Intro to Java Week 6 Coding Assignment

Tyler Livermore 4/2/2022

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:

App.java

```java
package week6FinalProject;

import java.util.Scanner;

public class App {

    public static void main(String[] args) {
        Deck deck = new Deck();
        Player player1 = new Player();
        Player player2 = new Player();

        //Receive and set user input for player names
        Scanner enterName = new Scanner(System.in);
        System.out.print("Please Enter Player 1 Name: ");
        String playerName = enterName.next();
        player1.setPlayer(playerName);

        System.out.print("Please Enter Player 2 Name: ");
        playerName = enterName.next();
        player2.setPlayer(playerName);
        enterName.close();
        //scanner closed to avoid leak. Must be after all inputs completed or else exception on the 2nd input b/c the System.in is closed

        deck.shuffle();

        //divvy out the players' hands
        for (int i = 0; i < 52; i++) {
            if (i % 2 == 0) {   //using modulus to alternate draws as even/odd turns
                player1.draw(deck);
            } else if (i % 2 != 0) {
                player2.draw(deck);
            }
        }

        //run each round of War, increment score for each round winner +1
        for (int j = 0; j < 26; j++) {
            int player1Card = player1.flip();
            int player2Card = player2.flip();

            if (player1Card > player2Card) {
                player1.incrementScore();

            } else if (player2Card > player1Card) {
                player2.incrementScore();
            }
        }

        player1.getScore();
        player2.getScore();
        System.out.println(player1.getPlayerName() + ": " + player1.getScore());
        System.out.println(player2.getPlayerName() + ": " + player2.getScore());

        //compare final scores
        if (player1.getScore() > player2.getScore()) {
            System.out.println(player1.getPlayerName() + " wins!");
        } else if (player2.getScore() > player1.getScore()) {
            System.out.println(player2.getPlayerName() + " wins!");
        } else {
            System.out.println("The game ends in a draw.");
        }
    }
}
```

Card.java

```java
package week6FinalProject;

public class Card {
    private int value;      //2-14 (2-Ace)
    private String name;    //Ace of Diamonds,  etc.

    public void setCard(String name, int value) {
        this.name = name;
        this.value = value;
    }

    public int getCardValue() {
        int cardValue = this.value;
        return cardValue;
    }

    public String getCardName() {
        String cardName = this.name;
        return cardName;
    }

    public void describe() {
        System.out.println("(" + getCardValue() + ") " + getCardName());
    }
}
```

Deck.java

```java
package week6FinalProject;

import java.util.ArrayList;

public class Deck {
    private Map <String, Integer> cards;
    private List <String> shuffledCards;

        public Deck() { //create the 52 standard cards
            //Hearts
            Card card1 = new Card();
            card1.setCard("Two of Hearts", 2);
            Card card2 = new Card();
            card2.setCard("Three of Hearts", 3);
            Card card3 = new Card();
            card3.setCard("Four of Hearts", 4);
            Card card4 = new Card();
            card4.setCard("Five of Hearts", 5);
            Card card5 = new Card();
            card5.setCard("Six of Hearts", 6);
            Card card6 = new Card();
            card6.setCard("Seven of Hearts", 7);
            Card card7 = new Card();
            card7.setCard("Eight of Hearts", 8);
            Card card8 = new Card();
            card8.setCard("Nine of Hearts", 9);
            Card card9 = new Card();
            card9.setCard("Ten of Hearts", 10);
            Card card10 = new Card();
            card10.setCard("Jack of Hearts", 11);
            Card card11 = new Card();
            card11.setCard("Queen of Hearts", 12);
            Card card12 = new Card();
            card12.setCard("King of Hearts", 13);
            Card card13 = new Card();
            card13.setCard("Ace of Hearts", 14);
            //Spades
            Card card14 = new Card();
            card14.setCard("Two of Spades", 2);
            Card card15 = new Card();
            card15.setCard("Three of Spades", 3);
            Card card16 = new Card();
            card16.setCard("Four of Spades", 4);
            Card card17 = new Card();
            card17.setCard("Five of Spades", 5);
            Card card18 = new Card();
            card18.setCard("Six of Spades", 6);
            Card card19 = new Card();
            card19.setCard("Seven of Spades", 7);
            Card card20 = new Card();
            card20.setCard("Eight of Spades", 8);
            Card card21 = new Card();
            card21.setCard("Nine of Spades", 9);
            Card card22 = new Card();
            card22.setCard("Ten of Spades", 10);
            Card card23 = new Card();
            card23.setCard("Jack of Spades", 11);
            Card card24 = new Card();
            card24.setCard("Queen of Spades", 12);
            Card card25 = new Card();
            card25.setCard("King of Spades", 13);
            Card card26 = new Card();
            card26.setCard("Ace of Spades", 14);
            //Diamonds
            Card card27 = new Card();
            card27.setCard("Two of Diamonds", 2);
            Card card28 = new Card();
            card28.setCard("Three of Diamonds", 3);
            Card card29 = new Card();
            card29.setCard("Four of Diamonds", 4);
            Card card30 = new Card();
            card30.setCard("Five of Diamonds", 5);
            Card card31 = new Card();
            card31.setCard("Six of Diamonds", 6);
            Card card32 = new Card();
            card32.setCard("Seven of Diamonds", 7);
            Card card33 = new Card();
            card33.setCard("Eight of Diamonds", 8);
            Card card34 = new Card();
            card34.setCard("Nine of Diamonds", 9);
            Card card35 = new Card();
            card35.setCard("Ten of Diamonds", 10);
            Card card36 = new Card();
            card36.setCard("Jack of Diamonds", 11);
            Card card37 = new Card();
            card37.setCard("Queen of Diamonds", 12);
            Card card38 = new Card();
            card38.setCard("King of Diamonds", 13);
            Card card39 = new Card();
            card39.setCard("Ace of Diamonds", 14);
            //Clubs
            Card card40 = new Card();
            card40.setCard("Two of Clubs", 2);
            Card card41 = new Card();
            card41.setCard("Three of Clubs", 3);
            Card card42 = new Card();
            card42.setCard("Four of Clubs", 4);
            Card card43 = new Card();
            card43.setCard("Five of Clubs", 5);
            Card card44 = new Card();
            card44.setCard("Six of Clubs", 6);
            Card card45 = new Card();
            card45.setCard("Seven of Clubs", 7);
            Card card46 = new Card();
            card46.setCard("Eight of Clubs", 8);
            Card card47 = new Card();
            card47.setCard("Nine of Clubs", 9);
            Card card48 = new Card();
            card48.setCard("Ten of Clubs", 10);
            Card card49 = new Card();
            card49.setCard("Jack of Clubs", 11);
            Card card50 = new Card();
            card50.setCard("Queen of Clubs", 12);
            Card card51 = new Card();
            card51.setCard("King of Clubs", 13);
            Card card52 = new Card();
            card52.setCard("Ace of Clubs", 14); //52 = have all the cards

```

```
122             cards = Map.ofEntries(
123                 /*populate a map of the 52 cards, value linked to name(key)
124                 *Map use allows lookup of values by methods later in the program
125                 name is the key b/c of uniqueness vs. duplicate values*/
126                 Map.entry(card1.getCardName(), card1.getCardValue()),
127                 Map.entry(card2.getCardName(), card2.getCardValue()),
128                 Map.entry(card3.getCardName(), card3.getCardValue()),
129                 Map.entry(card4.getCardName(), card4.getCardValue()),
130                 Map.entry(card5.getCardName(), card5.getCardValue()),
131                 Map.entry(card6.getCardName(), card6.getCardValue()),
132                 Map.entry(card7.getCardName(), card7.getCardValue()),
133                 Map.entry(card8.getCardName(), card8.getCardValue()),
134                 Map.entry(card9.getCardName(), card9.getCardValue()),
135                 Map.entry(card10.getCardName(), card10.getCardValue()),
136                 Map.entry(card11.getCardName(), card11.getCardValue()),
137                 Map.entry(card12.getCardName(), card12.getCardValue()),
138                 Map.entry(card13.getCardName(), card13.getCardValue()),
139                 Map.entry(card14.getCardName(), card14.getCardValue()),
140                 Map.entry(card15.getCardName(), card15.getCardValue()),
141                 Map.entry(card16.getCardName(), card16.getCardValue()),
142                 Map.entry(card17.getCardName(), card17.getCardValue()),
143                 Map.entry(card18.getCardName(), card18.getCardValue()),
144                 Map.entry(card19.getCardName(), card19.getCardValue()),
145                 Map.entry(card20.getCardName(), card20.getCardValue()),
146                 Map.entry(card21.getCardName(), card21.getCardValue()),
147                 Map.entry(card22.getCardName(), card22.getCardValue()),
148                 Map.entry(card23.getCardName(), card23.getCardValue()),
149                 Map.entry(card24.getCardName(), card24.getCardValue()),
150                 Map.entry(card25.getCardName(), card25.getCardValue()),
151                 Map.entry(card26.getCardName(), card26.getCardValue()),
152                 Map.entry(card27.getCardName(), card27.getCardValue()),
153                 Map.entry(card28.getCardName(), card28.getCardValue()),
154                 Map.entry(card29.getCardName(), card29.getCardValue()),
155                 Map.entry(card30.getCardName(), card30.getCardValue()),
156                 Map.entry(card31.getCardName(), card31.getCardValue()),
157                 Map.entry(card32.getCardName(), card32.getCardValue()),
158                 Map.entry(card33.getCardName(), card33.getCardValue()),
159                 Map.entry(card34.getCardName(), card34.getCardValue()),
160                 Map.entry(card35.getCardName(), card35.getCardValue()),
161                 Map.entry(card36.getCardName(), card36.getCardValue()),
162                 Map.entry(card37.getCardName(), card37.getCardValue()),
163                 Map.entry(card38.getCardName(), card38.getCardValue()),
164                 Map.entry(card39.getCardName(), card39.getCardValue()),
165                 Map.entry(card40.getCardName(), card40.getCardValue()),
166                 Map.entry(card41.getCardName(), card41.getCardValue()),
167                 Map.entry(card42.getCardName(), card42.getCardValue()),
168                 Map.entry(card43.getCardName(), card43.getCardValue()),
169                 Map.entry(card44.getCardName(), card44.getCardValue()),
170                 Map.entry(card45.getCardName(), card45.getCardValue()),
171                 Map.entry(card46.getCardName(), card46.getCardValue()),
172                 Map.entry(card47.getCardName(), card47.getCardValue()),
173                 Map.entry(card48.getCardName(), card48.getCardValue()),
174                 Map.entry(card49.getCardName(), card49.getCardValue()),
175                 Map.entry(card50.getCardName(), card50.getCardValue()),
176                 Map.entry(card51.getCardName(), card51.getCardValue()),
177                 Map.entry(card52.getCardName(), card52.getCardValue())
178             );
179         }
180
```

```java
    public void shuffle() {
        //Card names from Map moved into a List for calling indexes, values called later from Map by name
        List<String> shuffledCards = new ArrayList<>(cards.keySet());
        this.shuffledCards = shuffledCards;
        Collections.shuffle(shuffledCards); //shuffle method used from imported Collections library
    }
    //select top card of shuffled deck and then remove
    public String draw() {
        String drawnCard = shuffledCards.get(0);
        shuffledCards.remove(0);
        return drawnCard;
    }
    //lookup card value in Map table
    public Integer retrieveValue(String cardName) {
        return cards.get(cardName);
    }
}
```

Player.java

```java
package week6FinalProject;

import java.util.ArrayList;

public class Player {
    private String name;
    private List <String> hand;
    private int score = 0;

    public Player() {
        List<String> hand = new ArrayList<>();
        this.hand = hand;
    }

    public void describe() {
        System.out.println(name);
        System.out.println(score);
    }


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

    public String getPlayerName() {
        return this.name;
    }

    public int getScore() {
        return score;
    }

    //build up the player's cards with loop in Main
    public void draw(Deck deck) {
        hand.add(deck.draw());
    }

    public Integer flip() {
        //select top card of hand and remove after
        String cardFromHand = hand.get(0);
        hand.remove(0);
        Deck deck = new Deck();
        //look up card value by name
        return deck.retrieveValue(cardFromHand);
    }

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

Screenshots of Running Application:

Confirming population of Map deck (was testing with an abbreviated card list):

```
Console  Problems  Declaration
<terminated> App (1) [Java Application] D:\Software Development\Tools\Eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.1.v20211116-1657
{Five of Hearts=5, Seven of Hearts=7, King of Hearts=13, Ten of Hearts=10, Nine of
Hearts=9, Eight of Hearts=8, Six of Hearts=6, Four of Hearts=4, Three of Hearts=3, Two
of Hearts=2, Queen of Hearts=12, Jack of Hearts=11, Ace of Hearts=14}
```

Shuffling the Map deck (as a List):

```
Console  Problems  Declaration
<terminated> App (1) [Java Application] D:\Software Development\Tools\Eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.1.v20211116-1657
{Five of Hearts=5, Ace of Hearts=14, Jack of Hearts=11, Queen of Hearts=12, Two of
Hearts=2, Three of Hearts=3, Four of Hearts=4, Six of Hearts=6, Eight of Hearts=8, Nine
of Hearts=9, Ten of Hearts=10, King of Hearts=13, Seven of Hearts=7}
[Queen of Hearts, Three of Hearts, Ace of Hearts, Nine of Hearts, Eight of Hearts, Four
of Hearts, Two of Hearts, King of Hearts, Ten of Hearts, Six of Hearts, Seven of
Hearts, Jack of Hearts, Five of Hearts]
```

Looking up value of “Two of Hearts” correctly returns the card’s value of 2:

```
Console  Problems  Declaration
<terminated> App (1) [Java Application] D:\Software Development\Tools\Eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.1.v20211116-165
{Two of Hearts=2, Queen of Hearts=12, Jack of Hearts=11, Ace of Hearts=14, Five of
Hearts=5, Seven of Hearts=7, King of Hearts=13, Ten of Hearts=10, Nine of Hearts=9,
Eight of Hearts=8, Six of Hearts=6, Four of Hearts=4, Three of Hearts=3}
[Ace of Hearts, Six of Hearts, Ten of Hearts, Eight of Hearts, Two of Hearts, Three of
Hearts, Seven of Hearts, Jack of Hearts, Queen of Hearts, Five of Hearts, Nine of
Hearts, King of Hearts, Four of Hearts]
2
```

Draw top card and check value:

```
Console  Problems  Declaration
<terminated> App (1) [Java Application] D:\Software Development\Tools\Eclips
Five of Hearts
5
```

Full deck check:

```
Console  Problems  Declaration
<terminated> App (1) [Java Application] D:\Software Development\Tools\Eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.1.v20211116-1657\jre\bin\javaw.exe (Apr 1, 2022,
{Ten of Clubs=10, Six of Spades=6, Ace of Clubs=14, Three of Spades=3, Queen of Spades=12, Two of
Clubs=2, Ace of Spades=14, Six of Diamonds=6, Seven of Spades=7, Two of Diamonds=2, Seven of
Diamonds=7, Jack of Clubs=11, Queen of Clubs=12, Four of Spades=4, Ten of Diamonds=10, Five of
Diamonds=5, Two of Spades=2, Queen of Diamonds=12, Eight of Clubs=8, Jack of Spades=11, Five of
Hearts=5, Five of Clubs=5, Ten of Hearts=10, Eight of Hearts=8, King of Spades=13, Three of Diamonds=3,
Six of Hearts=6, Nine of Spades=9, Four of Diamonds=4, Jack of Diamonds=11, Three of Hearts=3, King of
Diamonds=13, Seven of Clubs=7, Eight of Diamonds=8, Queen of Hearts=12, Nine of Diamonds=9, Nine of
Clubs=9, Ace of Hearts=14, Seven of Hearts=7, Four of Clubs=4, Ace of Diamonds=14, Four of Hearts=4,
Two of Hearts=2, Six of Clubs=6, Jack of Hearts=11, King of Hearts=13, Nine of Hearts=9, Five of
Spades=5, Ten of Spades=10, King of Clubs=13, Eight of Spades=8, Three of Clubs=3}
Ace of Clubs
14
```

Checking that each Player's hand was filled with their 26 cards:

```
Console  Problems  Declaration
<terminated> App (1) [Java Application] D:\Software Development\Tools\Eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.1.v20211116-1657\jre\bin\javaw.exe (Apr 1, 2022, 4:01:
James
[Ten of Hearts, Jack of Diamonds, Three of Diamonds, Eight of Diamonds, Two of Spades, Eight of Spades,
King of Hearts, Nine of Diamonds, Ace of Diamonds, King of Clubs, Queen of Hearts, Four of Clubs, Two of
Diamonds, Five of Hearts, Nine of Hearts, Four of Hearts, Three of Hearts, Seven of Spades, Six of Clubs,
Six of Spades, Six of Diamonds, Two of Clubs, Jack of Hearts, Seven of Hearts, Three of Clubs, Eight of
Clubs]
0
Mary
[Ace of Hearts, Four of Spades, Queen of Spades, King of Diamonds, Queen of Clubs, Ten of Clubs, Five of
Clubs, Seven of Diamonds, Eight of Hearts, Jack of Spades, Ten of Spades, Ace of Spades, Five of Diamonds,
King of Spades, Queen of Diamonds, Seven of Clubs, Jack of Clubs, Four of Diamonds, Ten of Diamonds, Five
of Spades, Nine of Clubs, Two of Hearts, Six of Hearts, Nine of Spades, Three of Spades, Ace of Clubs]
0
```

Testing flip(), comparison of result, and incrementScore():

```
Console  Problems  Declaration
<terminated> App (1) [Java Application] D:\Software Developme
P1 Card: 10
P2 Card: 14
James's Score: 0
Mary's Score: 1
P1 Card: 13
P2 Card: 3
James's Score: 1
Mary's Score: 1
P1 Card: 3
P2 Card: 5
James's Score: 1
Mary's Score: 2
P1 Card: 6
P2 Card: 14
James's Score: 1
Mary's Score: 3
P1 Card: 4
P2 Card: 8
James's Score: 1
Mary's Score: 4
P1 Card: 8
P2 Card: 2
James's Score: 2
Mary's Score: 4
```

Some outcomes of a completed game:

```
Console  Problems  Declaration
<terminated> App (1) [Java Application] D:\Software Developn
James: 13
Mary: 11
James wins!
```

```
Console  Problems  Declaration
<terminated> App (1) [Java Application] D:\Software Develop
James: 11
Mary: 15
Mary wins!
```

```
Console  Problems  Declaration
<terminated> App (1) [Java Application] D:\Software Development\T
James: 15
Mary: 9
James wins!
```

```
Console  Problems  Declaration
<terminated> App (1) [Java Application] D:\Software Developmer
James: 12
Mary: 12
The game ends in a draw.
```

User input (scanner in) for player names:

```
Console   Problems   Declaration
<terminated> App (1) [Java Application] D:\Software Development\Tools\Eclipse
Please Enter Player 1 Name: Mikey
Please Enter Player 2 Name: Rebecca
Mikey: 13
Rebecca: 9
Mikey wins!
```

URL to GitHub Repository:

https://github.com/tylerjlivermore/Week6_CardsWar