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*.

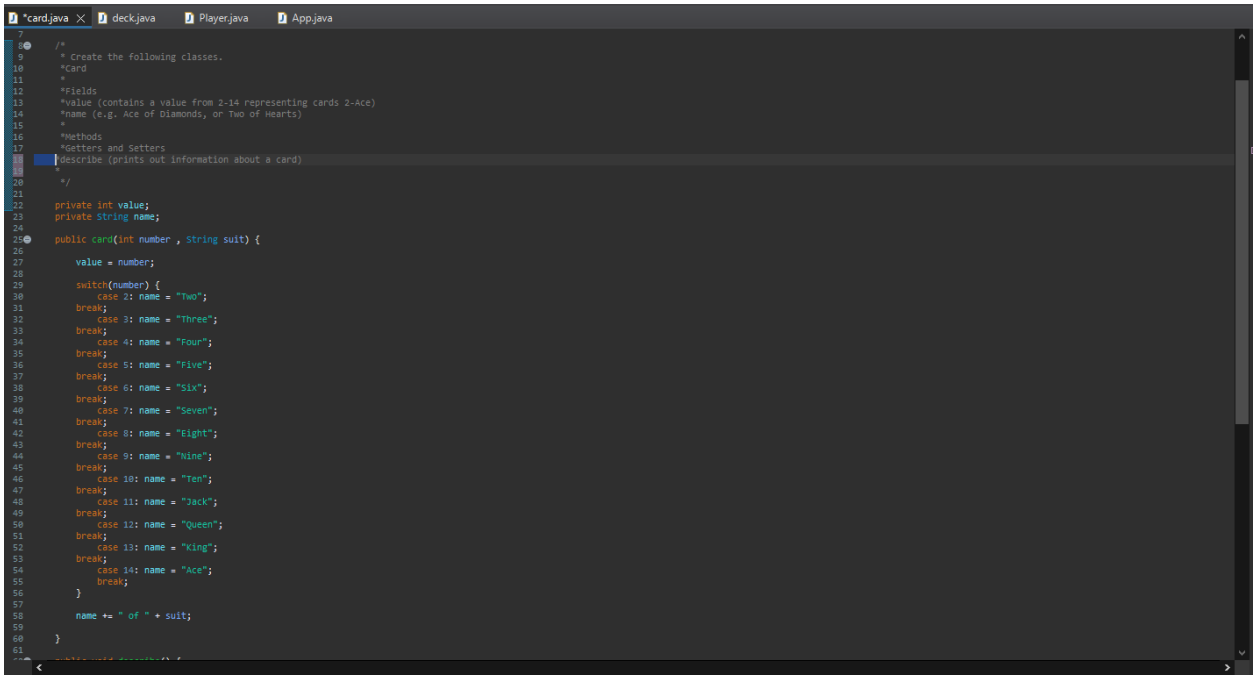
1. Create the following classes.
    a. Card
        i. Fields
            1. **value** (contains a value from 2-14 representing cards 2-Ace)
            2. **name** (e.g. Ace of Diamonds, or Two of Hearts)
        ii. Methods
            1. Getters and Setters
            2. **describe** (prints out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




```
/*
 * Create the following classes.
 *Card
 *
 *fields
 *value (contains a value from 2-14 representing cards 2-Ace)
 *name (e.g. Ace of Diamonds, or Two of Hearts)
 *
 *Methods
 *Getters and Setters
 *describe (prints out information about a card)
 *
 */

private int value;
private String name;

public card(int number , String suit) {

    value = number;

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

    name += " of " + suit;

}
```

```
        name += " of " + suit;

    }

    public void describe() {
        System.out.println("Card: " + name  + " with the value of " + value );
    }


    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;
    }



}
```

*card.java | deck.java | Player.java | App.java

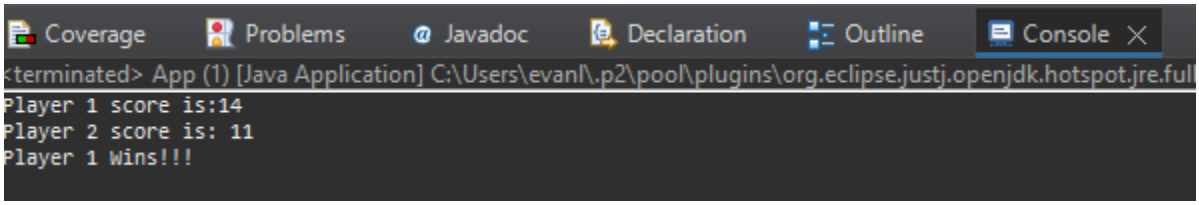
```java
package IntrotoJavaWeek6CodingAssignment;

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

/*
 * Deck
 *
 * Fields
 * cards (List of Card)
 *
 *
 * Methods
 * shuffle (randomizes the order of the cards)
 * draw (removes and returns the top card of the Cards field)
 * In the constructor, when a new Deck is instantiated, the Cards field should be populated with the standard 52 cards.
 */



public class deck {

    List<card> cards = new ArrayList<card>();


    public deck() {
        for (int i = 2; i < 15; i++) {
            cards.add(new card(i, "Diamonds"));
            cards.add(new card(i, "Hearts"));
            cards.add(new card(i, "Clubs"));
            cards.add(new card(i, "Spades"));

        }
    }
    public void shuffle() {
        Collections.shuffle(cards);
    }

    public card draw() {
        return cards.remove(0);
    }
}

```

*card.java | deck.java | Player.java | App.java

```java
 * Player
 * Fields
 * hand (List of Card)
 * score (set to 0 in the constructor)
 * name
 *
 * Methods
 * describe (prints out information about the player and calls the describe method for each card in the Hand List)
 * flip (removes and returns the top card of the Hand)
 * draw (takes a Deck as an argument and calls the draw method on the deck, adding the returned Card to the hand field)
 * incrementScore (adds 1 to the Player's score field)
 */


public class Player {


    private List<card> hand = new ArrayList<card>();
    private int score;
    private String name;

    public Player() {
        score = 0;

    }

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

    public void describe() {
        System.out.println("Player: " + name + " " + "has the following cards:");
        for (card card : hand) {
            card.describe();
        }
    }
    public card flip() {
        return hand.remove(0);

    }

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

    }
    public void addScore() {
        score++;

    }

    public int getscore() {
        return score;
    }
```

```java
card.java  deck.java  Player.java  App.java

/*
 * Create a class called App with a main method.
 * Instantiate a Deck and two Players, call the shuffle method on the deck.
 * Using a traditional for loop, iterate 52 times calling the Draw method on the other player each iteration using the Deck you instantiated.
 * Using a traditional for loop, iterate 26 times and call the flip method for each player.
 * 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.
 * After the loop, compare the final score from each player.
 * 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.
 *
 *
 */

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

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


    for (int i = 0; i < 52; i++) {
        if (i % 2 == 0) {
            player1.draw(deck);
        }else {
            player2.draw(deck);
        }
    }

    for (int i = 0; i <26; i++) {
        card player1card = player1.flip();
        card player2card = player2.flip();
        if(player1card.getvalue() > player2card.getvalue()) {
            player1.addScore();
        }else if (player1card.getvalue() < player2card.getvalue()) {
            player2.addScore();
        }
    }

    System.out.println("Player 1 score is:" + player1.getscore());
    System.out.println("Player 2 score is: " + player2.getscore());
    if(player1.getscore() > player2.getscore()) {
        System.out.println("Player 1 Wins!!! ");
    }else if (player1.getscore() < player2.getscore()) {
        System.out.println("Player 2 Wins!!!");
    }else {
        System.out.println("Its a DRAW!!!!!!!!!");
    }

}
}
```

Screenshots of Running Application:



URL to GitHub Repository:

https://github.com/Evan3167/Intro-to-Java-Week-6-Coding-Assignment-done/tree/main