**Project 3 Expectations**

Please review ***General Programming Expectations*** document located in Week 2 under **Week 2 Checklist**.

**Obtaining data for Assignment 3**

All the input data for this assignment is entered in the TextField labeled ***Enter n:*** of the GUI window displayed in the **Project 3.pdf** document. There are two types of output data for this assignment: Result and Efficiency. The first 11 values displayed int the ***Result:*** TextField are displayed below in Processing Values. The challenge for this assignment are the values that display in the ***Efficiency:*** TextField.

The assignment offers a process to generate a set of results based on a formula. Using the set of instructions in **Project3.pdf**, the following are the first 11 generations of Processing Values:

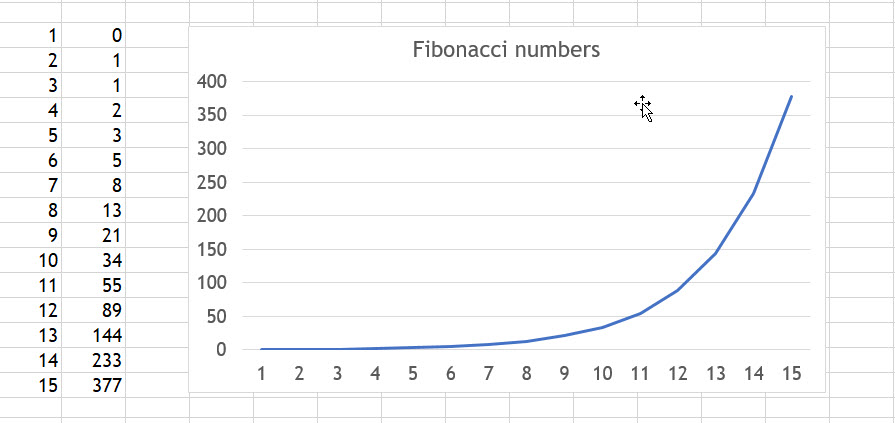
0th = 0  
1st = 1  
2nd = 2; [(2\*1 )+ 0 = 2]  
3rd = 5; [(2\*2) + 1 = 5]  
4th = 12;[(2\*5) + 2 = 12]  
5th = 29; [(2\*12) + 5 = 29]  
6th = 70; [(2\*29) + 12 = 70]  
7th = 169; [(2\*70) + 29 = 169]  
8th = 408; [(2\*169) + 70 = 408]  
9th = 985; [(2\*408) + 169 = 985]  
10th = 2378; [(2\*985) + 408 = 2378]

There are two ways to process this formula: recursive or interactive.

When designing the program, there is a main () method that calls the GUI window. If the **Iterative** radio button is selected, the *computerIterative* method is called when the **Compute** button is clicked. When the **Recursive** radio button is selected, the *computerRecursive* method is called when the **Compute** button is selected. Within each of these methods, there is a call to generate the value displayed in the ***Result:*** TextField and another call to generate the value in the ***Efficiency:*** TextField.

Each time the user enters an ***n*** value and clicks the **Compute** button, the resulting three values are entered into a text file, ***Efficiences.txt***. In addition to these three value the program should insert either the text of *interactive* or *recursive* as the first value of four entries.

After closing the program, take the values of the from the text file, copy them into an Excel spreadsheet. Create two scatter or line graphs with one line for the efficiencies of recursive and interactive the other showing the results. The example below displays a line graph for Fibonacci Numbers:



Reference to Fibonacci Numbers:

<https://io9.gizmodo.com/5985588/15-uncanny-examples-of-the-golden-ratio-in-nature>

* Provides an event handler to handle the Compute button click and another handler will be needed to produce the file described above when the window is closed. The latter handler is an object of an inner class that extends the WindowAdapter class.

**YouTube Discussion of Recursion:**

These might be helpful:

<https://www.youtube.com/watch?v=uCigwawdAnU>

<https://www.youtube.com/watch?v=ttTH_WX-Cbo>

**A reference from a website:**

<https://www.greeksforgeeks.org/pell-number/>

**The classic Recursion Problem**

The 'Towers of **Hanoi**' is a classical problem used to illustrate the power of recursion. The puzzle goes as follows. There are three poles and 64 discs of varied sizes. Initially, all the discs are placed on the first pole with the largest disc at the bottom and the smallest one at the top.

<http://www.javawithus.com/programs/towers-of-hanoi>

<https://www.tutorialspoint.com/javaexamples/method_tower.htm>

<http://simpledeveloper.com/towers-of-hanoi/>

**Suggested Design for Assignment 3**

Here is one skeleton design of this program.

package cmis242.prj3smithx;

[imports here]

public class CMIS242 PRJ3SmithX {

public static void WriteFile () [

[Code to open, write, and close file]

} // end WriteFile

public static class GUI {

*[Constructor ()]*

*[Call to computerIterative]*

*[Call to computerRecursive]*

*[Call to getEfficiency when appropriate]*

} // end public static class GUI

public static class Sequence {

static Integer *computerIterative* {

*[Calculation for n , counter, and, write to text]*

} // end static Integer *computerIterative*

static Integer *computerRecursive* {

*[Calculation for n, counter, and write to text]*

} // end static Integer *computerRecursive*

static Integer getEfficiency () {

[counter and write to text]

}// end static Integer getEfficiency

} // public static class Sequence

public static void main (String args[]) {

*[Declare instance of class]*

*[Open text file]*

*[Close text file]*

} // end public static void main (String args []) {

} // end public class CMIS242 PRJ3SmithX

**There are some considerations in designing the program**:

* Add Exit/End button should be added to the window.
* It is assumed that the user will not enter the n value in a consistent ascending order. Write all the values as the user enters them into the text file, ***Efficiencies.txt***.
* When the text file is closed and copied into the spreadsheet, the data can be sorted by type of processing and the efficiency values in ascending order. Name the spreadsheet ***Efficiencies.xlsx***.
* The table for the line graph should consist of n and the value for efficiency for each type of processing.
* If there is no access to Excel, then a spreadsheet available can be used. The bottom line is that there is a line graph displaying the different efficiencies between the initiative and recursive processing.

**This is the process for the assignment:**

1. Execute the java program.
2. The java program will generate a text file using the write process of java.
3. The text file will contain the necessary information for each of the two modules.
4. Copy the data from the text file into an Excel spreadsheet.
5. Use the data from the table pasted into the Excel spreadsheet to generate the appropriate graph.

The text file is an attachment to the Assignment folder. The table/graph can be its own attachment to the Assignment folder or it can be copied into the Test Plan document inserted at the end of the document.

**The Graph with wrong numbers for recursive:**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |
| n | Interactive | Recursive |  |  | | | | | | | |
| 0 | 0 | 1 |  |
| 1 | 1 | 1 |  |
| 2 | 1 | 3 |  |
| 3 | 2 | 4 |  |
| 4 | 3 | 7 |  |
| 5 | 4 | 11 |  |
| 6 | 5 | 14 |  |
| 7 | 6 | 20 |  |
| 8 | 7 | 33 |  |
| 9 | 8 | 46 |  |
| 10 | 9 | 65 |  |
|  |  |  |  |