AP Computer Science
Homework Set 1
Fundamentals

P1A. Using "MyFirstApp.java" as a model, write a similar program, "MySecondApp.java", that prints your "favorites". Your program should do the following:

    a.   create and initialize an `int` with your favorite integer,
    b.   create and initialize a `double` with your favorite real number,
    c.   create and initialize a `boolean` with your favorite boolean value,
    d.   create and initailize a `String` with your favorite musical group.

Print each one of these on a separate line with identifying text.

P1B. In this program we will use Java's mathematical operators to calculate the roots of a polynomial using the quadratic formula. Recall that if we have:

$$ax^2 + bx + c = 0$$

then the roots of the equation are given by:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Here the specs for your program:

a.   Create three variables of type `int` and assign them numbers of your choosing (here's a good set to start with: a = 1, b = -2, c = -15),

b.   Calculate and print the value "$\sqrt{b^2 - 4ac}$" using `System.out.println()`. There is a built in function, `Math.sqrt()` that will return the square root of a number. For example, the statement `Math.sqrt( 9 )` will return "3".

c.   Calculate and print the <u>first</u> root of the equation using the quadratic formula.

d.   Calculate and print the <u>second</u> root of the equation using the quadratic formula.

When printing your roots, you may get the message: "`NaN`", which is an acronym for "`Not a Number`". This means that there is a mathematical calculation error of some sort in your program. In this program, this is most likely due to taking the square root of a negative number (i.e., negative discriminant b^2-4ac).

e.   Use an "if-else" statement to determine if the discriminant is negative BEFORE we take the square root of it. If the discriminant IS positive, the two real roots will be printed using `System.out.println()` statements as shown below:
```
The value of root 1 is = 5
The value of root 2 is = -3
```

f.   If the discriminant is negative , then the following statement (or a clever facsimile of) should be printed:
```
  "Sorry, your discriminant is negative. We cannot
process imaginary root right now…"
```

The following values for a, b, and c will yield a negative discriminant:
a = 1
b = 1
c = 1

P1C. Let's write a program that uses an <u>if-else statement</u> that will print whether a number is an even OR an odd number.

Here's some sample output if the number is 15 (or any odd number):
```
15 is an odd number
```

And if the value is 8 (or any even number):
```
8 is an even number
```

Note: you can use the "modulus" or "remainder" operator (%) to determine the remainder of a division. For example, if the following lines of code is executed, the value of "remainder1" will be "1", since 2 goes into 15 seven times with a remainder of 1. However, the value of "remainder2" will be "0" since 2 goes into 16 eight times with a remainder of "0".

By definition, 0 % (any number) = 0. For example, 0 % 3 = 0, 0 % 22 = 0, etc.

```
int remainder1 = 15 % 2;        // remainder1 is 1
int remainder2 = 16 % 2;        // remainder2 is 0
int remainder3 = 17 % 2;        // remainder3 is 1
int remainder4 = 18 % 2;        // remainder4 is 0
```

The modulus operator is helpful for finding remainders, but it is also the foundation for "public-key cryptography" which allows us to send information such as grade transcripts, online purchases, credit card payments, and bank statements securely through the Internet.

P1D. Let's write a program that uses a <u>for loop</u> to print numbers 1 through 10 along with its square and a random number between the number and its square inclusive, each separated by a tab. In Java, you can create a tab by using the escape code "\t" as shown below:

```
System.out.println( number + "\t" + numberSquared);
```

Here's some sample output:

```
1    1    1
2    4    2
3    9    5
4    16   11
5    25   22
6    36   32
7    49   49
8    64   55
9    81   56
10   100  67
```

11 is a random number between 4 and 16 (inclusive)

49 is a random number between 7 and 49 (inclusive)

P1E. Now we'll write a program to determine the number of digits in a given integer. Create an integer called "`number`" and place any number of your choosing in the variable. Your program should print the number of digits in the variable "`number`". Hint: use a loop and integer division to in this algorithm.

P1F. Let's write a program that creates an <u>array of Strings</u> of your top three favorite movies and an <u>array of Strings</u> of your top three favorite songs. Then use a for loop to print out your three favorite movies **on separate lines** AND a while loop to print out your three favorite songs all **on the same line**, separated by commas. Leave two blank lines between the movies and the songs. Below are some basics for creating, assigning values to, and printing array elements:

1.   You can create an array that can hold three `String` objects with the following line of code:
```
String[] movies = new String[3];
```

2.   You can assign values to individual elements of the array with the following line of code:
```
movies[0] = new String( "Star Wars: Episode 4" );
```

3.   Finally, you can print all elements of an array using a for loop as shown below:
```
for( int index = 0; index < movies.length; index++ )
  {
  System.out.println( movies[index] );
  } // end for loop
```

4.   See LewTube for an example of how to use while loops

P1G. Write a program that creates an array that can hold 10 integer values. Then complete the following tasks:

a.   Using a for loop and `Math.random()`, fill the array with random integer values between 0 and 100 inclusive.

b.   Use a second for loop to print each of the numbers in the array with a single space between each number.

c.   Use a third for loop to add up the numbers in array. Print the sum on a new line after the third for loop.

Note: When using a for loop with arrays, use the variable name "index" as the counter in the loop as shown below. This reinforces the fact that integer number that is changing represents the index of an array element.

```
for( int index = 0, index < myArray.length; index++ )
{
// body of your for loop code goes here
} // end for
```

d.   Finally, write a fourth for loop to print the array in reverse order. Hint: you might have to change the initialization, test, and/or increment parts of the for loop.
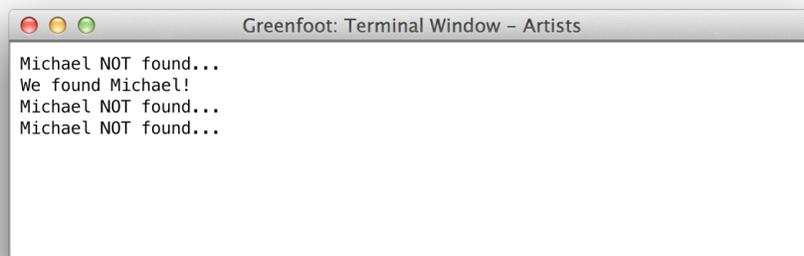
P1H. Write a program that creates an array that can hold 10 integer values. Then fill and print the array according the following specifications:

a.   Use a for loop and `Math.random()` to fill the array with random integer values between 0 and 100 THAT CAN BE EITHER POSITIVE OR NEGATIVE.

b.   Use a separate while loop to print each of the numbers in the array on a separate line.

c.   Use a second for loop to add the value of "10" to each element in the array. Be sure to actually change the values in the array as opposed to simply printing out the number in the array+10.

d.   The program should then print out the number of positive integers in the array.

e.   Use a third for loop to traverse the array and count the number of positive integers in the array.

P1I. Write a program that creates an array that can hold four Strings. Populate the array with <u>four</u> of your favorite musical artists. Below is a sample set of artists:

```
// feel free to use your own favorite artists
artists[0] = new String( "Mozart" );
artists[1] = new String( "Michael Jackson" );
artists[2] = new String( "Led Zeppelin );
artists[3] = new String( "Rush" );
```

a.   Use a for loop to traverse the array to attempt find "Michael Jackson". Print a statement for each corresponding element in the array stating whether or not you found Michael as shown below:

```
● ● ●              Greenfoot: Terminal Window – Artists

Michael NOT found...
We found Michael!
Michael NOT found...
Michael NOT found...
```

Check out the LewTube video on how to use the `.equals()` method of the `String` class to check for the "equality" of `Strings`.

Be sure to write your program so that if the number of elements in the array is changed, the traversal and "locating" code still works without having to make changes to your code, i.e., your code does not "break". The phrase "breaking your code" or "broken code" is used when changes to your data (length of an array or contents of an array among other things) causes incorrect output by your program.

b.   Now, leave one element in the array "empty" by place the value of "null" into the array element. For example, the following line of code will place the value of "null" into element 2.

```
artists[2] = null;   // note that there are no quotes
                     // around null; it represents a value of
                     // "nothing" but can be checked using if
                     // statements
```

Now attempt to find "Michael Jackson" again. What type of "error" do you get? Using comments, write a brief explanation of what you think this "error" means. How could this "error" be avoided so that your program does not crash?

P1J. Write a program to simulate the login process for a website. Allow the user to input a `String` for the <u>username</u> and an `int` for a <u>password</u> using a `JOptionPane` as shown below. We will address more complex passwords later.

(See LewTube for a tutorial on how to create and use "dialog boxes" as shown above)

If the username AND password match the correct values, then print the following statement (or a reasonable facsimile of) using `System.out.println()`:

`"You have successfully logged in"`

If the username/password combination does not match, print the following statement (or a reasonable facsimile of) using `System.out.println()`:

`"Login unsuccessful…try again"`

Learning Objective Checklist
(please print and complete after you have had all HW Set 1 programs checked off)

Place a check next to those items that you have mastered

	Define and initialize a primitive of type `int`.
	Define and initialize a primitive of type `double`.
	Define and initialize a primitive of type `boolean`.
	Define and initialize an object of type `String`.
	Define and initialize an object of array of type `int`, `double`, `boolean`, or `String`.
	Initialize and access array elements using `[ ]` notation.
	Write `if` and `if-else` statements to analyze `boolean` statements.
	Write a `for loop` to traverse and access array elements "forward" starting with index = 0
	Write a to traverse and access array elements "in reverse" starting with index = `array.length-1`
	Write a `while` loop to traverse and access array elements.
	Use the relational operators >, <, >=, <=, = =, and `!=` within the context of an `if` statement, `for` loop, and `while` loop.
	Use the `boolean` operators `&&` and `||` within the context of `if` statements and `while` loops.
	Use a "counter" variable to count the number of elements in an array that meet a given criteria.
	Use a "sum" variable to add any or all of the values in the array
	Write comments using `//`.
	Write the Java code to print output using "`System.out.println()`".
	Write code using `Math.random()` that returns a double precision number (double) from 0 inclusive to 1 exclusive.
	Use the `(int)` cast to "cast" or convert a double precision number into an `int`
	Write the Java code to create random numbers in any given range using `Math.random()`.

Name: _____

Date: _____