason played King of Diamonds
Alfred played 2 of Spades
Player 1 wins this round!

Jason played 8 of Clubs
Alfred played 4 of Diamonds
Player 1 wins this round!

Jason played Queen of Diamonds
Alfred played 6 of Spades
Player 1 wins this round!

Jason played Ace of Diamonds
Alfred played Jack of Diamonds
Player 1 wins this round!

Jason played 5 of Spades
Alfred played 3 of Clubs
Player 1 wins this round!

Jason played 5 of Hearts
Alfred played 2 of Hearts
Player 1 wins this round!

Jason played Ace of Spades
Alfred played Queen of Hearts
Player 1 wins this round!
```



```
Jason played 2 of Clubs
Alfred played 8 of Hearts
Player 2 wins this round!

Jason played Jack of Hearts
Alfred played King of Spades
Player 2 wins this round!

===========GAME OVER=============
Player 1 won the game with a score of 8
```

URL to GitHub Repository:

https://github.com/egwalls/JavaFinal