CSC 120
Programming Project #2

Due Date: ***by 9:30am on*** **Thursday, February 16th**

Objectives:

- To gain experience using the Eclipse IDE.
- To understand & develop program logic.
- To test code that contains assignment statements, object instantiations and method calls in the Card, Deck & GUI classes, as well as output statements.
- To add code to an existing program.
- To design test cases to make sure the program works properly.

Specifications:

Understanding Program Logic:

Download ***Project2.java*** and ***cardgames.jar*** from Blackboard. Save cardgames.jar into your workspace folder. A .jar file contains executable .class files (byte code) as well as other files such as images. Many user-defined classes are saved into a .jar file (also known as a library) which can be used by the IDE to access those classes. You will tell Eclipse how to access this .jar file during the project creation.

Creating a New Project and Modifying the Build Path

Begin Eclipse. Use the Project wizard to create a new project.

- From the main menu choose **File**. Choose **New**, choose **Java Project…**
- In the **Project Name:** text box type **proj2** ***(all lower-case letters)***
- Click **Next**

In the next step of project creation you will see a dialog box with multiple tabs. Click the **Libraries** tab to add the .jar file you copied earlier to the Eclipse workspace as a library.

- Click **Add External JARs**.
- Browse to locate the **cardgames.jar** file in the **workspace** directory and select it.
- Click **Open**.
- Click **Finish**.

Within this project create a class called ***Project2*** (the project name should begin with a capital letter, everything else is lower case, no spaces). Be careful to create your project and class correctly. Use lab1 and lab3 handouts as a reference. Remember to specify the package name as ***proj2*** when you create the Project2 class.

Open the ***Project2.java*** file you downloaded and copy/paste it into the class you just created.

Run the program. You should see two cards in the GUI window and the card information is displayed in the console window. Each time you run the program the two cards will not be the same because they are dealt from the deck which is re-shuffled each time the program is run.

Run the program 5 times. Write down the cards that are displayed and the output (pair or not a pair) each time the program is run in Table 1 on page 5.

Writing Java Statements:
The current program contains statements which deal and store a card, display it in the window and output the card information to the screen. Use the current code as reference to write statements to do each of the following:

- Deal another card – name the card ***card3***.
- Display the ***card3*** information on the screen.
- Add ***card3*** to the window.

Compile and run the program. Make sure the third card appears and that it displays prior to the pair/not pair output. The output should be similar to:

```
Card 1 is: 6 of hearts
Card 2 is: 12 of clubs
Card 3 is: 7 of hearts
Not a pair
```

Now that there are 3 cards, there are more combinations that need to be checked to determine if the cards contain a pair (two cards with the same value).

- Modify the if statement in the program to identify a pair. Use the ***If Statement Reference*** on Blackboard for examples of if statements.

Compile and run the program. Run the program several times to see if it appears to work properly. You can't thoroughly test the program because the deck randomly deals cards – you will address this after the remaining statements are written.

Use the current code to write statements to do each of the following. Compile and run the program after each bullet to see if it seems to works properly.

- Write if statements to determine and display the ***value*** of the highest of the three cards. If two cards have the same highest value, it should display that value. For example, if the cards are 10 of clubs, 3 of hearts and 10 of spades it should display 10.
- Calculate the sum of the three cards using the ***point*** value for each card. Display the sum of the ***point*** values. Read the documentation carefully to ensure you are using the correct method.
- If the sum of the point values is 25 or higher, the player wins, otherwise the computer wins. Display an appropriate message based upon the sum previously calculated.

This completed program should produce output in the following format. Since the values are randomly generated, realize that your output will vary each time the program is run.

```
Card 1 is: 6 of hearts
Card 2 is: 12 of clubs
Card 3 is: 7 of hearts
Not a pair
The highest value is 12
The sum is 23
Computer wins
```

Testing Your Program Logic:
Since the values of the cards are random, it is possible that in 5 executions you never had a pair or that you haven't had a sum greater than 25. Therefore, you may not have thoroughly tested the if statements to make sure the program is correctly identifying a pair, the highest value, and the sum.

To make sure your program works properly, it must be tested in a number of ways. Comment out the three lines of code where the deck deals a card (ie. the ***dealCard*** method calls) by placing // at the beginning of each line. Instead, you will specify what cards you want by calling the Card constructor instead of dealing from the Deck. The card constructor requires an integer as an argument (ie. ***new Card(5)***). The integer will determine the value and suit of the card. The list below represents several different cards that you might use for testing.

Argument	Card
5	5 of clubs
18	5 of diamonds
31	5 of hearts
44	5 of spades
11	11 of clubs
24	11 of diamonds
37	11 of hearts
50	11 of spades
14	1 of diamonds

For example, if you want card1 to be the 5 of hearts, then you would comment out

// Card card1 = theDeck.dealCard();

and type the following statement after it

Card card1 = new Card(31);

Fill in Table 2 on page 5 to indicate the cards (ie. 5 of hearts, 5 of spades, etc.) you used to test the various parts of the program. You need to test each possibility (for example, to determine a pair, card 1 and card 2 could be a pair, but what other cards could make a pair?). Be sure each test case tests a different combination (ie. don't test card 1 and card 2 being a pair twice, once with 5s and the second time as 11s). As you complete the table, think about what testing should be done in order to verify that the code works properly. For each test case, change the values of the cards, run the program and see what the output is and record the information.

Are the outcomes correct? If not, why not? If the problem is in the code, fix it so it works properly.

When you are done, place a comment at the beginning of your program that includes the ***title*** of the program (something like Project 2: Working with the Card class), a ***description*** of what the program does (this should be 2-3 sentences that explains what your program does as if you were talking to someone who doesn't know anything about programming) and your name as the ***author***. In addition, explain the code you added to the program by commenting each group of statements.

Citation Policy:

This assignment should be worked on individually to help you practice the material covered in class. However, you may work with other students as long as you cite the name(s) of who you worked with and all students understand the assignment submitted. Failure to cite other people is considered plagiarism. You should never email your program to another student or let someone take a picture of your code. You may receive help from your instructor or from the Computer Learning Center in B225.

Grading:

This assignment will be graded in 2 parts.

Part 1: Correctness of the tables and the program submitted. Does the program submitted meet the requirements? Does it display all information? Are the values correct? Have you commented your code? (60 pts)

Part 2: Grade on an in class quiz. On the due date, you will take a 10 minute quiz at the beginning of class based upon the assignment you submit. ***Bring a printout with line numbers*** to class with you to use as you take the quiz. The printout must match the program you uploaded to Blackboard and contain your name. Please make sure you are on time for class. Students who are late may not have enough time to complete the quiz or may miss it entirely. (40 pts)

Submission:

Hand in the following at the beginning of class on February 16th

- The answers to the questions on the sheet attached.
- Page 5 with the completed tables.
- The source code for the completed Project2.java.
- The output of Project2.java (should match the source code just printed).

Upload the Project2.java file for grading by 9:30am on February 16th.

CSC 120
Programming Project #2
Due: Thursday, February 16th

Name: _____

Table 1:

Card1	Card2	What is displayed on the screen? Pair or Not a Pair?

Table 2:

In each location, put the actual card (ie. 5 of hearts), not the number (ie. 31) that you used to test each combination. The outcome column should show what was actually displayed on the screen for that specific test (ie. you don't need to show all output just what is relevant for that test case).

	Card 1	Card 2	Card 3	Outcome
Pair cards 1 & 2				
Pair cards ___ & ___				
Pair cards ___ & ___				
Not a pair				
Highest value card 1				
Highest value card ___				
Highest value card ___				
Two values are highest cards ___ & ___				
Two values are lowest cards ___ & ___				
Sum < 25				
Sum = 25				
Sum > 25				

CSC 120
Programming Project #2
Due: Thursday, February 16th

Name: _____

How many hours did it take you to complete this assignment? (circle one)

- < 1
- 1-3
- 3-5
- 5-7
- > 7

Where did you work on this assignment? (circle all that apply)

- home
- school
- work

How did you go about getting help on this assignment? (circle all that apply)

- talked with an assistant in the computer learning center (who?)
- talked with my professor
- talked with another student
- read the textbook
- looked at notes from class
- looked at lab exercises
- looked at Oracle's web site
- other _____ (please specify)

Is there anything that doesn't work as you'd like or expect? If so, what?

Did you find anything difficult when you were writing this program? If so, please describe your difficulty here.