**Aconex Coding Challenge**

1. Problem Selection

I selected 1800-Coding-Challenge because it involved lots of critical thinking of how to use different data algorithms, performance consideration, and design patterns. Those are the fundamental knowledge on how to help the team to develop and implement a great product, which I enjoyed the most in my career.

1. Design

The design involves the design pattern of Singleton and Strategy. The following is the explanation of the usage on the classes.

* Dictionary

It’s the object class, which stores the dictionary data in Set, which get used of the nature of Set class to avoid duplicate data.

* PhoneNumber

It’s the object class, which stores the phone number data in Set, which get used of the nature of Set class to avoid duplicate data.

* Command

It’s the POJO, which stores the instruction data to execute the program.

* CmdLineProcessor

It processes the logics of how to convert the phone number and match the converted phone number with dictionary, which contains methods to search and convert the phone number recursively. And then it stores the data with Map in a key-value pair, which the key is the original phone number and the value contains a list of converted phone number.

* FileProcessor

It inherits the functionalities of CmdLineProcessor for processing file based matching.

1. Approach

I read through the requirements and started to write down the pseudocode. Once I got through the logics correctly, I started developing my testing program by writing a simple method to read the file and store the data in the object with different scenarios and exception handling, which I used Java 8 Lambda expression. And then I enhanced the method by filtering all punctuation and non-alphabetic characters with regular expression.

My next step was to develop an efficient way to convert every digit with all possible alphabets in the array by using recursion. I split the phone number by a block of digit from left to right. For example, 2255.63 would be [2255, 63] in the array. And then I created a placeholder array based on the length of a block of digit and inserted all possible alphabet in the placeholder where a number can be replaced. Each of the possible word would check with the dictionary before it would add to the result list. And then I performed the combination with all possible word together.

The final step was to simply put different method together by implementing the logics on how to read the options from the user input, which I decided to use Strategy Pattern, where I performed different operation based on the input arguments.