## **Why this challenge**

First, at reading all three provided challenges (1-800 , Vehicle Survey and GEDCOM parser), the 1-800 challenge seemed a little more clearly defined to me. Second, while reading the 1-800 challenge, a first design approach came to my mind.

## **Design and approach**

**Approach**

I choose a basic Input-Process-Output approach. After reading a phone number all the possible words are generated from the given number-alphabet mapping (also used in keypad phones).

Now each word from generated words are checked whether they can be broken into words that are present in dictionary. This approach is time feasible for any number ranging from 1-15 digits. Finally it will show possible matches for given phone number.

### **Design**

PhoneNumberEncoding is the main class of this coding challenge. Firstly it is analysed whether the phone number would be provided from text file or from STDIN. Also default dictionary is used if no dictionary is provided.

DictionaryReader and PhoneNumberProcessor helps to ignore all the punctuations, whitespaces, etc. Finally PhoneNumberProcessor call the Result class to get possible matches of valid phone number.

Result class calls PossibleWords to generate all possible words for given number with help of recursion. Now all these words are passed to WordBreak class to get the final result. WordBreak consider only those words which are made up of one or more word present in dictionary. Also no two consecutive digits can remain unchanged.