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:

Card Class

```java
package week6CodingProject;

public class Card {
    private int value; //contains a value from 2-14 representing cards 2-Ace
    private String name; //name of the said card

    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;
    }

    public String describe() { //returns String for ease of code use on line 20 of the application
        return "The value is: " + value + " the card name is: " + name;
    }

}

```

Deck Class

```java
package week6CodingProject;

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

public class Deck {

    private String ofHearts = " of Hearts";
    private String ofDiamonds = " of Diamonds";
    private String ofClubs = " of Clubs";
    private String ofSpades = " of Spades";
    private String J = "Jack";
    private String Q = "Queen";
    private String K = "King";
    private String A = "Ace";

    public List<Card> fullDeck = new ArrayList<Card>();

    public Deck() { //when the deck is instantiated, populate the list of Card with the standard 52 cards
        for (Integer i = 2; i <= 14; i++) { //iterates through values 1 through 14
            for (int j = 0; j <= 3; j++) { //iterates through the different suits
                if (i <= 10 && j == 0) {
                    fullDeck.add(new Card(i, i.toString() + ofHearts));
                } else if (i <= 10 && j == 1) {
                    fullDeck.add(new Card(i, i.toString() + ofDiamonds));
                } else if (i <= 10 && j == 2) {
                    fullDeck.add(new Card(i, i.toString() + ofClubs));
                } else if (i <= 10 && j == 3) {
                    fullDeck.add(new Card(i, i.toString() + ofSpades));
                } else if (i == 11 && j == 0) {
                    fullDeck.add(new Card(i, J + ofHearts));
                } else if (i == 11 && j == 1) {
                    fullDeck.add(new Card(i, J + ofDiamonds));
                } else if (i == 11 && j == 2) {
                    fullDeck.add(new Card(i, J + ofClubs));
                } else if (i == 11 && j == 3) {
                    fullDeck.add(new Card(i, J + ofSpades));
                } else if (i == 12 && j == 0) {
                    fullDeck.add(new Card(i, Q + ofHearts));
                } else if (i == 12 && j == 1) {
                    fullDeck.add(new Card(i, Q + ofDiamonds));
                } else if (i == 12 && j == 2) {
                    fullDeck.add(new Card(i, Q + ofClubs));
                } else if (i == 12 && j == 3) {
                    fullDeck.add(new Card(i, Q + ofSpades));
                } else if (i == 13 && j == 0) {
                    fullDeck.add(new Card(i, K + ofHearts));
                } else if (i == 13 && j == 1) {
                    fullDeck.add(new Card(i, K + ofDiamonds));
                } else if (i == 13 && j == 2) {
                    fullDeck.add(new Card(i, K + ofClubs));
                } else if (i == 13 && j == 3) {
                    fullDeck.add(new Card(i, K + ofSpades));
                } else if (i == 14 && j == 0) {
                    fullDeck.add(new Card(i, A + ofHearts));
                } else if (i == 14 && j == 1) {
                    fullDeck.add(new Card(i, A + ofDiamonds));
                } else if (i == 14 && j == 2) {
                    fullDeck.add(new Card(i, A + ofClubs));
                } else if (i == 14 && j == 3) {
                    fullDeck.add(new Card(i, A + ofSpades));
                }
            }
        }
    }
```

```
                }
            }
        }
    }

    public void shuffle() { //shuffles the cards in the deck
        Collections.shuffle(fullDeck);
    }

    public void deckCheck() { //prints out the cards in the deck. Can be use to check if the deck was instantiated and/or shuffled
        for (Card card : fullDeck) {
            System.out.println(card.describe());
        }
    }

    public Card draw() { //draws the top card from the deck
        if (fullDeck.isEmpty()) { //check if a card is still in the deck else returns null
            return null;
        } else {
            Card drawnCard = fullDeck.get(0); //card object to hold the card that is drawn from the top
            fullDeck.remove(0);
            return drawnCard;
        }
    }
}

```

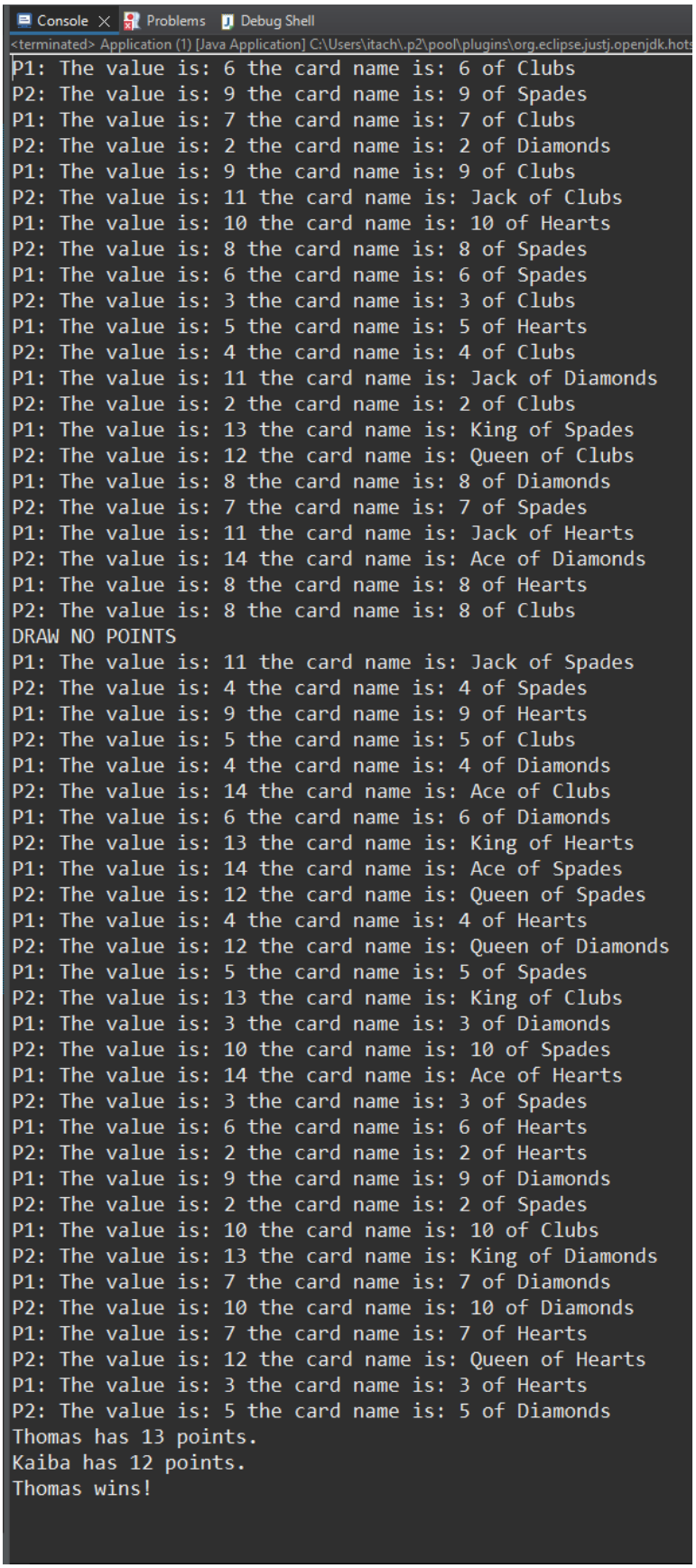
Player Class

```
package week6CodingProject;

import java.util.ArrayList;

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

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

    public void describe() { //describes each card in the player's hand and the name of the player
        for (Card card : hand) {
            card.describe();
        }
        System.out.println(name + " has " + score + " points.");
    }

    public Card flip() {
        Card drawnTopCard = hand.get(0);
        hand.remove(0);
        return drawnTopCard;
    }

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

    public void incrementScore() {
        this.score++;
    }

}
```

Application

```java
package week6CodingProject;

public class Application {
    static Deck deckOfCards = new Deck();
    public static void main(String[] args) {
        Deck deckOfCards = new Deck();
        Player Player1 = new Player("Thomas");
        Player Player2 = new Player("Kaiba");

        deckOfCards.shuffle();

        for (int i = 1; i <= 26; i++) { //each player draws their hand from the deck
            Player1.draw(deckOfCards);
            Player2.draw(deckOfCards);
        }

        for (int i = 1; i <= 26; i++) { //each player flips a card from their hand
            Card P1Card = Player1.flip();
            Card P2Card = Player2.flip();
            System.out.println("P1: " + P1Card.describe() + "\nP2: " + P2Card.describe()); //compares the flipped cards
            if (P1Card.getValue() > P2Card.getValue()) { //if P1's hand value > P2's hand value, P1 gets the point
                Player1.incrementScore();
            } else if (P2Card.getValue() > P1Card.getValue()) { //if P2's hand value > P1's hand value, P2 gets the point
                Player2.incrementScore();
            } else {
                System.out.println("DRAW NO POINTS"); //draw no points
            }
        }

        if (Player1.score > Player2.score) { //prints out the results depending on who has the higher score
            Player1.describe();
            Player2.describe();
            System.out.println(Player1.name + " wins!");
        } else if (Player2.score > Player1.score) {
            Player1.describe();
            Player2.describe();
            System.out.println(Player2.name + " wins!");
        } else {
            Player1.describe();
            Player2.describe();
            System.out.println("It's a draw!");
        }
    }
}

```

Screenshots of Running Application:


Console
Problems
Debug Shell
<terminated> Application (1) [Java Application] C:\Users\itach\.p2\pool\plugins\org.eclipse.justj.openjdk.hot
P1: The value is: 6 the card name is: 6 of Clubs
P2: The value is: 9 the card name is: 9 of Spades
P1: The value is: 7 the card name is: 7 of Clubs
P2: The value is: 2 the card name is: 2 of Diamonds
P1: The value is: 9 the card name is: 9 of Clubs
P2: The value is: 11 the card name is: Jack of Clubs
P1: The value is: 10 the card name is: 10 of Hearts
P2: The value is: 8 the card name is: 8 of Spades
P1: The value is: 6 the card name is: 6 of Spades
P2: The value is: 3 the card name is: 3 of Clubs
P1: The value is: 5 the card name is: 5 of Hearts
P2: The value is: 4 the card name is: 4 of Clubs
P1: The value is: 11 the card name is: Jack of Diamonds
P2: The value is: 2 the card name is: 2 of Clubs
P1: The value is: 13 the card name is: King of Spades
P2: The value is: 12 the card name is: Queen of Clubs
P1: The value is: 8 the card name is: 8 of Diamonds
P2: The value is: 7 the card name is: 7 of Spades
P1: The value is: 11 the card name is: Jack of Hearts
P2: The value is: 14 the card name is: Ace of Diamonds
P1: The value is: 8 the card name is: 8 of Hearts
P2: The value is: 8 the card name is: 8 of Clubs
DRAW NO POINTS
P1: The value is: 11 the card name is: Jack of Spades
P2: The value is: 4 the card name is: 4 of Spades
P1: The value is: 9 the card name is: 9 of Hearts
P2: The value is: 5 the card name is: 5 of Clubs
P1: The value is: 4 the card name is: 4 of Diamonds
P2: The value is: 14 the card name is: Ace of Clubs
P1: The value is: 6 the card name is: 6 of Diamonds
P2: The value is: 13 the card name is: King of Hearts
P1: The value is: 14 the card name is: Ace of Spades
P2: The value is: 12 the card name is: Queen of Spades
P1: The value is: 4 the card name is: 4 of Hearts
P2: The value is: 12 the card name is: Queen of Diamonds
P1: The value is: 5 the card name is: 5 of Spades
P2: The value is: 13 the card name is: King of Clubs
P1: The value is: 3 the card name is: 3 of Diamonds
P2: The value is: 10 the card name is: 10 of Spades
P1: The value is: 14 the card name is: Ace of Hearts
P2: The value is: 3 the card name is: 3 of Spades
P1: The value is: 6 the card name is: 6 of Hearts
P2: The value is: 2 the card name is: 2 of Hearts
P1: The value is: 9 the card name is: 9 of Diamonds
P2: The value is: 2 the card name is: 2 of Spades
P1: The value is: 10 the card name is: 10 of Clubs
P2: The value is: 13 the card name is: King of Diamonds
P1: The value is: 7 the card name is: 7 of Diamonds
P2: The value is: 10 the card name is: 10 of Diamonds
P1: The value is: 7 the card name is: 7 of Hearts
P2: The value is: 12 the card name is: Queen of Hearts
P1: The value is: 3 the card name is: 3 of Hearts
P2: The value is: 5 the card name is: 5 of Diamonds
Thomas has 13 points.
Kaiba has 12 points.
Thomas wins!

URL to GitHub Repository:

https://github.com/AkemiTCGyt/PromineoTechWeek6FinalCodingProject