

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*.

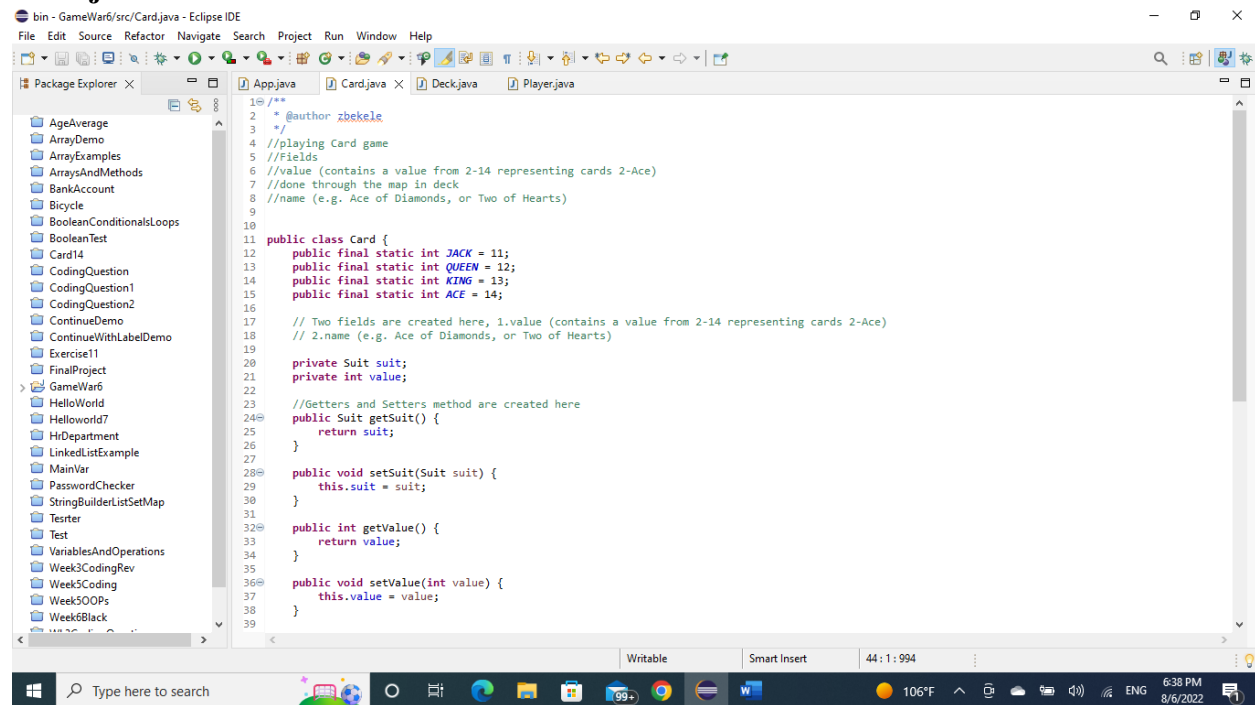
1. Create the following classes.

- 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)

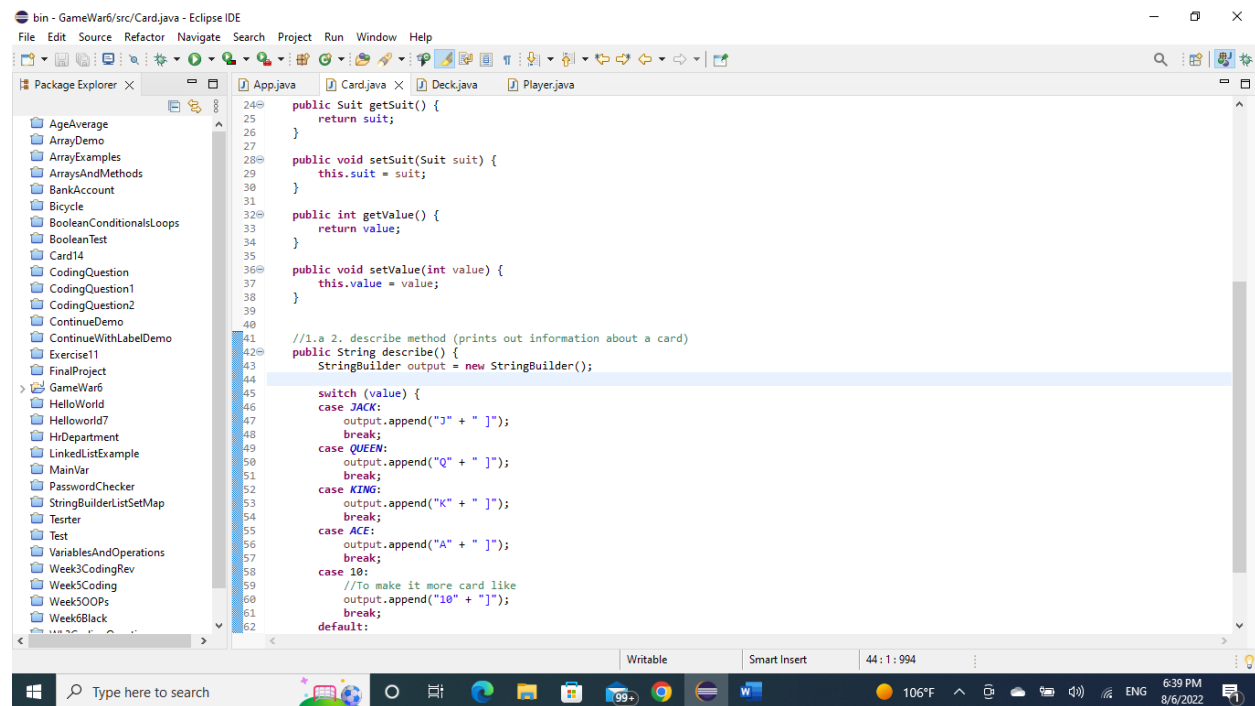
3. In the constructor, when a new Deck is instantiated, the Cards field should be populated with the standard 52 cards.
- c. Player
 - i. Fields
 1. **hand** (List of Card)
 2. **score** (set to 0 in the constructor)
 3. **name**
 - ii. Methods
 1. **describe** (prints out information about the player 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



```
10 /**
11  * @author zbekele
12  */
13 //playing Card game
14 //Fields
15 //value (contains a value from 2-14 representing cards 2-Ace)
16 //done through the map in deck
17 //name (e.g. Ace of Diamonds, or Two of Hearts)
18
19
20 public class Card {
21     public final static int JACK = 11;
22     public final static int QUEEN = 12;
23     public final static int KING = 13;
24     public final static int ACE = 14;
25
26     // Two fields are created here, 1.value (contains a value from 2-14 representing cards 2-Ace)
27     // 2.name (e.g. Ace of Diamonds, or Two of Hearts)
28
29     private Suit suit;
30     private int value;
31
32     //Getters and Setters method are created here
33     public Suit getSuit() {
34         return suit;
35     }
36
37     public void setSuit(Suit suit) {
38         this.suit = suit;
39     }
40
41     public int getValue() {
42         return value;
43     }
44
45     public void setValue(int value) {
46         this.value = value;
47     }
48 }
```



```
24 public Suit getSuit() {
25     return suit;
26 }
27
28 public void setSuit(Suit suit) {
29     this.suit = suit;
30 }
31
32 public int getValue() {
33     return value;
34 }
35
36 public void setValue(int value) {
37     this.value = value;
38 }
39
40
41 //1.a 2. describe method (prints out information about a card)
42 public String describe() {
43     StringBuilder output = new StringBuilder();
44
45     switch (value) {
46     case JACK:
47         output.append("J" + " ");
48         break;
49     case QUEEN:
50         output.append("Q" + " ");
51         break;
52     case KING:
53         output.append("K" + " ");
54         break;
55     case ACE:
56         output.append("A" + " ");
57         break;
58     case 10:
59         //To make it more card like
60         output.append("10" + " ");
61         break;
62     default:
```

The screenshot displays the Eclipse IDE environment. The Package Explorer on the left shows a project structure with various folders and files, including 'AgeAverage', 'ArrayDemo', 'ArrayExamples', 'ArraysAndMethods', 'BankAccount', 'Bicycle', 'BooleanConditionalLoops', 'BooleanTest', 'Card14', 'CodingQuestion', 'CodingQuestion1', 'CodingQuestion2', 'ContinueDemo', 'ContinueWithLabelDemo', 'Exercise11', 'FinalProject', 'GameWarf', 'HelloWorld', 'Helloworld7', 'HrDepartment', 'LinkedListExample', 'MainVar', 'PasswordChecker', 'StringBuilderSetMap', 'Tester', 'Test', 'VariablesAndOperations', 'Week3CodingRev', 'Week5Coding', 'Week5OOPs', and 'Week6Black'. The main editor window shows the code for 'Deck.java'.

```
1 //  
2 * @sbekele  
3  
4 import java.util.ArrayList;  
5  
6 public class Deck {  
7  
8     // Deck class created here  
9     // list of card fields  
10    // cards (List of Card)  
11    private List<Card> cards = new ArrayList<Card>();  
12  
13    // In the constructor, when a new Deck is instantiated, the Cards field should be populated with the standard 52 cards.  
14    public Deck() {  
15        Map<Integer, String> cardValueMap = new HashMap<Integer, String>();  
16        cardValueMap.put(2, "2");  
17        cardValueMap.put(3, "3");  
18        cardValueMap.put(4, "4");  
19        cardValueMap.put(5, "5");  
20        cardValueMap.put(6, "6");  
21        cardValueMap.put(7, "7");  
22        cardValueMap.put(8, "8");  
23        cardValueMap.put(9, "9");  
24        cardValueMap.put(10, "10");  
25        cardValueMap.put(11, "J");  
26        cardValueMap.put(12, "Q");  
27        cardValueMap.put(13, "K");  
28        cardValueMap.put(14, "A");  
29  
30        //to make each suit  
31        for (int i = 2; i <= 14; i++) {  
32            Card card = new Card(i);  
33            card.setSuit(Suit.HEART);  
34            card.setValue(i);  
35            cards.add(card);  
36        }  
37        for (int i = 2; i <= 14; i++) {  
38            Card card = new Card(i);  
39            card.setSuit(Suit.SPADE);  
40            card.setValue(i);  
41        }  
42    }  
43 }
```

The status bar at the bottom indicates the file is 'Writable', the cursor is at 'Smart Insert', the line number is '70', the column number is '57', the total line count is '1892', the temperature is '106°F', and the date and time are '8/6/2022 6:39 PM'.



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

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer

- AgeAverage
- ArrayDemo
- ArrayExamples
- ArraysAndMethods
- BankAccount
- Bicycle
- BooleanConditionalsLoops
- BooleanTest
- Card14
- CodingQuestion
- CodingQuestion1
- CodingQuestion2
- ContinueDemo
- ContinueWithLabelDemo
- Exercise11
- FinalProject
- GameWar6
- HelloWorld
- HelloWorld7
- HrDepartment
- LinkedListExample
- MainVar
- PasswordChecker
- StringBuilderListSetMap
- Tester
- Test
- VariablesAndOperations
- Week3CodingRev
- Week3Coding
- Week5OOPs
- Week6Black

```
1 10 /*
2  * @bakele
3  */
4 4 import java.util.ArrayList;
5 5 //import java.util.Map;
6 6
7 7
8 8 public class Player {
9 9 // Players
10 10 // hand, score and name fields
11 11 // hand (List of Card)
12 12 // score (set to 0 in the constructor)
13 13 // name
14 14 private List<Card> hand = new ArrayList<Card>();
15 15 private int score = 0;
16 16 private String name;
17 17
18 18 public Player() {} //to have multiple constructors
19 19
20 20 public Player(String name, List<Card> hand, int score) {
21 21 this.name = name;
22 22 this.hand = hand;
23 23 this.score = 0;
24 24 }
25 25
26 26 // Methods
27 27 describe (prints out information about the player
28 28 and calls the describe method for each card in the Hand List)
29 29 public void describe(Card card) {
30 30 System.out.print(this.name + " plays: " + card.describe());
31 31 }
32 32
33 33
34 34 // flip (removes and returns the top card of the Hand)
35 35 public Card flip() {
36 36 return hand.remove(0);
37 37 }
38 38
39 39
40 40 // draw (takes a Deck as an argument and calls the draw method on the deck, adding the returned Card to the hand field)
```

Writable Smart Insert 10:26:161

Type here to search 106°F 6:40 PM 8/6/2022

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

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer

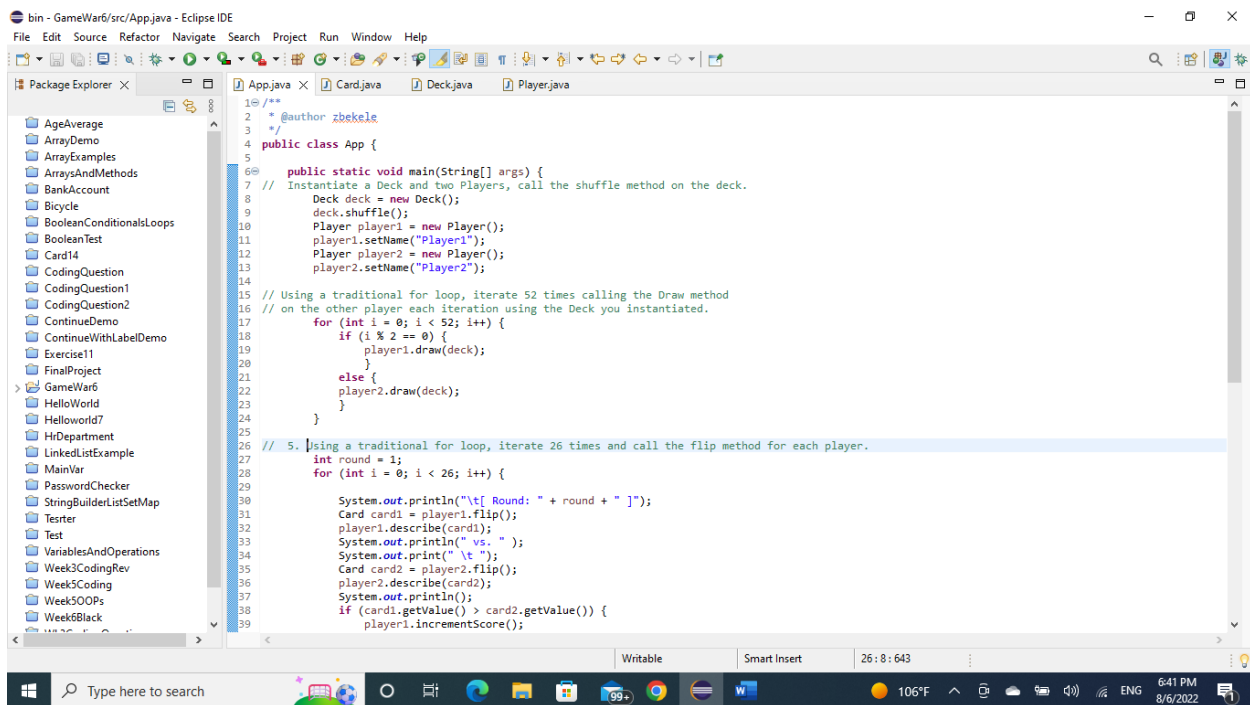
- AgeAverage
- ArrayDemo
- ArrayExamples
- ArraysAndMethods
- BankAccount
- Bicycle
- BooleanConditionalsLoops
- BooleanTest
- Card14
- CodingQuestion
- CodingQuestion1
- CodingQuestion2
- ContinueDemo
- ContinueWithLabelDemo
- Exercise11
- FinalProject
- GameWar6
- HelloWorld
- HelloWorld7
- HrDepartment
- LinkedListExample
- MainVar
- PasswordChecker
- StringBuilderListSetMap
- Tester
- Test
- VariablesAndOperations
- Week3CodingRev
- Week3Coding
- Week5OOPs
- Week6Black

```
33 33
34 34 // flip (removes and returns the top card of the Hand)
35 35 public Card flip() {
36 36 return hand.remove(0);
37 37 }
38 38
39 39
40 40 // draw (takes a Deck as an argument and calls the draw method on the deck, adding the returned Card to the hand field)
41 41 public void draw(Deck deck) {
42 42 hand.add(deck.draw());
43 43 }
44 44
45 45 //incrementScore (adds 1 to the Player's score field)
46 46 public int incrementScore() {
47 47 return this.score +1;
48 48 }
49 49
50 50 //GETTERS AND SETTERS
51 51 public List<Card> getHand() {
52 52 return hand;
53 53 }
54 54 public void setHand(List<Card> hand) {
55 55 this.hand = hand;
56 56 }
57 57 public int getScore() {
58 58 return score;
59 59 }
60 60 public void setScore(int score) {
61 61 this.score = score;
62 62 }
63 63 public String getName() {
64 64 return name;
65 65 }
66 66 public void setName(String name) {
67 67 this.name = name;
68 68 }
69 69
70 70
71 71
```

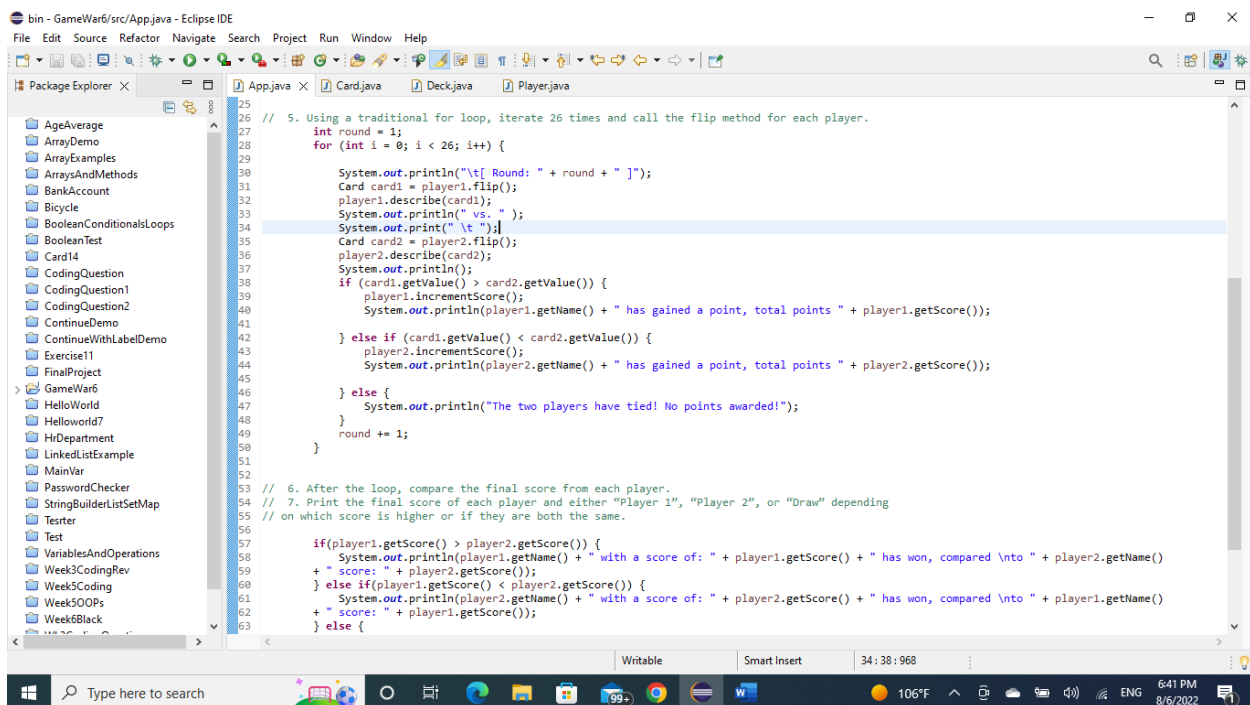
Writable Smart Insert 10:26:161

Type here to search 106°F 6:41 PM 8/6/2022

App Java c



```
1 /**
2  * @author zbekele
3  */
4  public class App {
5
6      public static void main(String[] args) {
7          // Instantiate a Deck and two Players, call the shuffle method on the deck.
8          Deck deck = new Deck();
9          deck.shuffle();
10         Player player1 = new Player();
11         player1.setName("Player1");
12         Player player2 = new Player();
13         player2.setName("Player2");
14
15         // Using a traditional for loop, iterate 52 times calling the Draw method
16         // on the other player each iteration using the Deck you instantiated.
17         for (int i = 0; i < 52; i++) {
18             if (i % 2 == 0) {
19                 player1.draw(deck);
20             }
21             else {
22                 player2.draw(deck);
23             }
24         }
25
26         // 5. Using a traditional for loop, iterate 26 times and call the flip method for each player.
27         int round = 1;
28         for (int i = 0; i < 26; i++) {
29
30             System.out.println("\t Round: " + round + " ");
31             Card card1 = player1.flip();
32             player1.describe(card1);
33             System.out.println(" vs. ");
34             System.out.print("\t ");
35             Card card2 = player2.flip();
36             player2.describe(card2);
37             System.out.println();
38             if (card1.getValue() > card2.getValue()) {
39                 player1.incrementScore();
40             }
41         }
42     }
43 }
```



```
25
26 // 5. Using a traditional for loop, iterate 26 times and call the flip method for each player.
27 int round = 1;
28 for (int i = 0; i < 26; i++) {
29
30     System.out.println("\t Round: " + round + " ");
31     Card card1 = player1.flip();
32     player1.describe(card1);
33     System.out.println(" vs. ");
34     System.out.print("\t ");
35     Card card2 = player2.flip();
36     player2.describe(card2);
37     System.out.println();
38     if (card1.getValue() > card2.getValue()) {
39         player1.incrementScore();
40         System.out.println(player1.getName() + " has gained a point, total points " + player1.getScore());
41     }
42     else if (card1.getValue() < card2.getValue()) {
43         player2.incrementScore();
44         System.out.println(player2.getName() + " has gained a point, total points " + player2.getScore());
45     }
46     else {
47         System.out.println("The two players have tied! No points awarded!");
48     }
49     round += 1;
50 }
51
52 // 6. After the loop, compare the final score from each player.
53 // 7. Print the final score of each player and either "Player 1", "Player 2", or "Draw" depending
54 // on which score is higher or if they are both the same.
55
56 if (player1.getScore() > player2.getScore()) {
57     System.out.println(player1.getName() + " with a score of: " + player1.getScore() + " has won, compared \nto " + player2.getName()
58     + " score: " + player2.getScore());
59 }
60 else if (player1.getScore() < player2.getScore()) {
61     System.out.println(player2.getName() + " with a score of: " + player2.getScore() + " has won, compared \nto " + player1.getName()
62     + " score: " + player1.getScore());
63 }
64 else {
65     System.out.println("The game is a draw!");
66 }
```

Screenshots of Running Application:

The screenshot shows the Eclipse IDE with the `Card.java` file open. The code defines a `Card` class with static final values for `JACK`, `QUEEN`, `KING`, and `ACE`. The console output shows the execution of the `App.java` file, which simulates a card game. The output displays the cards dealt to Player1 and Player2, the points gained, and the total points for each player across five rounds.

```
10 /**
11  * @author zbekele
12  */
13 //playing Card game
14 //value (contains a value from 2-14 representing cards 2-Ace)
15 //done through the map in deck
16 //name (e.g. Ace of Diamonds, or Two of Hearts)
17
18 public class Card {
19     public final static int JACK = 11;
20     public final static int QUEEN = 12;
21     public final static int KING = 13;
22     public final static int ACE = 14;
23 }
24
25 // 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.
```

The screenshot shows the Eclipse IDE with the `App.java` file open. The code prints the cards dealt to Player1 and Player2, the points gained, and the total points for each player across 26 rounds. The console output shows the execution of the `App.java` file, which simulates a card game. The output displays the cards dealt to Player1 and Player2, the points gained, and the total points for each player across 26 rounds. The game ends with a tie at round 26, and the final scores are displayed.

```
33 System.out.println(" vs. ");
34 System.out.print("\t ");
35 Card card2 = player2.flip();
36 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 \(github.com\)](#)