LAB #1
Introduction to Java Programming

(Random Number)

1. Start by creating a new project in Eclipse (refer to Lab 0 if you cannot remember how to do this), call the project something such as Lab1RandomNumber.

2. Create a new java file called <engr account username>Lab1.java, where <engr account username> is actually your engr account username (used to log into TEACH).
   (I would create a file called jessjoLab1.java)

3. Create a header similar to the one for Lab0, create the structure for the class we will be working in

```
/**
* Header
* Info
* Up
* Here
*/

/* My class would be called jessjoLab1,
*    this is because the name of the class and the name of the file it is in
*    need to have the same name (though the file should have.java at the end)
*/
public class <engr username>Lab1
{

}
```

4. Inside the class, create the main method we will use to make the program run
   (whenever you have some code inside another **block** of code (usually inside {}) be sure to indent it so that it is easier to read)

```
public static void main(String[] args)
  {

  }
```

5. Now double check that you created the file and program code structure correctly by tossing in a println, saving, and running it to see that it compiles and executes the way you expect it to.

```
 System.out.println("Did it work?");
```

   At this point you should have something that looks an awful lot like your hello world program from Lab0:

```
/**
        * Header
* Info
* Up
* Here
*/
public class <engr username>Lab1
{
   public static void main(String[] args)
      {
         System.out.println("Did it work?");
      }
}
```

From here on in I will just use jessjo as my username and you can substitute yours in in place of mine when necessary

6. Now lets use the important method of the lab: random.
   Before the println but after the **method header** (*public static void main(String[] args)*) and curly brace (sometimes just called braces, sometimes curly bracket, and as the chain of my previous department called them, sideways seagulls) type the following:

   ```
   int value = (int)(Math.random() * 52);
   ```

   There are several things going on here:
   - first, a variable of type int called value is being declared,
   - it is then being initialized to the value (int)(Math.random() * 52)
     - this value is created by first calling a method in the class called Math,
       - (you can guess it is a class because it starts with a capital letter)
     - the method is called random,
       - (you can tell it is a method because of the parentheses)
     - Math.random() returns a double value in the range from 0 to slightly less than 1
       - (you can look this up in the java API by searching the internet for "java 6 Math", then searching the oracle API page that should come up for the method random)
     - then whatever value that the method returns is being multiplied by 52,
       - (no trick, just a double value from 0 to less-than-1 being multiplied by 52)
     - this value is then type-cast into an int value
       - (this means that the fractional value after the radix point is lost, but the whole number value remains)
     - then lastly, this value is stored into the variable called value.

   This step is really full of a bunch of ideas, try doing these things after you get a print statement in your code to see what happens:
   - without the type-cast (int in parentheses),
     - (to solve the problem that comes up, think of what type you are now dealing with)
   - with an int or double variable you choose instead of the Math.random() method call,
   - with a value other than 52, such as 10, 100, or maybe 1000,
     - (then run the program a few times to see what values come up)

7. next lets create a Scanner object to use for getting user input.

   First, since I know the Scanner class is not part of the standard library (the stuff you can access without telling the compiler where your method were declared, such as the Math class we used earlier) we will tell the compiler that we want to use the Scanner class to create objects from it.

   Near the top of your code, before the **class header** (*public class jessjoLab1*, remember my username would be jessjo, yours will be different), but after your program header (I care about seeing who made it and what the program is about more than I care about what classes it uses internally) type the following:

```
import java.util.Scanner;
```

Notice that the word import changed color similar to when some other words change color, this denotes that the word import is a reserved keyword (you can look up all the reserved keywords on the Internet, but I like to note to my students that one of the keywords is goto, now go find out what it does :))

now that we told the compiler we might use the Scanner class, lets actually use it. Type the following just below your random number from earlier:

```
Scanner input = new Scanner(System.in);
```

There are a couple things happening here, first we declare an object of type *Scanner*, called input*, then we assign it to be the object created by the Scanner **constructor** that takes some sort of input **stream****.

*:(this Scanner object could just as easily be called: myInput, in, keyboard, scanner, scan, or even fred or sally, remember that variable names can be nearly anything, but that we choose to use names that we hope will help others read our code)

**:(you do not need to know what a constructor is, though you could probably guess it create objects somehow, and you do not need to know what a stream is).

This Scanner object will allow us to take the input stream (again no need to know the details, just know that it could be taken in from a keyboard, a file on disk, a network connection, or other similar sources of data) from the keyboard, and get useful pieces of data from the user of the program to put into variables and do computations.

8. Lets now ask the user for some input, store the decision in a variable, and do some calculation based on that input. Knowing that I want to store the response from the user I will make a variable to hold the actual response. Type the following below the Scanner declaration:

```
int selection = 0;
System.out.println("Please select an option from the following menu:");
System.out.println("[1]: View a random number from 1 to 52 (inclusinve)");
System.out.println("[2]: View a random card by suit and value");
System.out.println("Enter an integer value of 1 or 2, anything else will cause the program to exit.");
```

This creates a variable to hold user input then prints out some text to let the user know what kind of input we will be looking for in a moment.

Try it to see if it prints out your new menu.

Continue by typing:

```
if(input.hasNextInt())
selection = input.nextInt();
```

This checks if the variable input (the Scanner object we created) has an int next in its input stream, and if it does, then it takes that int and stores it in the variable called selection.

Continue by typing:

```
if(selection != 1 && selection != 2)
 {
 System.out.println("Sorry, that is an invalid entry. Exiting program");
 System.exit(0);
 }
```

Checking that user input matches a certain form, type, or range is very important in reducing the number of errors your program has, we will revisit this a couple other times in the future as well.

```
if(selection == 2)
 {
 int suit = value % 4 + 1;
 value /= 4;
 System.out.println("Suit = " + suit);
            }
            value += 1;

            System.out.println("Value = " + value);
```

This part is a combination of a couple actions, at first I might consider making an if statement for each version of the input from the menu above, but after I type this in and look at it some I notice that either way I will be printing something related to the value of the random number generated earlier.

So I check for the special case, which will decide whether I will be printing one output or two,

I then find the suit of the card by taking the random number mod 4 to determine which suit my card falls into (since the number mod 4 will guarantee that it is in one of the four suits I want, and equally divide the cards among the suits), though to print this for the user I think to myself that it might be easier to think of the suits from 1 to 4 rather than the 0 through 3 that the mod operator would give me.

I then do integer division by 4 to get the number in the suit of the random number (the way I represent this was not listed in the requirements so I decided that I would think of the cards as 13 groups of 4. Though I could have just as easily found the value first and the suit second by using mod 13 and integer division by 13 instead: **after you get the lab working in my way you should try this to see if you can get it to work**).

I similarly need to add 1 to the value if I want the values to be from 1 to 13 instead of from 0 to 12 (often a change that needs to be made in math and computer science).

I then print the value, which will be correct if I decide to just show the value from 1 to 52, or if I chose to show the suit and the value.

Save and print the code and see if it worked

9. This gives us the following code, which you should have checked to see if it was working periodically along the way.

```
/**
* Header
* Info
* Up
* Here
*/

import java.util.Scanner;

public class Lab1
{
    public static void main(String[] args)
      {
           int value = (int)(Math.random() * 52);
           Scanner input = new Scanner(System.in);
           int selection = 0;
           System.out.println("Please select an option from the following menu:");
           System.out.println("[1]: View a random number from 1 to 52 (inclusinve)");
           System.out.println("[2]: View a random card by suit and value");
           System.out.println("Enter an integer value of 1 or 2, anything else will cause the program to exit.");

           if(input.hasNextInt())
                   selection = input.nextInt();
           if(selection != 1 && selection != 2)
           {
                   System.out.println("Sorry, that is an invalid entry. Exiting program");
                   System.exit(0);
           }
           if(selection == 2)
           {
                   int suit = value % 4 + 1;
                   value /= 4;
                   System.out.println("Suit = " + suit);
           }
           value += 1;

           System.out.println("Value = " + value);
      }
  }
```

10. We already had a way to check if our data was an int, but why not use a better way of handling input errors than just quitting?

    Instead of handling the error by checking it with an if statement and then exiting the program, why not put in something that will instead repeatedly pester the user for more input as long as they do not enter good input (or quit).

    Lets replace the if-statement and its block of code:

```
if(selection != 1 && selection != 2)
{
        System.out.println("Sorry, that is an invalid entry. Exiting program");
        System.exit(0);
}
```

with a loop that repeatedly asks for input and checks it as long as the input is either bad or a signal to quit:

```
while(selection != 1 && selection != 2 && selection != 3)
{
        System.out.println("Sorry, that is an invalid entry. Please try again");
        System.out.println("Please select an option from the following menu:");
        System.out.println("[1]: View a random number from 1 to 52 (inclusinve)");
        System.out.println("[2]: View a random card by suit and value");
        System.out.println("[3]: Quit program");
        System.out.println("Enter an integer value from 1 to 3");
        if(input.hasNextInt())
                selection = input.nextInt();
}

if(selection == 3)
{
        System.out.println("You entered a 3, now quitting");
        System.exit(0);
}
```

A couple important notes:

This is called a *verification of input loop* (using a **while loop** in this case)

Remember to print the menu again in case the bad input has caused the printout to go off the screen

Remember to actually get a new input, otherwise the value getting you into this loop will not change and allow you to get back out.

This means we can also remove the part of the original menu that says that any other input will cause the program to exit, and include an option to quit the program without printing a value.

11. (optional to begin with) The last thing we will do for this lab is allow the user to repeatedly use the program without exiting.

    We can do this by putting a loop of some kind around the whole body of the code that we want to repeat. In this case though we want our code to execute at least once, so we could use a do while loop (though I normally prefer to use while loops and carefully structured conditions), though we need to make sure whatever variable we plan to check in our loop is initialized to something before the loop begins.

    Which could change our code to look like this:

```
int again = 0;
do
{

//the rest of our code...

        do
        {
                System.out.println("Select an option from the following menu to repeat the program:");
                System.out.println("[1]: Run the program again");
                System.out.println("[2]: Please stop drawing random cards");
                System.out.println("Enter an integer value of 1 or 2");
                if(input.hasNextInt())
                        again = input.nextInt();
                if(again !=1 && again != 2)
                        System.out.println("Sorry, that is not a valid answer");
        } while(again != 1 && again != 2);
} while(again != 2);
```

12. Be sure your code works, then turn in your lab (.java files).

The final state of my code for this lab is:

```
/**
* Header
* Info
* Up
* Here
*/

import java.util.Scanner;

public class Lab1
{
   public static void main(String[] args)
     {
         int again = 1;
         do
         {
                 int value = (int)(Math.random() * 52);
                 Scanner input = new Scanner(System.in);
                 int selection = 0;
                 System.out.println("Please select an option from the following menu:");
                 System.out.println("[1]: View a random number from 1 to 52 (inclusinve)");
                 System.out.println("[2]: View a random card by suit and value");
                 System.out.println("Enter an integer value of 1 or 2");

                 if(input.hasNextInt())
                         selection = input.nextInt();
                 while(selection != 1 && selection != 2 && selection != 3)
                 {
                         System.out.println("Sorry, that is an invalid entry. Please try again");
                         System.out.println("Please select an option from the following menu:");
                         System.out.println("[1]: View a random number from 1 to 52 (inclusinve)");
                         System.out.println("[2]: View a random card by suit and value");
                         System.out.println("[3]: Quit program");
                         System.out.println("Enter an integer value from 1 to 3");
                         if(input.hasNextInt())
                                 selection = input.nextInt();
                 }

                 if(selection == 3)
                 {
                         System.out.println("You entered a 3, now quitting");
                         System.exit(0);
                 }

                 if(selection == 2)
                 {
                         int suit = value % 4 + 1;
                         value /= 4;
                         System.out.println("Suit = " + suit);
                 }
                 value += 1;
```

```
                System.out.println("Value = " + value);
                do
                {
                        System.out.println("Select an option from the following menu to repeat the program:");
                        System.out.println("[1]: Run the program again");
                        System.out.println("[2]: Please stop drawing random cards");
                        System.out.println("Enter an integer value of 1 or 2");
                        if(input.hasNextInt())
                                again = input.nextInt();
                        if(again !=1 && again != 2)
                                System.out.println("Sorry, that is not a valid answer");
                } while(again != 1 && again != 2);
        }while(again == 1);
    }
}
```