Name: Ronald Macmaster

UT EID: rpm953

PA1: Fun with Pig Latin!

1) Analysis

Problem Statement:

Given a phrase in the English language, translate it into its Pig Latin equivalent.

Write a Java program that will convert a sequence of English language phrases into Pig Latin.

Input:

A file of English phrases that will be specified as a Command Line Argument.

Each phrase will be translated/input one string at a time into Pig Latin

Output:

Print both the original English phrase and its Pig Latin counterpart to the screen.

Properly label each phrase. Convert every phrase in the file.

Examples:

scratch -> atchscray

only -> onlyyay

high-powered -> ighay-oweredpay

can’t -> an’tcay

h1N1 -> h1N1

?.,sdf?d -> ?.,sdf?d

893 -> 893

Sample output:

The phrase that was input is:

    Fun with pig latin!

The pig latin equivalent phrase is:

    unFay ithway igpay atinlay!

Questions?

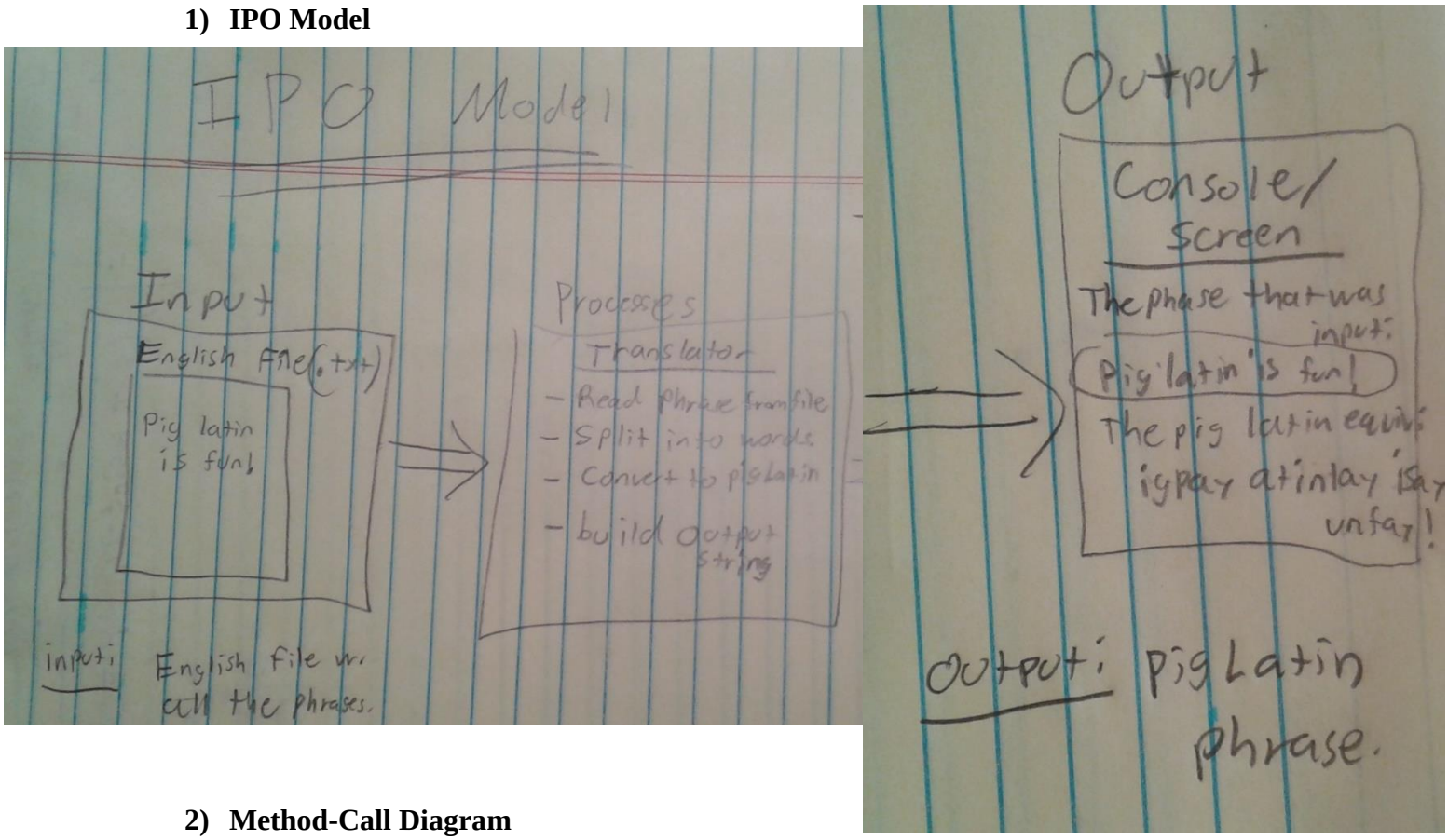
Should I attempt to move punctuation to the end?

Maybe it is better to just leave it alone.

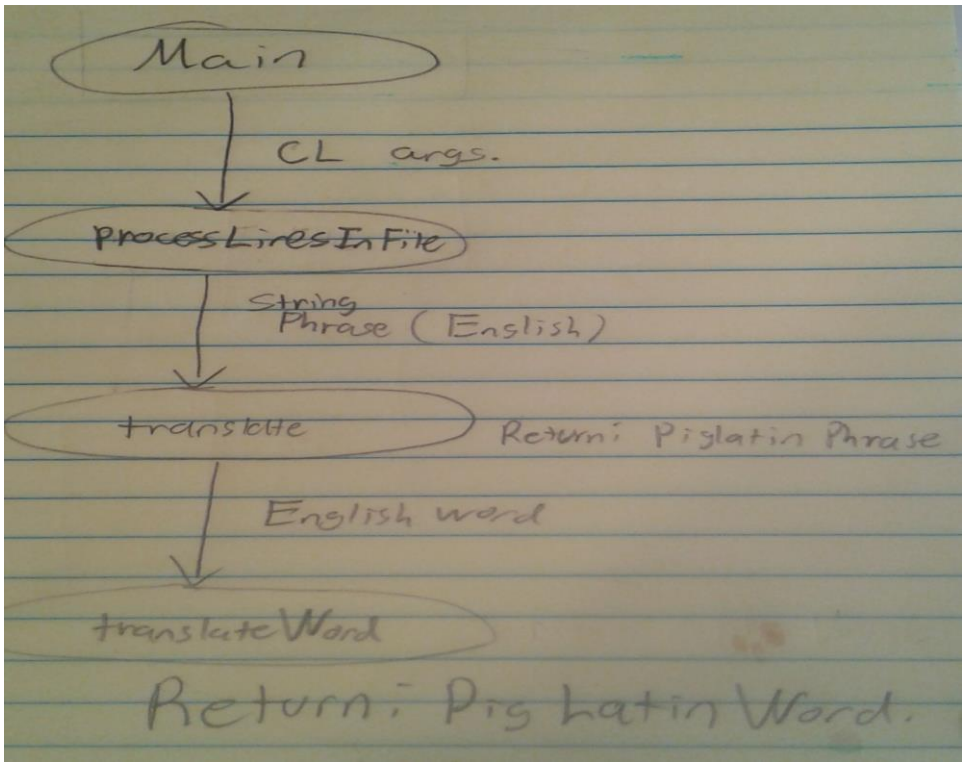
2) Design

Architecture Models:

1) **IPO Model**



2) **Method-Call Diagram**



Algorithm:

translate():

Split the input string into individual words.

Initialize an empty output string.

For each word in the input string, concatenate the Pig Latin form to the output string.

Return the output string.

translateWord():

Is the word valid? (no numbers or non letter characters. punctuation and hyphens are ok)

    No: return word unchanged.

Does the word contain a hyphen or punctuation?

    Split the string into two words

    Hyphen: Translate each word, then concatenate them with the hyphen in between

    Punctuation:

        Translate the sum of the two words, then

        Concatenate them with the punctuation at the end.

Does the word start with a vowel?

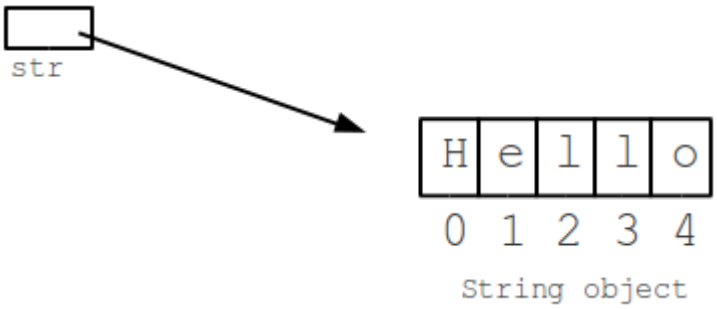
    Yes: append the letter ‘y’ to the end of the word

Loop: Does the word contain and begin with a vowel?

    No: rotate first char to the back and repeat.\

Append the phrase “ay” and return the word.

Data Structures:

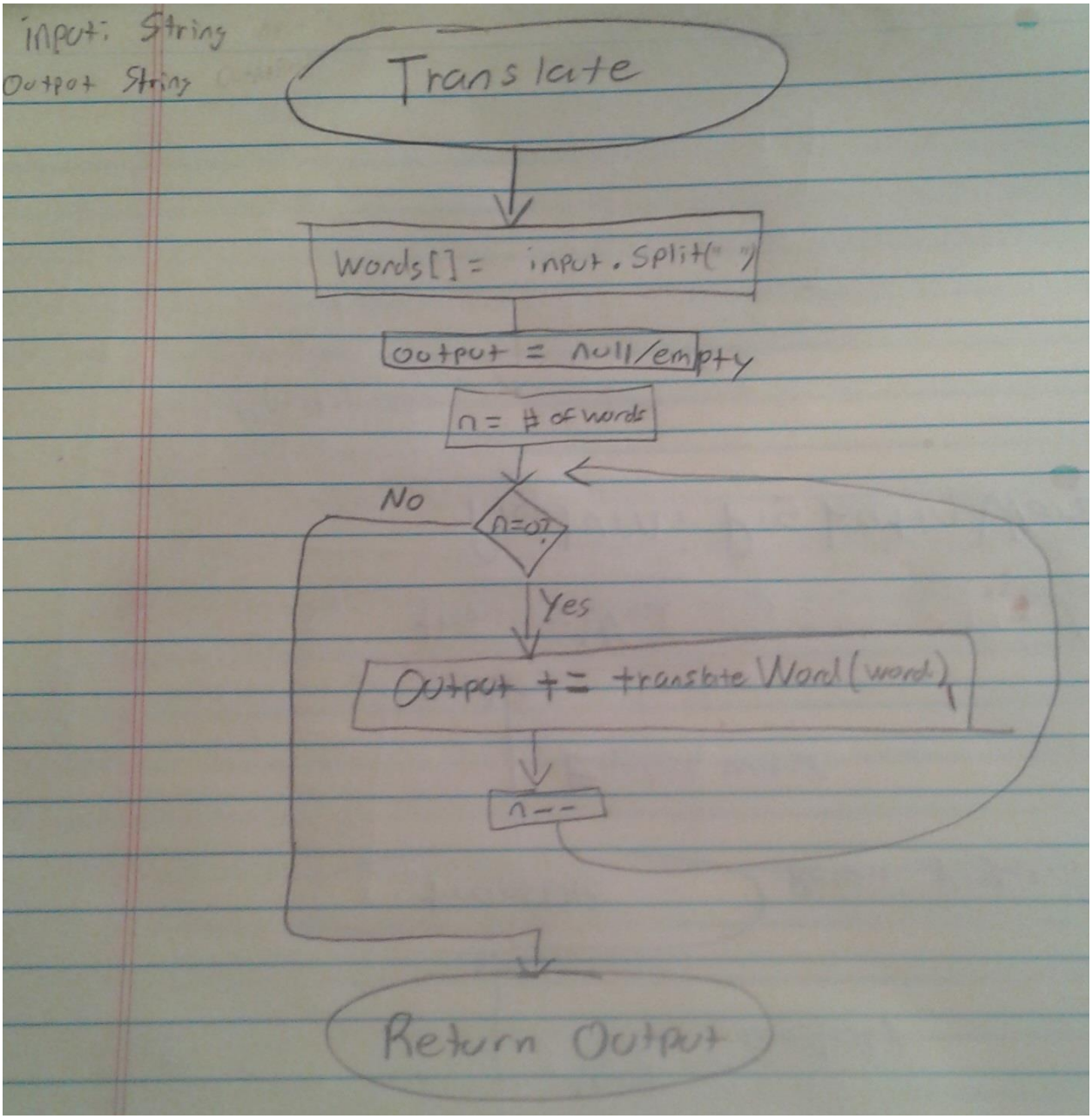


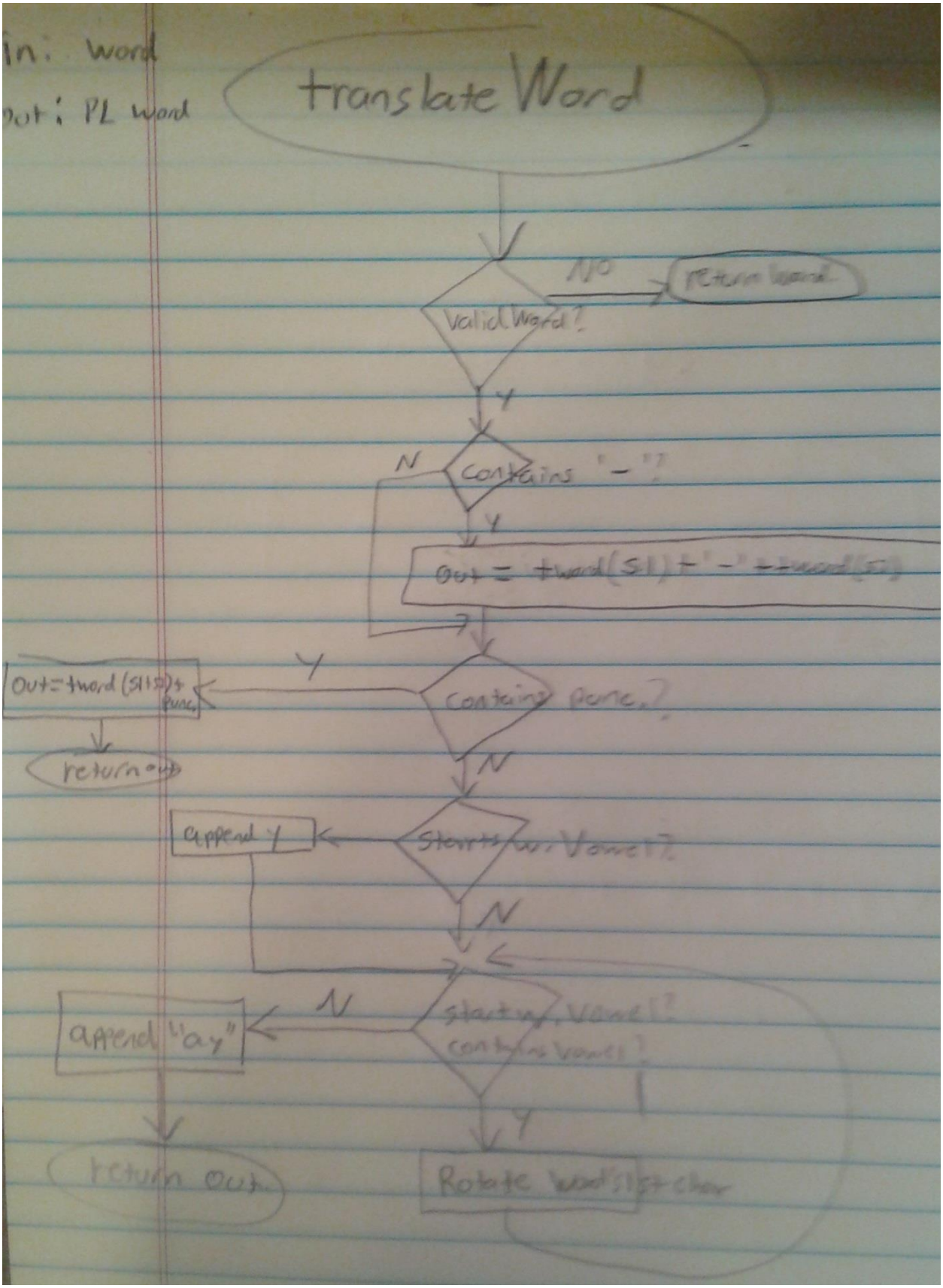
The translator will use the Java String class and its associated String methods to store and manipulate the phrases.

The phrases and various parts will be represented as Strings.

The phrases will be read in through a file which will be interfaced through a file handle.

Flowchart:


input: String
Output String
Translate
Words[] = input.Split(" ")
output = null/empty
n = # of words
No
n=0?
Yes
Output += translateWord(word)
n--
Return Output


in: word
out: PL word
translateWord
NO
return word
valid word?
Y
N
contains "-"?
Y
out = fword(s:i) + "-" + fword(s:
Y
contains pure?
out = fword(s:1:s) + punc
return out
N
append "y"
starts w/ vowel?
N
append "ay"
starts w/ vowel?
contains vowel?
Y
return out
Rotate word's 1st char