**CSCI 1152, Summer 2020  
Project**

# Objectives

To show that we learned several programming concepts, including:

* 1. How to process String’s.
  2. How to create and use methods.
  3. How to process array’s and List’s using ArraryList.
  4. How to use loop and if statements.
  5. How to read data from a file.
  6. How to read data from the command line.

# Description

Write a Java program to read data from a text file (file name given on command line), process the text file by performing the following:

1. Print the total number of words in the file.
2. Print the total number of unique words (case sensitive) in the file.
3. Print all words in **ascending** order without duplication, i.e. all unique words.
4. Print all line(s) with line number(s) of the file where a specific phrase is found. You must prompt the user to enter a specific phrase to search for. Under each output line indicate the location(s) of the first character of the matched phrase (refer to the Sample Ouput below).
5. After outputting results keep prompting user until the user enters EINPUT.

The file that is to be read is given to you in learn.cnm.edu. The file name is random.txt. This is the file you should use for the project.

# Requirements

1. **You are allowed to use built in Java List’s. For example, ArrayList to implement your solution.**
2. The java source file MUST abide by the google java style guide. It can be found at <https://google.github.io/styleguide/javaguide.html#s7.1-javadoc-formatting>

Insructions for automatically formatting your code using the google style guide can be found here: <https://learn.cnm.edu/bbcswebdav/pid-8979615-dt-content-rid-95578999_1/xid-95578999_1>

1. The class name is required to be Project\_<your email id>
   1. For example, my class would be:

public class Project\_mgonzales183 {

}

1. You must enter a javadoc comment before the class declaration. In the comment you must provide a small description of this program followed by your name.
   1. For example, my class would be:

/\*\*

\* Project is a program to…….put your description here…

\*

\* @author Mark Gonzales

\*/

public class Project\_mgonzales183 {

}

1. You must enter a javadoc comment before EACH method declaration. In the comment you must provide a small description of this method, specify the parameters and what the method returns. Follow the template below:

/\*\*

\* <write your description of method here>

\* @param <parameter name here> <write description of parameter is here>

\* @return <write description of what is returned here>

\*/

1. You must handle all Checked Exceptions appropriately:
   1. Must output an appropriate message.
   2. Must put the program into a state in which the user can recover from or exit if necessary.
2. There must be at a minimum the following methods:
   1. (10pts) Name: main
      1. Description: The main method should do the following.
         1. Retrieve the file name from the command line.
         2. The program should exit if the file does not exist or if the file name is not given on the command line. When exiting the program should print the message “File does not exist.”
         3. If the file exists the main method should call the processFile method. Any data that is needed from the main method must be passed to the processFile method via its parameters. See processFile method below.
   2. (30pts) Name: processFile
      1. Description: The processFile method should do the following:
         * 1. Read contents from the file.
           2. Print the total number of words in the file.
           3. Print the total number of unique words (case sensitive) in the file. In order to do part “c” you must call the createUniqueList method.
           4. Print all words in ascending order without duplication. In order to do part “d” you must call the sortList method.

Once the information above is printed to the console the processFile method should call the search method. Any data that is needed from the processFile method must be passed to the search method via its parameters.

* 1. (10 pts) Name: createUniqueList
     1. Description: The createUniqueList method should receive as input a List of words that are not unique, i.e. there are duplicate words in the List. The method must create a new List by adding only unique words from the input List.
        1. Parameter
           1. Name: wordList
           2. Type: ArrayList<String>
           3. Description: a list that has duplicate words
        2. Return
           1. Type: ArrayList<String>
           2. Description: a list of non-duplicate words
  2. (15 pts) Name: sortList
     1. Description: The sortList method should receive as input a List of unsorted words. The sortList method should modify the unsorted List into a sorted List. You must implement a sort algorithm. You can choose any sorting algorithm you want. Do not use a built in Java method to sort the list.
        1. Parameter
           1. Name: uniqueWordList
           2. Type: ArrayList<String>
           3. Description: a list of un-sorted words
        2. Return
           1. ArrayList<String>
           2. Description: a list of sorted words
  3. (35pts) Name: search
     1. Description: The search method should prompt the user for a search phrase. Given the search phrase the search method should search each line in the file for the location of the phrase on each line. If the search phrase is not found on a line then just go to the next line. The search method must find every occurrence of the phrase on a line. The search method should print the line number, the line and the location of the search phrase (“use ^”). After all lines have been searched the search method then prompts the user to enter another search phrase. The search method does not exit until the user enters the phrase EINPUT. See Sample Output.

1. The output should match the format of the Sample Output.

# Sample Output

Total number of words in file: 114

Total number of unique words in file: 105

Unique words of the input file in ascending order:

Abode

Added

Advantages

Alteration

amiable

and

any

Are

at

better

between

boisterous

boy

by

collected

collecting

companions

considered

contented

day

deal

delight

design

diminution

discovered

do

early

easy

end

entire

ever

excellence

fancy

first

former

forming

forth

fortune

garrets

get

greater

happiness

he

Her

horses

hours

how

if

is

its

jokes

joy

Justice

Leaf

led

left

many

mile

morning

new

no

nor

norland

not

Nothing

now

of

off

oh

old

our

Out

own

partiality

parties

placing

Pursuit

put

quiet

reasonable

shade

she

should

shy

Sigh

Sincerity

So

so

Square

staying

to

towards

two

unsatiable

Up

use

waiting

warrant

way

whole

winding

wishing

woman

ye

you

Enter Search Pattern: he

Line number 1

Her old collecting she considered discovered.

^

Line number 2

So at parties he warrant oh staying. Square new horses and put better end.

^

Line number 8

Pursuit he he garrets greater towards amiable so placing.

^ ^

Line number 9

Nothing off how norland delight. Abode shy shade she hours forth its use.

^

Enter Search Pattern: e

Line number 1

Her old collecting she considered discovered.

^ ^ ^ ^ ^ ^ ^

Line number 2

So at parties he warrant oh staying. Square new horses and put better end.

^ ^ ^ ^ ^ ^ ^ ^

Line number 3

Sincerity collected happiness do is contented.

^ ^ ^ ^ ^ ^

Line number 4

Sigh ever way now many. Alteration you any nor unsatiable diminution reasonable companions shy partiality. Leaf by left deal mile oh if easy.

^ ^ ^ ^ ^ ^ ^ ^ ^ ^ ^

Line number 5

Added woman first get led joy not early jokes.

^ ^ ^ ^ ^

Line number 6

Are own design entire former get should.

^ ^ ^ ^ ^ ^

Line number 7

Advantages boisterous day excellence boy. Out between our two waiting wishing.

^ ^ ^ ^ ^ ^ ^ ^^

Line number 8

Pursuit he he garrets greater towards amiable so placing.

^ ^ ^ ^ ^ ^

Line number 9

Nothing off how norland delight. Abode shy shade she hours forth its use.

^ ^ ^ ^ ^

Line number 10

Up whole of fancy ye quiet do. Justice fortune no to is if winding morning forming.

^ ^ ^ ^ ^

Enter Search Pattern: EINPUT

Bye!

# Testing

I will test your program as follows:

javac Project\_<your email id>.java

java Project\_<your email id> random.txt

I will use the random.txt file given as part of the project in learn.cnm.edu to test your code.

# Psuedo Code

## Methods in classes you will probably use

### ArrayList Methods

add

size

get

set

contains

### String Methods

toLowerCase

compareTo

## main Method

Read file name from command line

Validate that the file exists. If it does not exist then output a message and exit.

Call processFile method passing the file name as a parameter

## processFileMethod

Open file for reading

Create needed data lists to store lines from the file and words from the file.

While (file has data)

Read line from file and put it into list that holds the lines

Parse the line into words and put all words into the list that holds words

Close the file

Call createUniqueList method passing the word list as a parameter

Print out the total number of words in the file using the list that holds the words

Print out the total number of unique words that were returned from createUniqueList

Call sortList method passing the unique word list as a parameter.

Print out the unique words in ascending order

Call the search method passing the list that holds the lines as a parameter

## createUniqueList Method

Create a new list that will hold all the unique words

Iterate through the list that contains all the words and only add unique words to the new list

Return the new list

## sortList Method

implement insertion sort to sort the list that is passed as a parameter.

NOTE: no need to return a list because the list passed as a parameter is a reference object.

## search Method

prompt user to enter search term

while (the user has not entered EINPUT)

for every line in line list

if search term is in line

print line

At every index where search term is found print “^”

prompt user for search term

print “Bye!”