Java Programming Chapter 9 Check-off Name:_____

Fibonacci - A. 9.1a (Complete directions are in your text.)

Assignment:

1. Write a class ***Fibonacci*** that has a recursive method ***fib*** that takes in a single integer (x >= 0) and returns the appropriate Fibonacci number of the Fibonacci number series.
2. Run the program for the following numbers: `0, 3, 11`

Instructions:

1. Before submitting your code for grading make sure your name and period are documented near the top of your source code.
2. Your code must include the following:

    a. Your name and period
    b. Proper and descriptive variable names. Variable names must follow convention.
    c. Proper indentation
    d. Javadoc style comments – Comments the class and each method along with a description of the parameters and return values.

3. After your code passes CodeCheck scroll to the bottom, hit the Download link to download your files. This will create a zipped file with your code and output. DONOT alter this file in anyway. Any tampering with this file will result in a 0 for the lab. Put this file in your shared Google Drive folder for grading.

Multiplication - A. 9.1b

Assignment:

1. Write a class called ***Multiplication*** that has a recursive method called ***mult*** that solves a multiplication problem recursively.
2. The method takes in two arguments – the numbers to be multiplied.
   Use these sample run output values:
   `0 * 4 , 3 * 1 , 7 * 8 , 5 * 0 , 45 * 11`

Instructions:

1. Before submitting your code for grading make sure your name and period are documented near the top of your source code.

4. Your code must include the following:

    a. Your name and period
    b. Proper and descriptive variable names. Variable names must follow convention.
    c. Proper indentation
    d. Javadoc style comments – Comments the class and each method along with a description of the parameters and return values.

5. After your code passes CodeCheck scroll to the bottom, hit the Download link to download your

files. This will create a zipped file with your code and output. DONOT alter this file in anyway. Any tampering with this file will result in a 0 for the lab. Put this file in your shared Google Drive folder for grading.

DigitMatch - A. 9.2

Assignment:

1. Write a class called ***DigitMatch*** that has a recursive method called ***countMatch*** that counts the matching digits recursively.
2. The method accepts two non-negative integers as parameters and returns the number of digits that match between them. Two digits match if they are equal and have the same position relative to the end of the number (i.e. starting with the ones digit). In other words, the method should compare the last digits of each number, the second-to-last digits of each number, the third-to-last digits of each number, and so forth, counting how many pairs match. For example, for the call of countMatch(1072503891, 62530841), the method would compare as follows:

```
1 0 7 2 5 0 3 8 9 1
    | | | |   | | | |
  6 2 5 3 0 8 4 1
```

3. The method should return 4 in this case because 4 of these pairs match (2-2, 5-5, 8-8, and 1-1). Below are more examples:

Call	Value Returned
countMatch(38, 34)	1
countMatch(5, 5552)	0
countMatch(892, 892)	3
countMatch(298892, 7892)	3
countMatch(380, 0)	1
countMatch(123456, 654321)	0
countMatch(1234567, 67)	2

Instructions:

1. Before submitting your code for grading make sure your name and period are documented near the top of your source code.
2. Your code must include the following:

    a. Your name and period
    b. Proper and descriptive variable names. Variable names must follow convention.
    c. Proper indentation
    d. Javadoc style comments – Comments the class and each method along with a description of the parameters and return values.

3. After your code passes CodeCheck scroll to the bottom, hit the Download link to download your files. This will create a zipped file with your code and output. DONOT alter this file in anyway. Any tampering with this file will result in a 0 for the lab. Put this file in your shared Google Drive

folder for grading.

WriteSequence - A. 9.3

Assignment:

1. Write a class called ***WriteSequence*** that has a ***write*** method that accepts an integer n as a parameter and prints a symmetric sequence of n numbers with descending integers ending in 1 followed by ascending integers beginning with 1, as in the table below:

Call	Output
`write (1)`	`1`
`write (2)`	`1 1`
`write (3)`	`2 1 2`
`write (4)`	`2 1 1 2`
`write (5)`	`3 2 1 2 3`
`write (6)`	`3 2 1 1 2 3`
`write (7)`	`4 3 2 1 2 3 4`
`write (8)`	`4 3 2 1 1 2 3 4`
`write (9)`	`5 4 3 2 1 2 3 4 5`
`write (10)`	`5 4 3 2 1 1 2 3 4 5`

Notice that for odd numbers the sequence has a single 1 in the middle while for even values it has two 1s in the middle. A client using this method would have to call **`println`** to complete the line of output

Instructions:

1. Before submitting your code for grading make sure your name and period are documented near the top of your source code.
2. Your code must include the following:

    a. Your name and period
    b. Proper and descriptive variable names. Variable names must follow convention.
    c. Proper indentation
    d. Javadoc style comments – Comments the class and each method along with a description of the parameters and return values.

3. After your code passes CodeCheck scroll to the bottom, hit the Download link to download your files. This will create a zipped file with your code and output. DONOT alter this file in anyway. Any tampering with this file will result in a 0 for the lab. Put this file in your shared Google Drive folder for grading.

DigitCount - A. 9.4

Assignment:

1. Write a class called ***DigitCount*** that generates a random number and counts the digits in the number. It has three methods:
   a. A constructor that takes a seed and initializes a random number generator to the given seed
   b. Method ***generateNewNumber(int max)*** generates a new random number between 0 and max-1 and returns it
   c. Method ***countDigits(int num)*** returns the number of digits in num.

Here is a sample output:

```
Number 6159 has 4 digits
Number 27 has 2 digits
Number 7 has 1 digits
```

Instructions:

1. Before submitting your code for grading make sure your name and period are documented near the top of your source code.
2. Your code must include the following:

   a. Your name and period
   b. Proper and descriptive variable names. Variable names must follow convention.
   c. Proper indentation
   d. Javadoc style comments – Comments the class and each method along with a description of the parameters and return values.

3. After your code passes CodeCheck scroll to the bottom, hit the Download link to download your files. This will create a zipped file with your code and output. DONOT alter this file in anyway. Any tampering with this file will result in a 0 for the lab. Put this file in your shared Google Drive folder for grading.