SUBMISSION INSTRUCTIONS

Send ONE *.zip or *.rar compressed file containing all your *.java files.
Submit your assignment through blackboard (submit assignment link).
Name your assignment file (the zip file) Assignment_Two_<student name>.
For example; Assignment_Two_Neilsen_Janeil.zip

GRADING MATRIX

The following provides you with feedback on each area you are graded on for the assignments in this course.

Trait	90% — 100%	70% — 89%	50% - 69%	0% - 49%
Specifications	The program works and meets all of the specifications.	The program works and produces the correct results and displays them correctly. It also meets most of the other specifications.	The program produces correct results but does not display them correctly.	The program is producing incorrect results.
Readability	The code is exceptionally well organized and very easy to follow.	The code is fairly easy to read.	The code is readable only by someone who knows what it is supposed to be doing.	The code is poorly organized and very difficult to read.
Reusability	The code could be reused as a whole or each routine could be reused.	Most of the code could be reused in other programs.	Some parts of the code could be reused in other programs.	The code is not organized for reusability.
Documentation	The documentation is well written and clearly explains what the code is accomplishing and how.	The documentation consists of embedded comment and some simple header documentation that is somewhat useful in understanding the code.	The documentation is simply comments embedded in the code with some simple header comments separating routines.	The documentation is simply comments embedded in the code and does not help the reader understand the code.
Delivery	The program was delivered on time.	The program was delivered within a week of the due date.	The code was within 2 weeks of the due date.	The code was more than 2 weeks overdue.
Efficiency	The code is extremely efficient without sacrificing readability and understanding.	The code is fairly efficient without sacrificing readability and understanding.	The code is brute force and unnecessarily long.	The code is huge and appears to be patched together.

ASSIGNMENT BACKGROUND

This assignment builds upon the lecture notes and chapters 4 to 6.

ASSIGNMENT PREPARATION

In order to prepare for this assignment question you should attempt:

- all debugging exercise in chapters 4 to 6
- complete 5 exercises from chapters 4 to 6 (odd questions)

ASSIGNMENT QUESTION – Game Zone Part 1

In the last assignment, you created a card class. Modify the card class so the setValue() method does not allow a card's value to be less than 1 or higher than 13. If the argument to setValue() is out of range, assign 1 to the card's value.

You also created a PickTwoCards application that randomly selects two playing cards and displays their values. In that application, all card objects were arbitrarily assigned a suit represented by a single character, but they could have different values, and the player observed which of two card objects had the higher value. Now, modify the application so the suit and the value both are chosen randomly. Using two card objects play a very simple version of the card game War. Deal two cards – one for the computer and one for the player – and determine the higher card, then display a message indicating whether the cards are equal, the computer won, or the player won. (Playing cards are considered equal when they have the same value, no matter what the suit is). For this game, assume the Ace (value 1) is low. Make sure that the two cards dealt are not the same card. For example, a deck cannot contain more than one card representing the 2 of spades. If two cards are chosen to have the same value, change the suit for one of them. Save the application as **War.java**

ASSIGNMENT QUESTION – EXAMPLE OUTPUT Part 1

Computer Player	**Human Player**	**Result**
11 of Clubs	11 of Spades	Tie
Computer Player	**Human Player**	**Result**
5 of Spades	3 of Spades	Computer wins

ASSIGNMENT QUESTION – Game Zone Part 2

Now modify the game using the newly modified card class so that when a tie is declared, that each player "puts down 10 cards each" and compares the 11th card to see if there is a clear winner. If there is a tie, repeat the process until there is a clear winner. The table below shows four typical executions. Recall that in this version of War, you assume that the ace is the lowest-valued card. Save the game as War2.java.

So I expect a Card.java, War.java and War2.java file; Each working off the other. No need to reinvent the wheel! Lastly I expect, when WAR is called, to see all ten cards displayed.

ASSIGNMENT QUESTION – EXAMPLE OUTPUT Part 2

Computer Player	**Human Player**	**Result**
11 of Clubs	11 of Spades	Tie
WAR is CALLED!		
10 of Spades	11 of Clubs	Discard
4 of Clubs	2 of Hearts	Discard
7 of Hearts	4 of Diamonds	Discard
10 of Clubs	5 of Clubs	Discard
2 of Clubs	7 of Diamonds	Discard
4 of Hearts	4 of Spades	Discard
7 of Diamonds	1 of Hearts	Discard
9 of Clubs	2 of Clubs	Discard
12 of Hearts	5 of Diamonds	Discard
2 of Spades	7 of Clubs	Discard
5 of Spades	3 of Spades	Computer wins