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.*

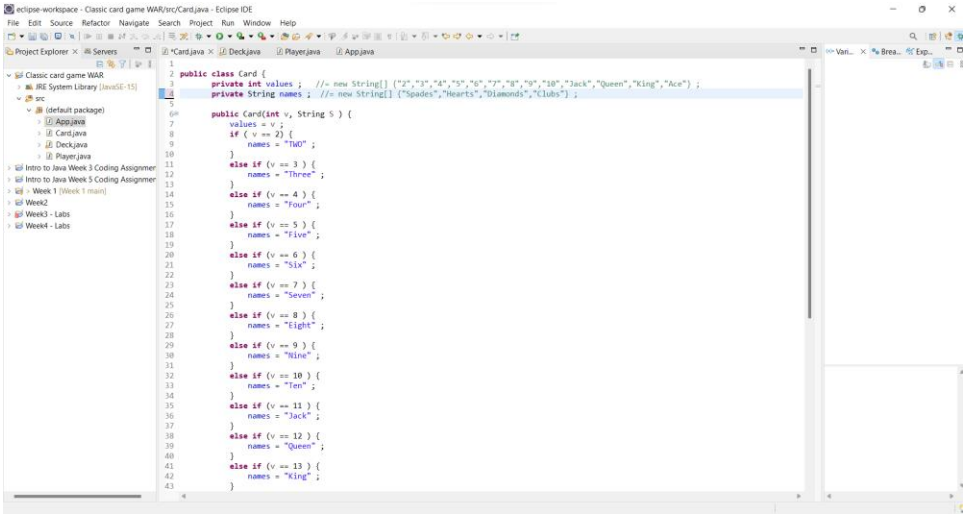
1. Create the following classes.
    a. Card
        i. Fields
            1. **value** (contains a value from 2-14 representing cards 2-Ace)
            2. **name** (e.g. Ace of Diamonds, or Two of Hearts)
        ii. Methods
            1. Getters and Setters
            2. **describe** (prints out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

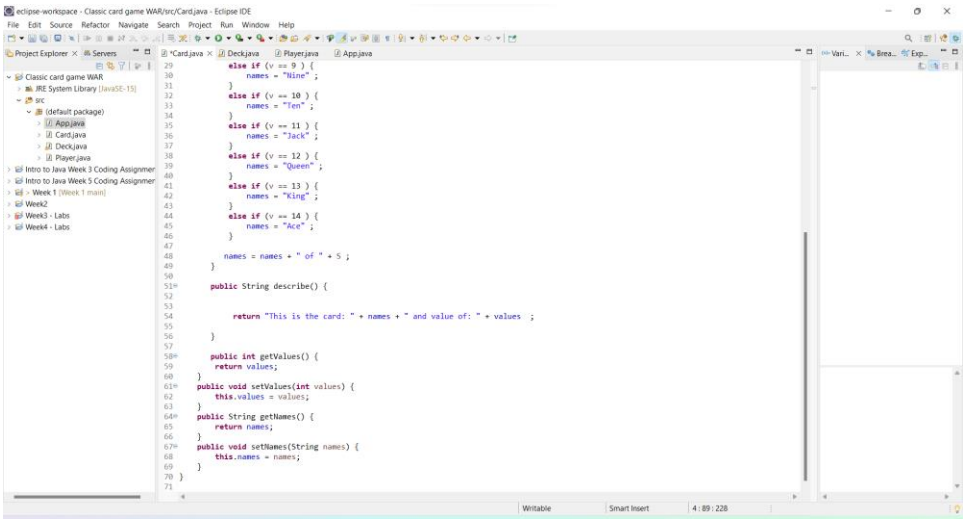
3. In the constructor, when a new Deck is instantiated, the Cards field should be populated with the standard 52 cards.

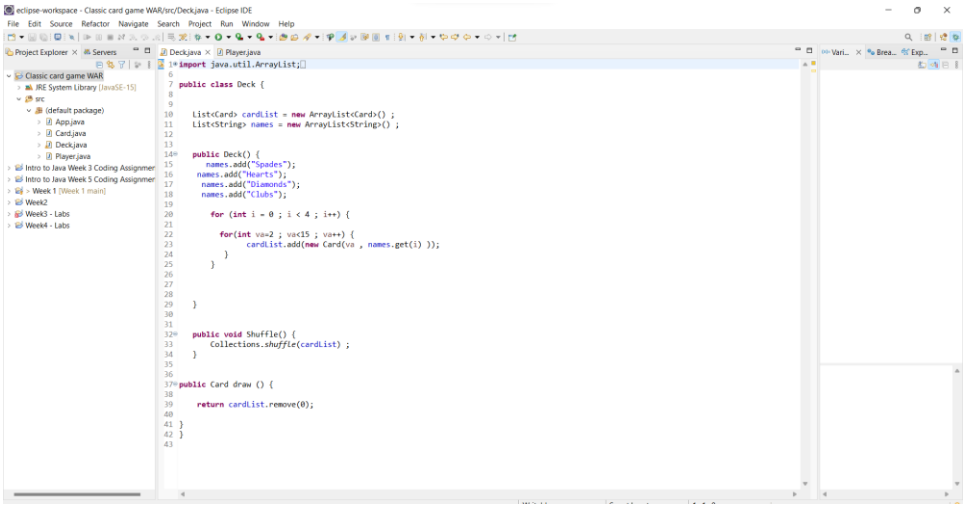
c. Player
    i. Fields
        1. **hand** (List of Card)
        2. **score** (set to 0 in the constructor)
        3. **name**
    ii. Methods
        1. **describe** (prints out information about the player and calls the describe method for each card in the Hand List)
        2. **flip** (removes 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)

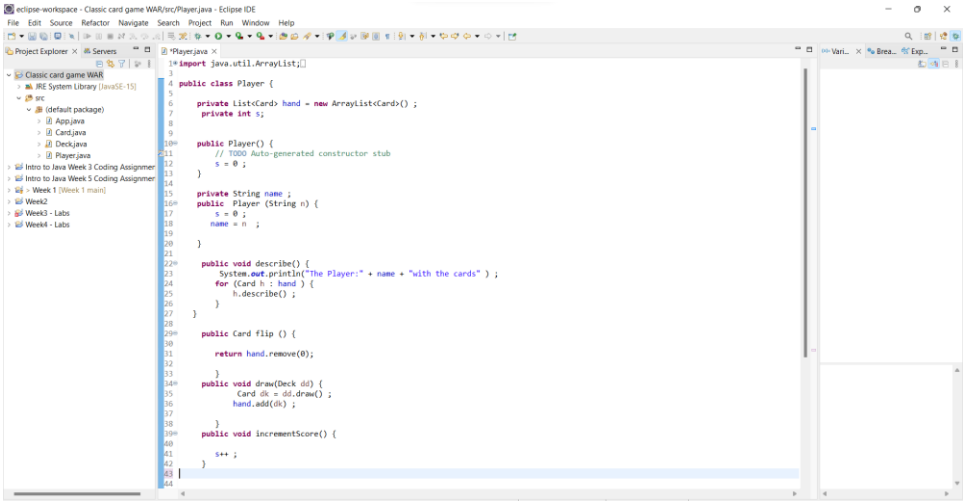
2. Create a class called App with a main method.
3. Instantiate a Deck and two Players, call the shuffle method on the deck.
4. Using a traditional for loop, iterate 52 times calling the Draw method on the other player each iteration using the Deck you instantiated.
5. Using a traditional for loop, iterate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

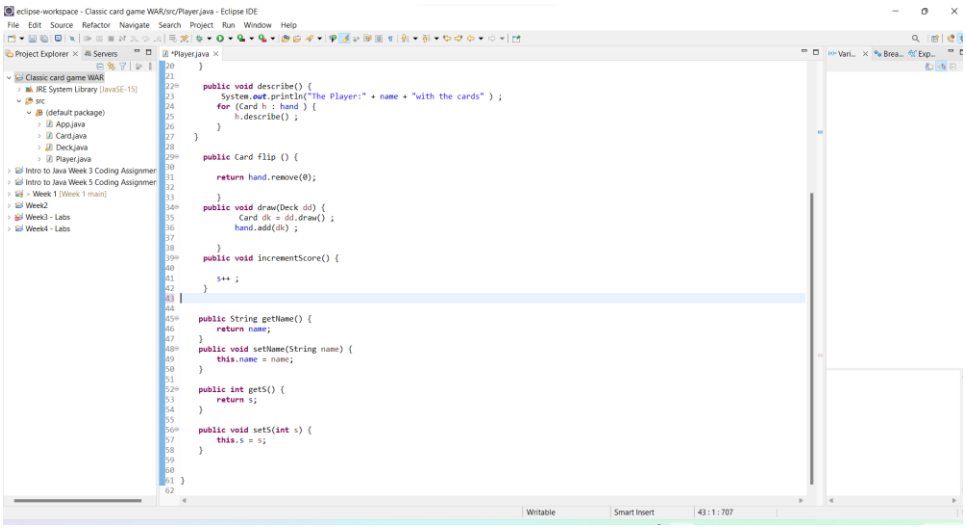
Screenshots of Code:

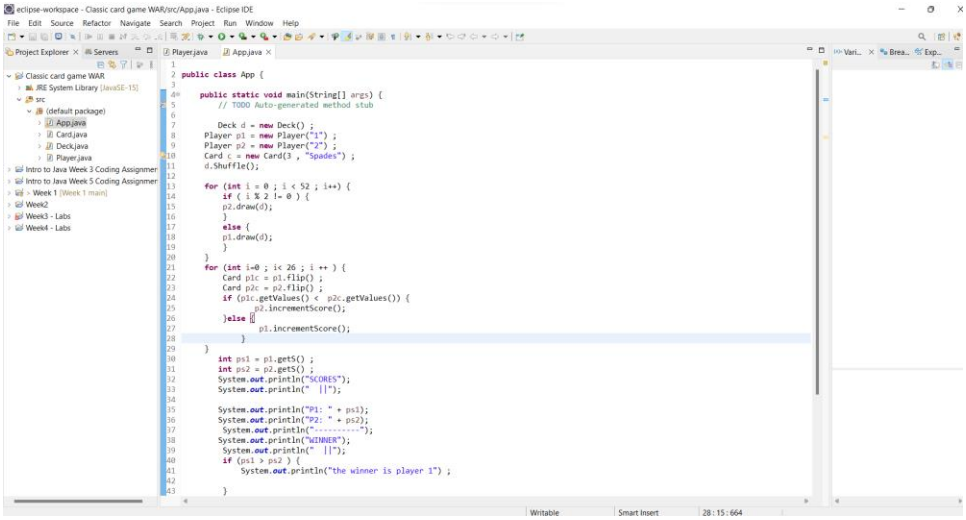

eclipse-workspace - Classic card game WAR/src/Card.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Servers
*Card.java
Deck.java
Player.java
App.java
Classic card game WAR
JRE System Library [JavaSE-15]
src
(default package)
App.java
Card.java
Deck.java
Player.java
Intro to Java Week 3 Coding Assignmer
Intro to Java Week 5 Coding Assignmer
Week 1 [Week 1 main]
Week2
Week3 - Labs
Week4 - Labs
public class Card {
private int values ; //= new String[] {"2","3","4","5","6","7","8","9","10","Jack","Queen","King","Ace"} ;
private String names ; //= new String[] {"Spades","Hearts","Diamonds","Clubs"} ;
public Card(int v, String S) {
values = v ;
if (v == 2) {
names = "TWO" ;
}
else if (v == 3) {
names = "Three" ;
}
else if (v == 4) {
names = "Four" ;
}
else if (v == 5) {
names = "Five" ;
}
else if (v == 6) {
names = "Six" ;
}
else if (v == 7) {
names = "Seven" ;
}
else if (v == 8) {
names = "Eight" ;
}
else if (v == 9) {
names = "Nine" ;
}
else if (v == 10) {
names = "Ten" ;
}
else if (v == 11) {
names = "Jack" ;
}
else if (v == 12) {
names = "Queen" ;
}
else if (v == 13) {
names = "King" ;
}

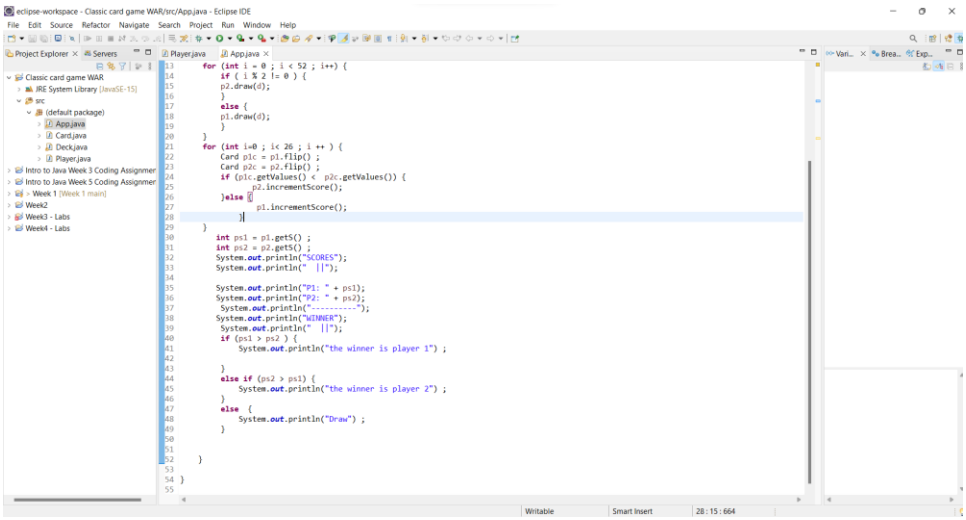

eclipse-workspace - Classic card game WAR/src/Card.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Servers
*Card.java
Deck.java
Player.java
App.java
Classic card game WAR
JRE System Library [JavaSE-15]
src
(default package)
App.java
Card.java
Deck.java
Player.java
Intro to Java Week 3 Coding Assignmer
Intro to Java Week 5 Coding Assignmer
Week 1 [Week 1 main]
Week2
Week3 - Labs
Week4 - Labs
else if (v == 9) {
names = "Nine" ;
}
else if (v == 10) {
names = "Ten" ;
}
else if (v == 11) {
names = "Jack" ;
}
else if (v == 12) {
names = "Queen" ;
}
else if (v == 13) {
names = "King" ;
}
else if (v == 14) {
names = "Ace" ;
}
names = names + " of " + S ;
}
public String describe() {
return "This is the card: " + names + " and value of: " + values ;
}
public int getValues() {
return values;
}
public void setValues(int values) {
this.values = values;
}
public String getNames() {
return names;
}
public void setNames(String names) {
this.names = names;
}
}
Writable
Smart Insert
4 : 89 : 228


eclipse-workspace - Classic card game WAR/src/Deck.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Servers
Deck.java
Player.java
Classic card game WAR
JRE System Library [JavaSE-15]
src
(default package)
App.java
Card.java
Deck.java
Player.java
Intro to Java Week 3 Coding Assignmer
Intro to Java Week 5 Coding Assignmer
Week 1 [Week 1 main]
Week2
Week3 - Labs
Week4 - Labs
import java.util.ArrayList;
public class Deck {
List<Card> cardList = new ArrayList<Card>() ;
List<String> names = new ArrayList<String>() ;
public Deck() {
names.add("Spades");
names.add("Hearts");
names.add("Diamonds");
names.add("Clubs");
for (int i = 0 ; i < 4 ; i++) {
for(int va=2 ; va<15 ; va++) {
cardList.add(new Card(va , names.get(i)));
}
}
}
public void Shuffle() {
Collections.shuffle(cardList) ;
}
public Card draw () {
return cardList.remove(0);
}
}

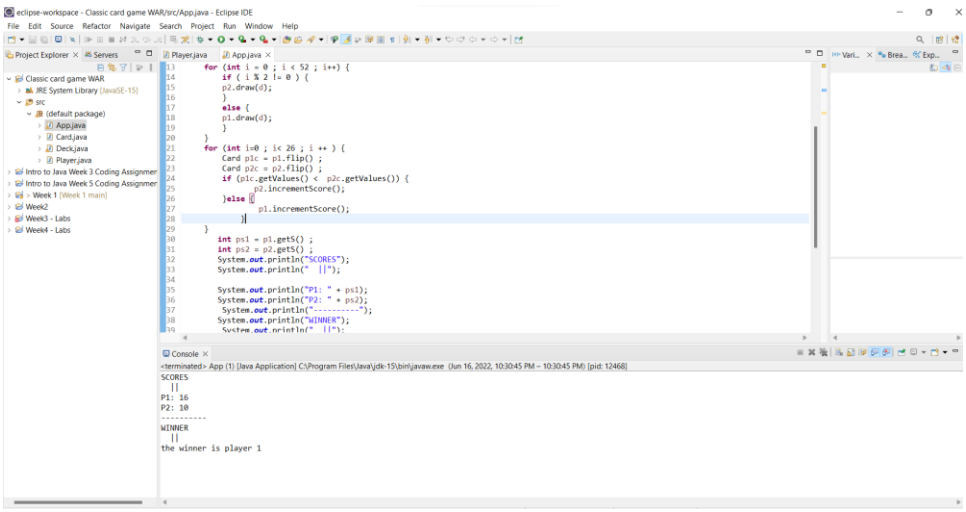

eclipse-workspace - Classic card game WAR/src/Player.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Servers
*Player.java
Classic card game WAR
JRE System Library [JavaSE-15]
src
(default package)
App.java
Card.java
Deck.java
Player.java
Intro to Java Week 3 Coding Assignmer
Intro to Java Week 5 Coding Assignmer
Week 1 [Week 1 main]
Week2
Week3 - Labs
Week4 - Labs
1 import java.util.ArrayList;
3
4 public class Player {
5
6 private List<Card> hand = new ArrayList<Card>() ;
7 private int s;
8
9
10 public Player() {
11 // TODO Auto-generated constructor stub
12 s = 0 ;
13 }
14
15 private String name ;
16 public Player (String n) {
17 s = 0 ;
18 name = n ;
19
20 }
21
22 public void describe() {
23 System.out.println("The Player:" + name + "with the cards") ;
24 for (Card h : hand) {
25 h.describe() ;
26 }
27 }
28
29 public Card flip () {
30
31 return hand.remove(0);
32
33 }
34 public void draw(Deck dd) {
35 Card dk = dd.draw() ;
36 hand.add(dk) ;
37
38 }
39 public void incrementScore() {
40
41 s++ ;
42 }
43
44


eclipse-workspace - Classic card game WAR/src/Player.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Servers
*Player.java
Classic card game WAR
JRE System Library [JavaSE-15]
src
(default package)
App.java
Card.java
Deck.java
Player.java
Intro to Java Week 3 Coding Assignmer
Intro to Java Week 5 Coding Assignmer
Week 1 [Week 1 main]
Week2
Week3 - Labs
Week4 - Labs
20 }
21
22 public void describe() {
23 System.out.println("The Player:" + name + "with the cards") ;
24 for (Card h : hand) {
25 h.describe() ;
26 }
27 }
28
29 public Card flip () {
30
31 return hand.remove(0);
32
33 }
34 public void draw(Deck dd) {
35 Card dk = dd.draw() ;
36 hand.add(dk) ;
37
38 }
39 public void incrementScore() {
40
41 s++ ;
42 }
43
44
45 public String getName() {
46 return name;
47 }
48 public void setName(String name) {
49 this.name = name;
50 }
51
52 public int getS() {
53 return s;
54 }
55
56 public void setS(int s) {
57 this.s = s;
58 }
59
60
61 }
62
Writable
Smart Insert
43 : 1 : 707





Screenshots of Running Application:



URL to GitHub Repository

https://github.com/cel90/Java-Final-Coding-Project