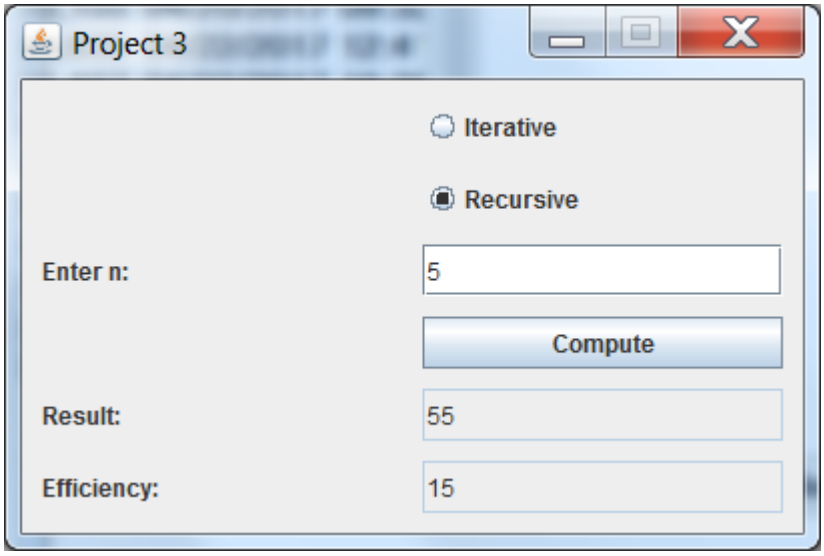
CMIS 242 - Project 3

1. Specification

The third programming project involves writing a program to calculate the terms of the following sequence of numbers: 0 1 3 8 21 55 144 377 … where each term is three times the previous term minus the second previous term. The 0^{th} term of the sequence is 0 and the 1^{st} term of the sequence is 1.
The example below shows how to calculate the next sequence term:

Current sequence: 0 1 Calculate next term: 3 * 1 - 0 = 3
Current sequence: 0 1 3 Calculate next term: 3 * 3 - 1 = 8
Current sequence: 0 1 3 8 Calculate next term: 3 * 8 - 3 = 21
Current sequence: 0 1 3 8 21 Calculate next term: 3 * 21 - 8 = 55
… etc.

The interface to the program should be a Swing based GUI that looks similar to the following:



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 (the **Iterative** radio button should be initially selected). When the user enters a value for *n* and then clicks the *Compute* button, the n^{th} term of the sequence (counting from zero) 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 program will check the validity of the user input value which should be a positive integer. A message will be shown in a **JOptionPane** for illegal entered values.

When the window is closed, the efficiency values should be computed with values of *n* from 0 to 15 and written to a text file **outData.txt**. 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 and with the value of *n* along the *x*-axis. The graph should be included in the solution description document that accompanies this project and should also contain a brief explanation of the observed results.

The program should consist of two classes: **P3GUI** and **Sequence**.

1. Class **P3GUI** should define the Swing based GUI and should be hand-coded and not generated by a GUI generator. 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. Class **Sequence** 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 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.

Your program should compile without errors.

The Google recommended Java style guide (https://google.github.io/styleguide/javaguide.html) 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 a table with columns indicating what aspect is tested, the input values, expected output, actual output and if the test case passed or failed. This table should contain 5 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 kinds of employees and situations to completely test the program.

2. Submission Requirements

Submit the following to the Project 3 assignment area no later than the due date listed in your LEO classroom.

1. Source files **P3GUI.java**, **Sequence.java** and the program generated output file **outData.txt**. The source code should use Java code conventions and appropriate code layout (white space management and indents) and comments. All submitted files may be included in a .zip file.

2. The solution description document **P3SolutionDescription** (.pdf or .doc / .docx) containing the following:
(1) Assumptions, main design decisions, error handling;
(2) Test cases table
(3) The graph of the value of the efficiencies for both the iterative and recursive options along with a brief explanation of the observed results.
(4) Screen captures showing successful program compilation and test cases execution. Each screen capture should be properly labeled, clearly indicated what the screen capture represents.
(5) Lessons learned from the project;

3. Grading Rubric

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

Attribute	**Meets**	**Does not meet**
P3GUI class	**40 points** a) Defines the GUI. b) 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. c) 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. d) User input value is checked and warning message is displayed if the entered value is not a positive integer. e) 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. f) The Iterative radio button is initially set to selected. g) When the window is closed, the efficiency values computes with values of n from 0 to 15 and writes them to a file. h) Each line of the output file contains the value of n, the efficiency of the iterative method for that value of n and	**0 points** a) Does not define the GUI. b) 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. c) 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. d) User input value is checked and warning message is displayed if the entered value is not a positive integer. e) 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. f) The Iterative radio button is not initially set to selected. g) When the window is closed, the efficiency values does not compute with values of n from 0 to 15 and the output file is not generated. h) Each line of the output file does not contain the value of n, the efficiency of

	the efficiency of the recursive method. i) The values of the output file are separated by commas so the file can be opened with Excel. j) 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 iterative method for that value of n and the efficiency of the recursive method. i) The values of the output file are not separated by commas so that the file can be opened by Excel. j) 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.
Sequence class	**30 points** a) All methods are class (static) methods. b) Contains three public methods. c) Contains computeIterative method that accepts a value of n and returns the corresponding element in the sequence using iteration. d) Contains method computeRecursive that accepts a value of n and returns the corresponding element in the sequence using recursion. e) The computeRecurvise method will initialize the efficiency counter before calling the private recursive method that will actually perform the recursive computation. f) The getEfficiency method returns the efficiency counter left behind by the previous call to either of the above two methods.	**0 points** a) All methods are not class (static) methods. b) Does not contain three public methods. c) Does not contain the computeIterative method that accepts a value of n and returns the corresponding element in the sequence using iteration. d) Does not contain the computeRecursive method that accepts a value of n and returns the corresponding element in the sequence using recursion. e) The computeRecurvise method does not initialize the efficiency counter before calling the private recursive method that will actually perform the recursive computation. f) The getEfficiency method does not return the efficiency counter left behind by the previous call to either of the above two methods.
Test Cases	**10 points** a) Test cases are supplied in the form of table with columns indicating test case objective, the input values, expected output, actual output and if the test case passed or failed.	**0 points** a) No test cases were provided.

	b) Enough scenarios selected to completely test the program. c) Test cases were included in the supporting word or PDF documentation.	
Documentation and Style guide	**20 points** a) Solution description document **P3SolutionDescription** includes all the required sections appropriate titled. b) Solution description document **P3SolutionDescription** includes project specific, meaningful information. **Source code** criteria c) Header comments include filename, author, date and brief purpose of the program. d) In-line comments used to describe major functionality of the code. e) Meaningful variable names and prompts applied. f) Class names are written in UpperCamelCase. g) Variable names are written in lowerCamelCase. h) Constant names are in written in All Capitals. i) Braces use K&R style. i) Declare all instance variables private. k) Avoids the duplication of code.	**0 points** a) No **solution description document** is included. **Source code** criteria b) Java style guide was not used to prepare the Java code. c) All instance variables not declared private. d) Duplication of code was not avoided.