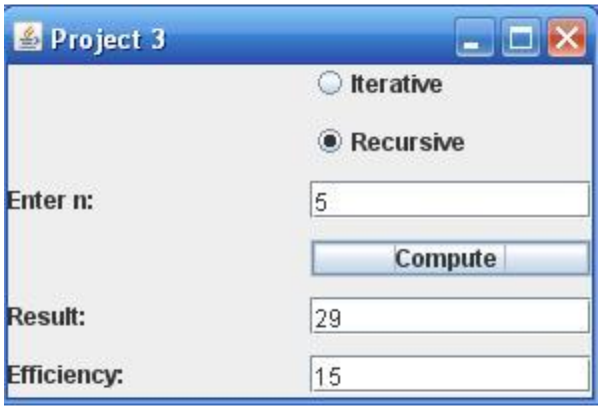
Project 3

This programming project involves writing a program to calculate the terms of the following sequence of numbers: 0 1 2 5 12 29 ... where each term of the sequence is twice the previous term plus the second previous term. The 0th term of the sequence is 0 and the 1st term of the sequence is 1.

For example:

0 1 2 -> (0 + 1 + 2) + 2 = 5
0 1 2 5 -> (0 + 1 + 2 + 5) + 5 = 12
0 1 2 5 12 -> (0 + 1 + 2 + 5 + 12) + 12 = 29
…
The interface to the program should be a GUI that looks similar to the following:



( ) Iterative
(●) Recursive
Enter n: 5
Result: 29
Efficiency: 15

The pair of radio buttons allows the user to choose whether an iterative or recursive method is used to compute the term of the sequence. When the user enters a value for *n* and then clicks the *Compute* button, the *n*th term of the sequence should be displayed in the *Result* field. The *Efficiency* field should contain the number of calls to the recursive method when the recursive option is chosen and the number of iterations of the loop when the iterative option is selected.

The *Iterative* radio button should be initially set to selected.

When the window is closed, the efficiency values should be computed with values of *n* from 0 to 10 and written to a file. Each line of the file should contain the value of *n*, the efficiency of the iterative method for that value of *n* and the efficiency of the recursive method. The values should be separated by commas so the file can be opened with Excel and used to graph the value of the efficiencies for both the iterative and recursive options along the *y* axis with the value of *n* along the *x*-axis. The graph should be included in the Word document that accompanies this project and should also contain a brief explanation of the observed results.

The program should consist of two classes.

1. The first class should define the GUI. In addition to the main method and a constructor to build the GUI, an event handler will be needed 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 should be an object of an inner class that extends the WindowAdapter class.

2. The other class should be named Sequence. It should be a utility class meaning that all its methods must be class (static) methods and no objects should be able to be generated for that class. It should contain three public methods:

   a. The first method computeIterative should accept a value of *n* and return the corresponding element in the sequence using iteration.
   b. The second method computeRecursive should accept a value of *n* and return the corresponding element in the sequence using recursion. This method will be a helper method because it will need to initialize the efficiency counter before calling the private recursive method that will actually perform the recursive computation.
   c. The third method getEfficiency will return the efficiency counter left behind by the previous call to either of the above two methods.

The google recommended Java style guide, provided as link in the week 2 content, should be used to format and document your code. Specifically, the following style guide attributes should be addressed:

- Header comments include filename, author, date and brief purpose of the program.
- In-line comments used to describe major functionality of the code.
- Meaningful variable names and prompts applied.
- Class names are written in UpperCamelCase.
- Variable names are written in lowerCamelCase.
- Constant names are in written in All Capitals.
- Braces use K&R style**.**

In addition the following design constraints should be followed:

- Declare all instance variables private
- Avoid the duplication of code

Test cases should be supplied in the form of table with columns indicating the input values, expected output, actual output and if the test case passed or failed. This table should contain 4 columns with appropriate labels and a row for each test case. Note that the actual output should be the actual results you receive when running your program and applying the input for the test record. Be sure to select enough different scenarios to completely test the program.

Note: All code should compile and run without issue.

Submission requirements
Deliverables include all Java files (.java) and a single word (or PDF) document. The Java files should be named appropriately for your applications. The word (or PDF) document should include screen captures showing the successful compiling and running of each of the test cases. Each screen capture should be properly labeled clearly indicated what the screen capture represents. The test cases table should be included in your word or PDF document and properly labeled as well.

Submit your files to the Project 3 assignment area no later than the due date listed in your LEO classroom. You should include your name and P3 in your word (or PDF) file submitted (e.g. firstnamelastnameP3.docx or firstnamelastnameP3.pdf).

Grading Rubric:

The following grading rubric will be used to determine your grade:

Attribute	Meets	Does not meet
GUI Class ✓ ✓ ✓ ✓ ✓ ✓ ✓	40 points Defines the GUI Contains a pair of radio buttons allowing the user to choose whether an iterative or recursive method is used to compute the term of the sequence. Allows the user to enter a value for n and click the Compute button, to display the nth term of the sequence in the Result field. Allows the Efficiency field to contain the number of calls to the recursive method when the recursive option is chosen and the number of iterations of the loop when the iterative option is selected. The Iterative radio button is initially set to selected. When the window is closed, the efficiency values computes with values of n from 0 to 10 and writes them to a file. Each line of the output file contains the value of n, the efficiency of the iterative method for that value of n and the efficiency of the recursive method.	0 points Does not defines the GUI Does not contain a pair of radio buttons allowing the user to choose whether an iterative or recursive method is used to compute the term of the sequence. Does not allows the user to enter a value for n and click the Compute button, to display the nth term of the sequence in the Result field. Does not allow the Efficiency field to contain the number of calls to the recursive method when the recursive option is chosen and the number of iterations of the loop when the iterative option is selected. The Iterative radio button is not initially set to selected. When the window is closed, the efficiency values does not compute with values of n from 0 to 10 and writes them to a file. Each line of the output file does not contain the value of n, the efficiency of the iterative method for that value of n and the efficiency of the recursive method.

✓ ✓	The values of the output file are separated by commas so the file can be opened with Excel. 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.	The values of the output file are not separated by commas so the file can be opened with Excel. Does not 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. Code does not Compile.
Sequence class ✓ ✓ ✓ ✓ ✓ ✓	30 points All methods are class (static) methods. Contains three public methods. Contains computeIterative method that accepts a value of n and returns the corresponding element in the sequence using iteration. Contains method computeRecursive that accepts a value of n and returns the corresponding element in the sequence using recursion. The computeRecurvise method will initialize the efficiency counter before calling the private recursive method that will actually perform the recursive computation. The getEfficiency method returns the efficiency counter left behind by the previous call	0 points All methods are not class (static) methods. Does not contain three public methods. Does not contain the computeIterative method that accepts a value of n and returns the corresponding element in the sequence using iteration. Does not contain the computeRecursive method that accepts a value of n and returns the corresponding element in the sequence using recursion. The computeRecurvise method does not initialize the efficiency counter before calling the private recursive method that will actually perform the recursive computation. The getEfficiency method does not return the efficiency counter left behind by the

	to either of the above two methods.	previous call to either of the above two methods. Code does not Compile.
Test Cases	10 points ✓ Test cases are supplied in the form of table with columns indicating the input values, expected output, actual output and if the test case passed or failed. ✓ Enough scenarios selected to completely test the program. ✓ Test cases were included in the supporting word or PDF documentation.	0 points No test cases were provided.
Documentation and Style guide	20 points ✓ Screen captures were provided and labeled for compiling your code, and running each of your test cases. ✓ Header comments include filename, author, date and brief purpose of the program. ✓ In-line comments used to describe major functionality of the code. ✓ Meaningful variable names and prompts applied. ✓ Class names are written in UpperCamelCase. ✓ Variable names are written in lowerCamelCase. ✓ Constant names are in written in All Capitals.	0 points No documentation included. Java style guide was not used to prepare the Java code. All instance variables not declared private. Duplication of code was not avoided. Does not graph the value of the efficiencies for both the iterative and recursive options along the y axis with the value of n along the x-axis. The graph is not included in the Word document and does not contain a brief explanation of the observed results.

✓ ✓ ✓ ✓	Braces use K&R style**.** Declare all instance variables private. Avoids the duplication of code. Graphs the value of the efficiencies for both the iterative and recursive options along the y axis with the value of n along the x-axis. The graph is included in the Word document and contains a brief explanation of the observed results.	