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)

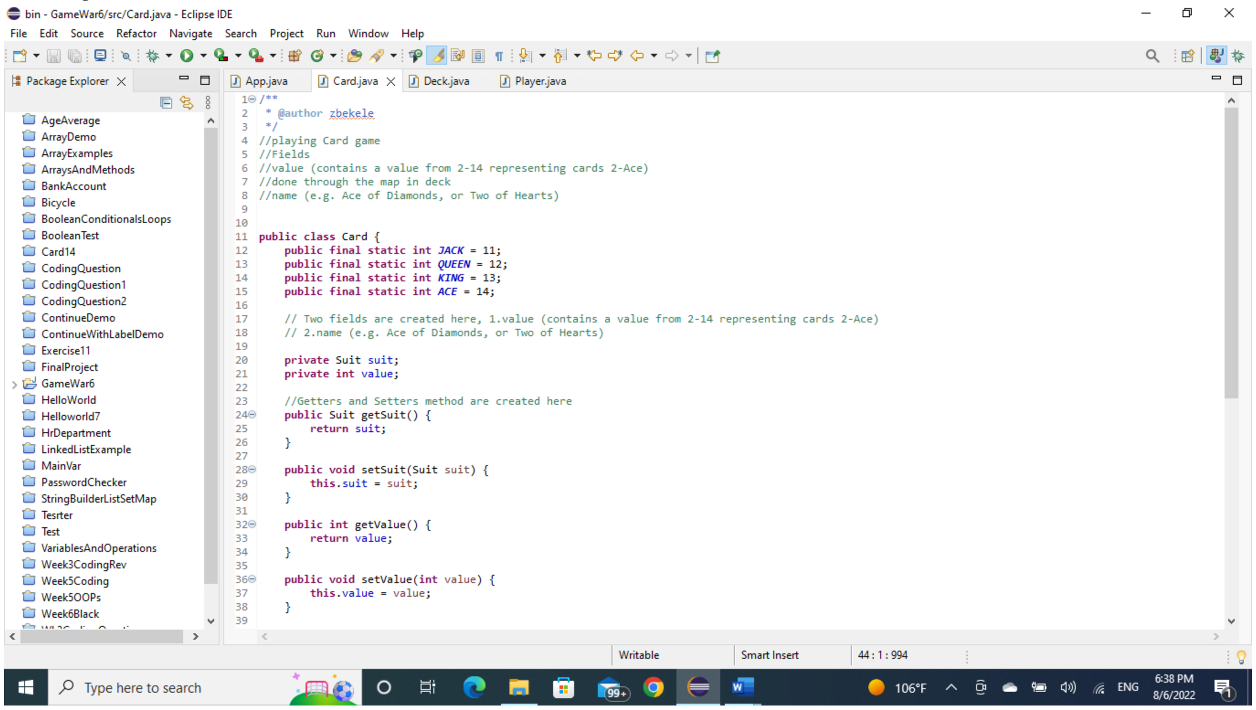
            3. In the constructor, when a new Deck is instantiated, the Cards field should be populated with the standard 52 cards.

    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)

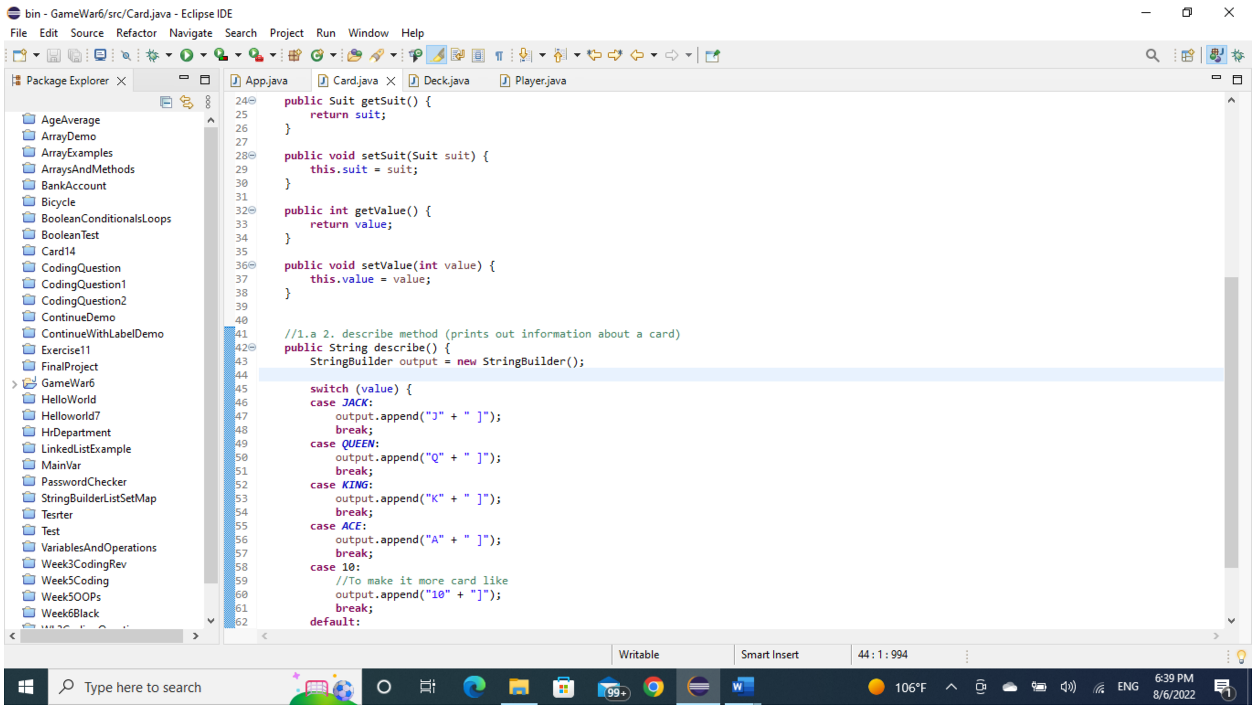
2. Create a class called App with a main method.
3. Instantiate a Deck and two Players, call the shuffle method on the deck.
4. Using a traditional for loop, iterate 52 times calling the Draw method on the other player each iteration using the Deck you instantiated.
5. Using a traditional for loop, iterate 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 java

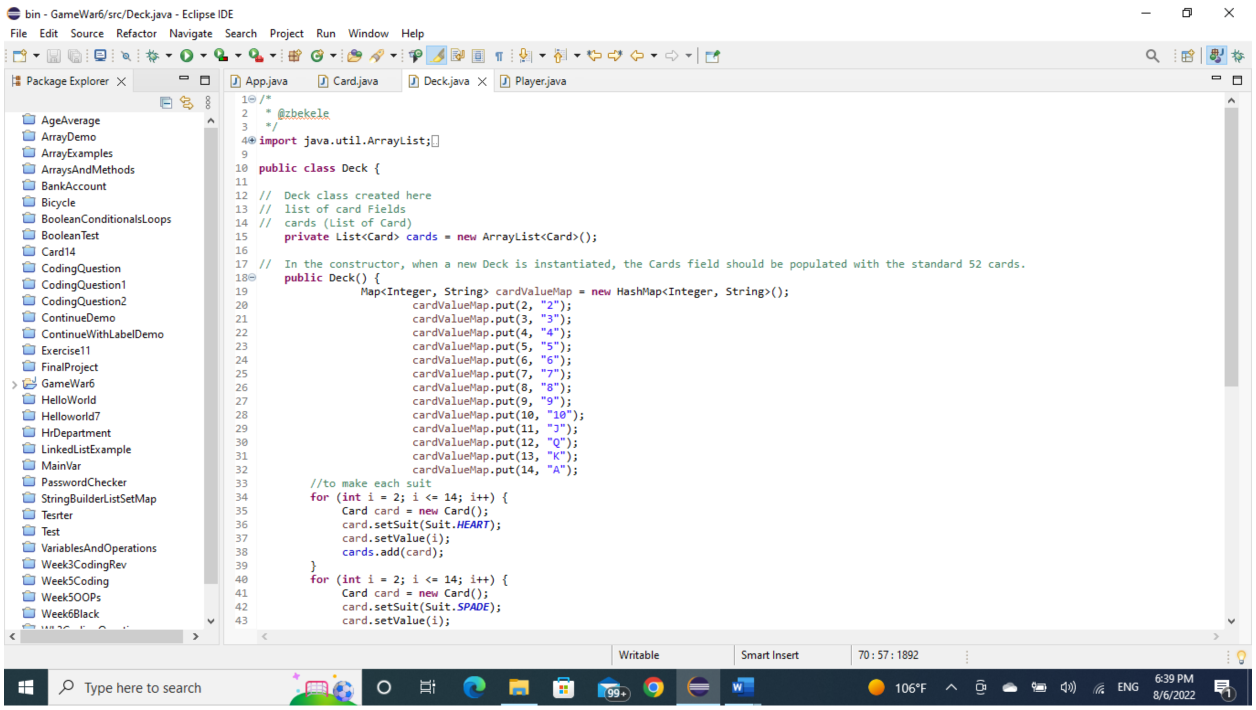


```java
/**
 * @author zbekele
 */
//playing Card game
//Fields
//value (contains a value from 2-14 representing cards 2-Ace)
//done through the map in deck
//name (e.g. Ace of Diamonds, or Two of Hearts)

public class Card {
    public final static int JACK = 11;
    public final static int QUEEN = 12;
    public final static int KING = 13;
    public final static int ACE = 14;

    // Two fields are created here, 1.value (contains a value from 2-14 representing cards 2-Ace)
    // 2.name (e.g. Ace of Diamonds, or Two of Hearts)

    private Suit suit;
    private int value;

    //Getters and Setters method are created here
    public Suit getSuit() {
        return suit;
    }

    public void setSuit(Suit suit) {
        this.suit = suit;
    }

    public int getValue() {
        return value;
    }

    public void setValue(int value) {
        this.value = value;
    }

```

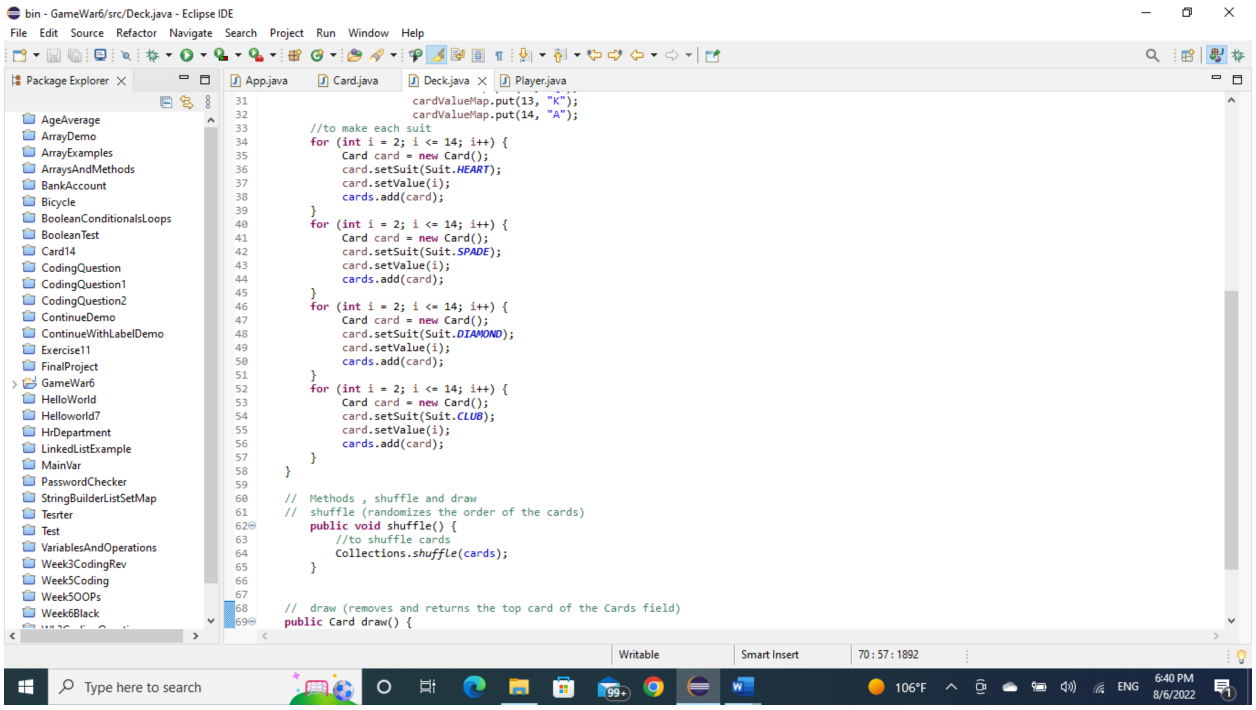


```java
    public Suit getSuit() {
        return suit;
    }

    public void setSuit(Suit suit) {
        this.suit = suit;
    }

    public int getValue() {
        return value;
    }

    public void setValue(int value) {
        this.value = value;
    }


    //1.a 2. describe method (prints out information about a card)
    public String describe() {
        StringBuilder output = new StringBuilder();

        switch (value) {
        case JACK:
            output.append("J" + " ]");
            break;
        case QUEEN:
            output.append("Q" + " ]");
            break;
        case KING:
            output.append("K" + " ]");
            break;
        case ACE:
            output.append("A" + " ]");
            break;
        case 10:
            //To make it more card like
            output.append("10" + "]");
            break;
        default:
```

Deck Java

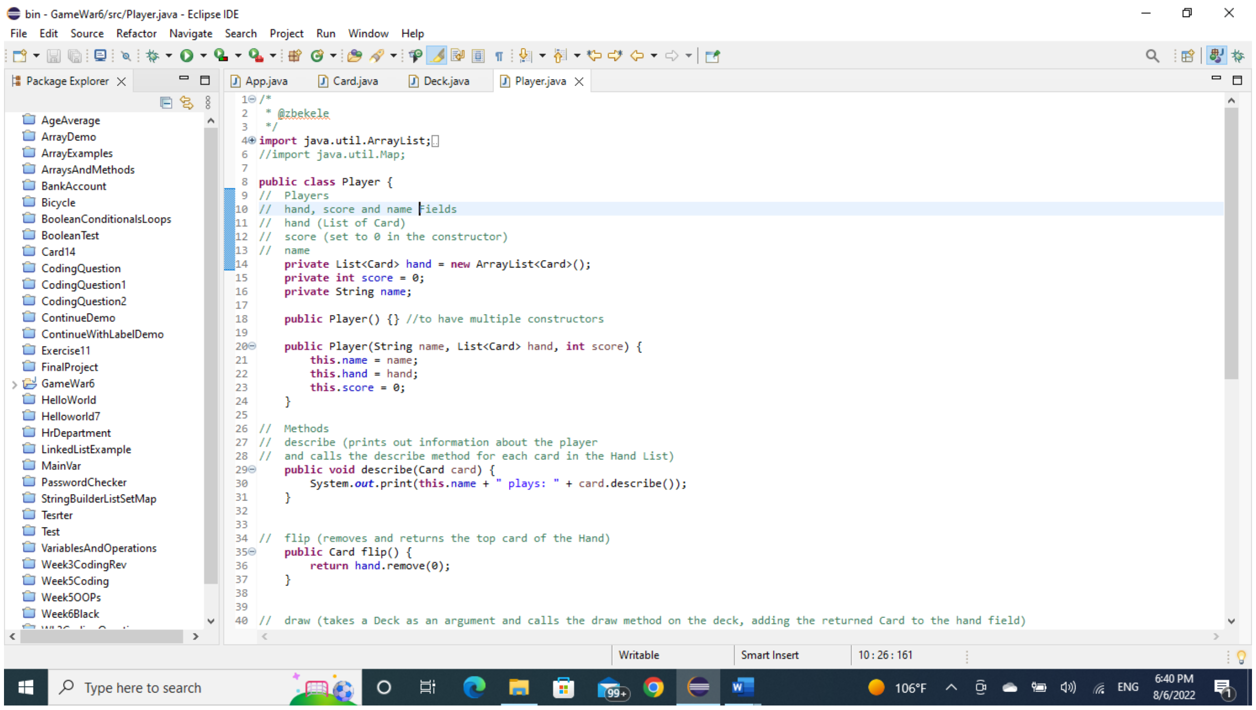


```java
/*
 * @zbekele
 */
import java.util.ArrayList;

public class Deck {

//  Deck class created here
//  list of card Fields
//  cards (List of Card)
    private List<Card> cards = new ArrayList<Card>();

//  In the constructor, when a new Deck is instantiated, the Cards field should be populated with the standard 52 cards.
    public Deck() {
                Map<Integer, String> cardValueMap = new HashMap<Integer, String>();
                        cardValueMap.put(2, "2");
                        cardValueMap.put(3, "3");
                        cardValueMap.put(4, "4");
                        cardValueMap.put(5, "5");
                        cardValueMap.put(6, "6");
                        cardValueMap.put(7, "7");
                        cardValueMap.put(8, "8");
                        cardValueMap.put(9, "9");
                        cardValueMap.put(10, "10");
                        cardValueMap.put(11, "J");
                        cardValueMap.put(12, "Q");
                        cardValueMap.put(13, "K");
                        cardValueMap.put(14, "A");
        //to make each suit
        for (int i = 2; i <= 14; i++) {
             Card card = new Card();
             card.setSuit(Suit.HEART);
             card.setValue(i);
             cards.add(card);
        }
        for (int i = 2; i <= 14; i++) {
             Card card = new Card();
             card.setSuit(Suit.SPADE);
             card.setValue(i);
```

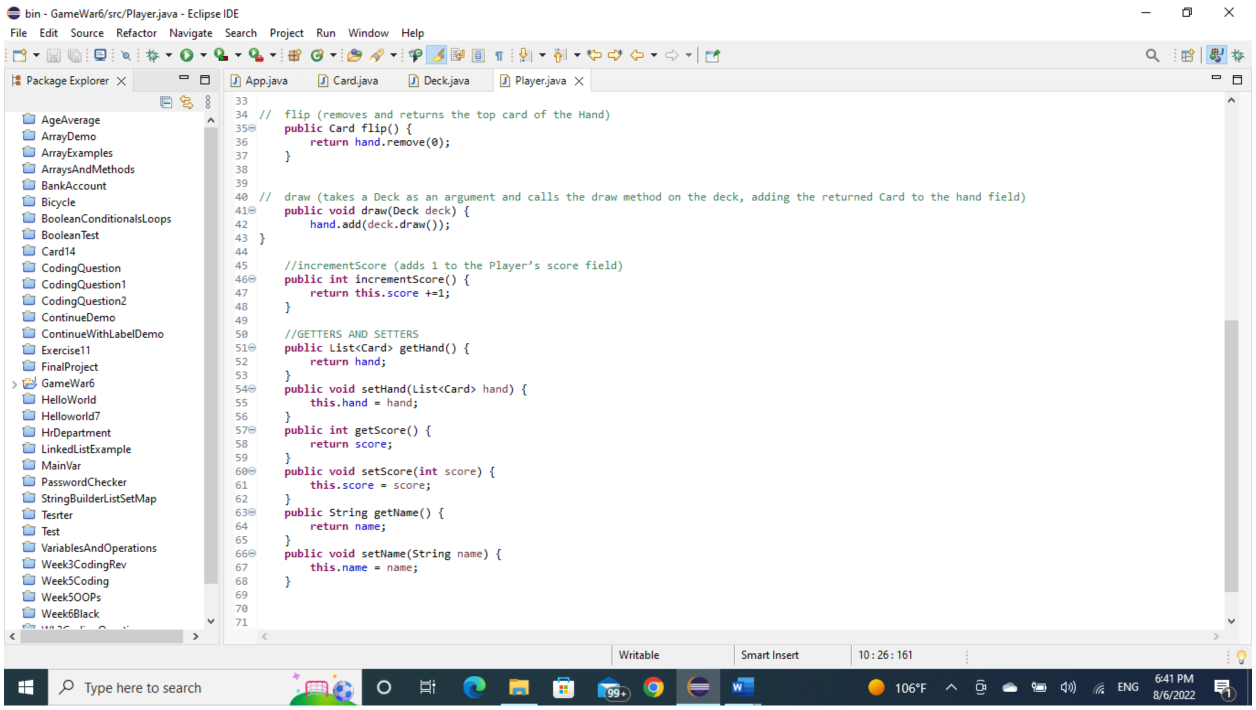


```java
                        cardValueMap.put(13, "K");
                        cardValueMap.put(14, "A");
        //to make each suit
        for (int i = 2; i <= 14; i++) {
             Card card = new Card();
             card.setSuit(Suit.HEART);
             card.setValue(i);
             cards.add(card);
        }
        for (int i = 2; i <= 14; i++) {
             Card card = new Card();
             card.setSuit(Suit.SPADE);
             card.setValue(i);
             cards.add(card);
        }
        for (int i = 2; i <= 14; i++) {
             Card card = new Card();
             card.setSuit(Suit.DIAMOND);
             card.setValue(i);
             cards.add(card);
        }
        for (int i = 2; i <= 14; i++) {
             Card card = new Card();
             card.setSuit(Suit.CLUB);
             card.setValue(i);
             cards.add(card);
        }
    }

//  Methods , shuffle and draw
//  shuffle (randomizes the order of the cards)
        public void shuffle() {
            //to shuffle cards
            Collections.shuffle(cards);
        }


//  draw (removes and returns the top card of the Cards field)
    public Card draw() {
```

Player java

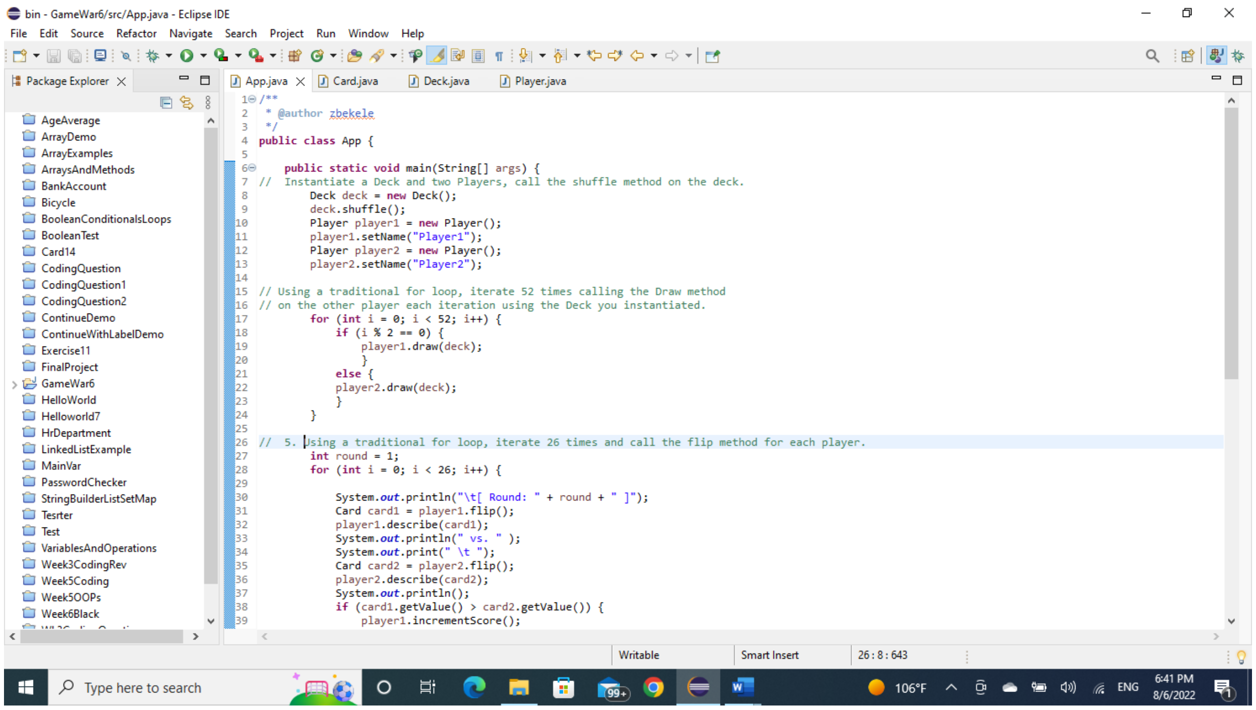


```java
/*
 * @zbekele
 */
import java.util.ArrayList;
//import java.util.Map;

public class Player {
//  Players
//  hand, score and name Fields
//  hand (List of Card)
//  score (set to 0 in the constructor)
//  name
    private List<Card> hand = new ArrayList<Card>();
    private int score = 0;
    private String name;

    public Player() {} //to have multiple constructors

    public Player(String name, List<Card> hand, int score) {
        this.name = name;
        this.hand = hand;
        this.score = 0;
    }

//  Methods
//  describe (prints out information about the player
//  and calls the describe method for each card in the Hand List)
    public void describe(Card card) {
        System.out.print(this.name + " plays: " + card.describe());
    }


//  flip (removes and returns the top card of the Hand)
    public Card flip() {
        return hand.remove(0);
    }


//  draw (takes a Deck as an argument and calls the draw method on the deck, adding the returned Card to the hand field)
```



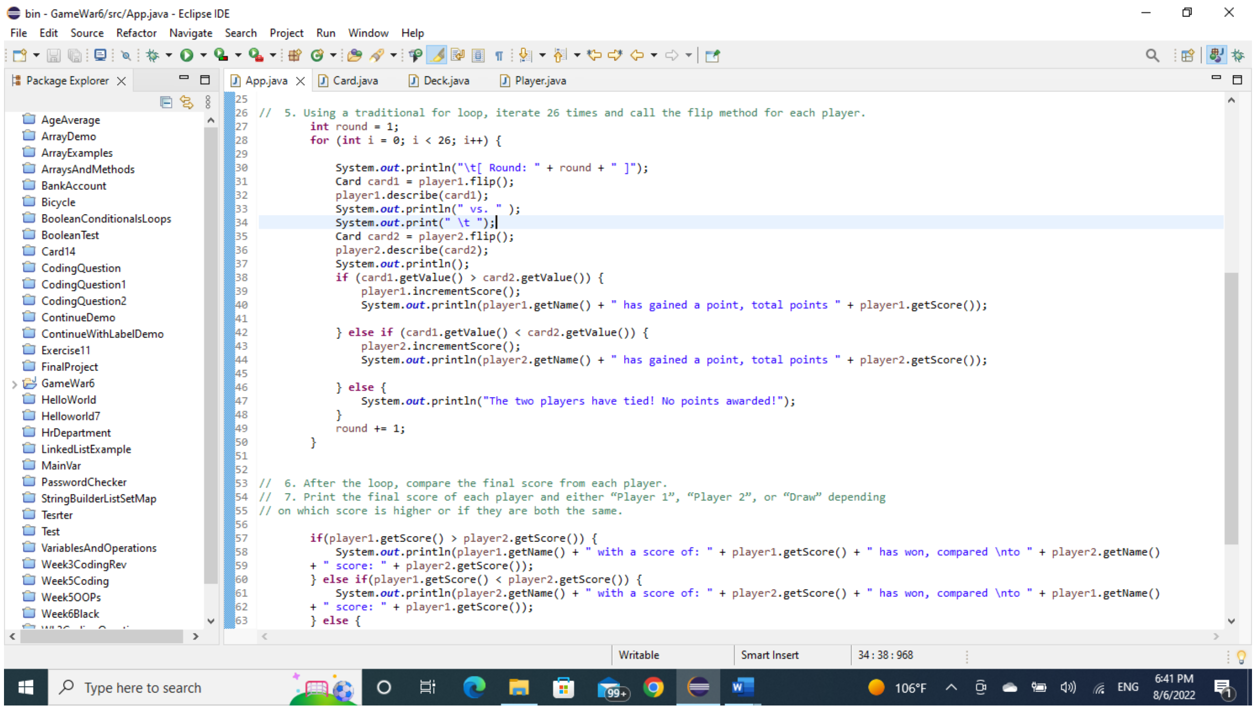
```java

//  flip (removes and returns the top card of the Hand)
    public Card flip() {
        return hand.remove(0);
    }


//  draw (takes a Deck as an argument and calls the draw method on the deck, adding the returned Card to the hand field)
    public void draw(Deck deck) {
        hand.add(deck.draw());
}

    //incrementScore (adds 1 to the Player's score field)
    public int incrementScore() {
        return this.score +=1;
    }

    //GETTERS AND SETTERS
    public List<Card> getHand() {
        return hand;
    }
    public void setHand(List<Card> hand) {
        this.hand = hand;
    }
    public int getScore() {
        return score;
    }
    public void setScore(int score) {
        this.score = score;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }



```

App Java c



bin - GameWar6/src/App.java - Eclipse IDE

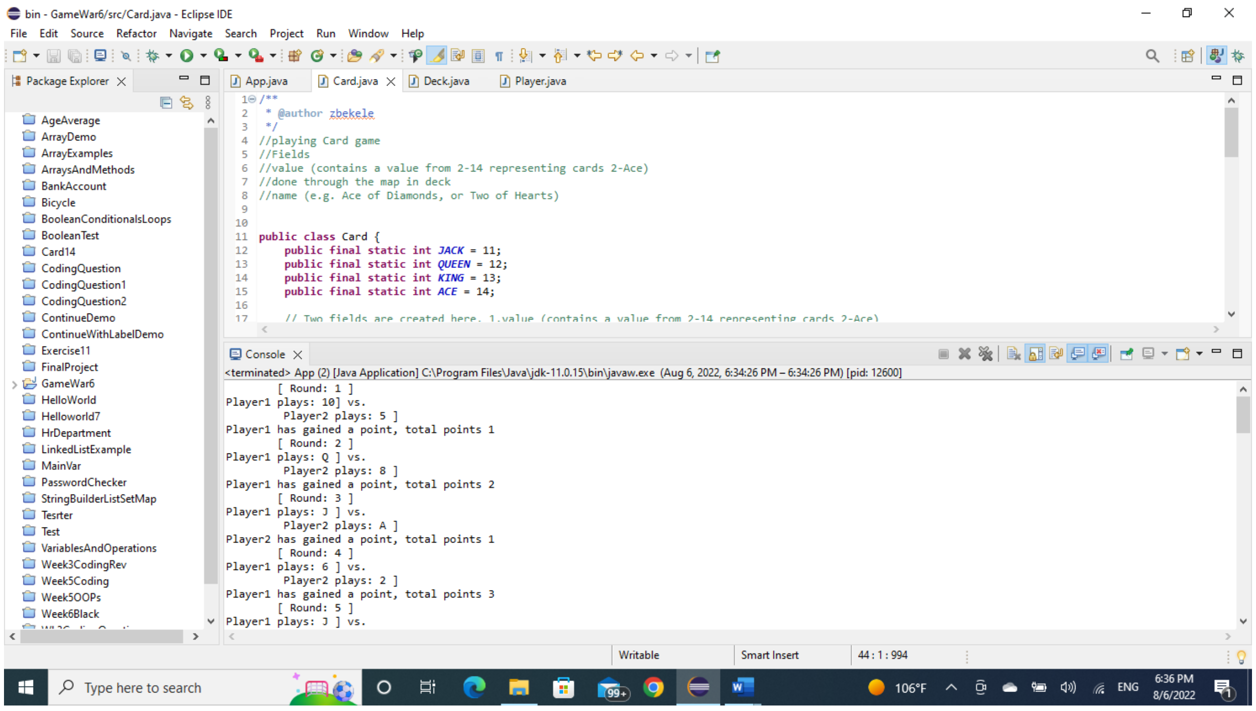
```java
/**
 * @author zbekele
 */
public class App {

    public static void main(String[] args) {
//  Instantiate a Deck and two Players, call the shuffle method on the deck.
        Deck deck = new Deck();
        deck.shuffle();
        Player player1 = new Player();
        player1.setName("Player1");
        Player player2 = new Player();
        player2.setName("Player2");

// Using a traditional for loop, iterate 52 times calling the Draw method
// on the other player each iteration using the Deck you instantiated.
        for (int i = 0; i < 52; i++) {
            if (i % 2 == 0) {
                player1.draw(deck);
                }
            else {
            player2.draw(deck);
            }
        }

//  5. Using a traditional for loop, iterate 26 times and call the flip method for each player.
        int round = 1;
        for (int i = 0; i < 26; i++) {

            System.out.println("\t[ Round: " + round + " ]");
            Card card1 = player1.flip();
            player1.describe(card1);
            System.out.println(" vs. " );
            System.out.print(" \t ");
            Card card2 = player2.flip();
            player2.describe(card2);
            System.out.println();
            if (card1.getValue() > card2.getValue()) {
                player1.incrementScore();
```



bin - GameWar6/src/App.java - Eclipse IDE

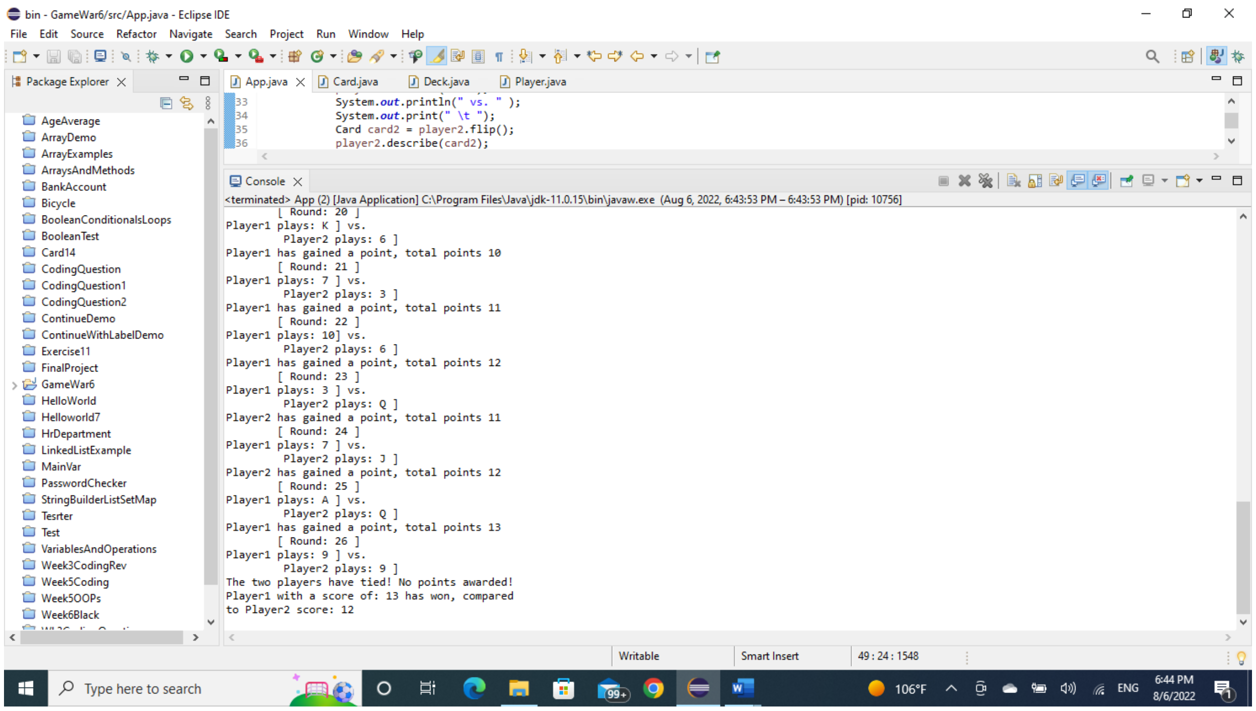
```java

//  5. Using a traditional for loop, iterate 26 times and call the flip method for each player.
        int round = 1;
        for (int i = 0; i < 26; i++) {

            System.out.println("\t[ Round: " + round + " ]");
            Card card1 = player1.flip();
            player1.describe(card1);
            System.out.println(" vs. " );
            System.out.print(" \t ");
            Card card2 = player2.flip();
            player2.describe(card2);
            System.out.println();
            if (card1.getValue() > card2.getValue()) {
                player1.incrementScore();
                System.out.println(player1.getName() + " has gained a point, total points " + player1.getScore());

            } else if (card1.getValue() < card2.getValue()) {
                player2.incrementScore();
                System.out.println(player2.getName() + " has gained a point, total points " + player2.getScore());

            } else {
                System.out.println("The two players have tied! No points awarded!");
            }
            round += 1;
        }


//  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.

        if(player1.getScore() > player2.getScore()) {
            System.out.println(player1.getName() + " with a score of: " + player1.getScore() + " has won, compared \nto " + player2.getName()
        + " score: " + player2.getScore());
        } else if(player1.getScore() < player2.getScore()) {
            System.out.println(player2.getName() + " with a score of: " + player2.getScore() + " has won, compared \nto " + player1.getName()
        + " score: " + player1.getScore());
        } else {
```

Screenshots of Running Application:



```java
/**
 * @author zbekele
 */
//playing Card game
//Fields
//value (contains a value from 2-14 representing cards 2-Ace)
//done through the map in deck
//name (e.g. Ace of Diamonds, or Two of Hearts)


public class Card {
    public final static int JACK = 11;
    public final static int QUEEN = 12;
    public final static int KING = 13;
    public final static int ACE = 14;

    // Two fields are created here. 1.value (contains a value from 2-14 representing cards 2-Ace)
```

```
<terminated> App (2) [Java Application] C:\Program Files\Java\jdk-11.0.15\bin\javaw.exe  (Aug 6, 2022, 6:34:26 PM – 6:34:26 PM) [pid: 12600]
        [ Round: 1 ]
Player1 plays: 10] vs.
         Player2 plays: 5 ]
Player1 has gained a point, total points 1
        [ Round: 2 ]
Player1 plays: Q ] vs.
         Player2 plays: 8 ]
Player1 has gained a point, total points 2
        [ Round: 3 ]
Player1 plays: J ] vs.
         Player2 plays: A ]
Player2 has gained a point, total points 1
        [ Round: 4 ]
Player1 plays: 6 ] vs.
         Player2 plays: 2 ]
Player1 has gained a point, total points 3
        [ Round: 5 ]
Player1 plays: J ] vs.
```



```java
            System.out.println(" vs. " );
            System.out.print(" \t ");
            Card card2 = player2.flip();
            player2.describe(card2);
```

```
<terminated> App (2) [Java Application] C:\Program Files\Java\jdk-11.0.15\bin\javaw.exe  (Aug 6, 2022, 6:43:53 PM – 6:43:53 PM) [pid: 10756]
        [ Round: 20 ]
Player1 plays: K ] vs.
         Player2 plays: 6 ]
Player1 has gained a point, total points 10
        [ Round: 21 ]
Player1 plays: 7 ] vs.
         Player2 plays: 3 ]
Player1 has gained a point, total points 11
        [ Round: 22 ]
Player1 plays: 10] vs.
         Player2 plays: 6 ]
Player1 has gained a point, total points 12
        [ Round: 23 ]
Player1 plays: 3 ] vs.
         Player2 plays: Q ]
Player2 has gained a point, total points 11
        [ Round: 24 ]
Player1 plays: 7 ] vs.
         Player2 plays: J ]
Player2 has gained a point, total points 12
        [ Round: 25 ]
Player1 plays: A ] vs.
         Player2 plays: Q ]
Player1 has gained a point, total points 13
        [ Round: 26 ]
Player1 plays: 9 ] vs.
         Player2 plays: 9 ]
The two players have tied! No points awarded!
Player1 with a score of: 13 has won, compared
to Player2 score: 12
```

URL to GitHub Repository:

Upload files · Zbekele2022/Week-6-Java-Final-project (github.com)