ITCS 209 Object Oriented Programming	Name:	Lab Score	Challenge Bonus	Peer Bonus
	ID:			
	Sec:			

Lab13: Searching on Name Lists

Searching employee information in organization is an important feature of the human resource management system. To search for the information, we need a key or a query word for finding and pull out the information to display.

In this lab, we are about to implement a simple mechanism to search for id of employee (id is a position of name located in the file.) by their full name, using **sequential** search. The system is ***case-insensitive***, in the sense that every character will be automatically lowercased before processing. A document name lists is defined as a sequence of String one line per name.

You have to use method compareTo() from String to compare the given string name with current string name string1.compareTo(string2), If string1 is lexicographically greater than string2, it returns positive number (difference of character value). If string1 is less than string2 lexicographically, it returns negative number and if string1 is lexicographically equal to string2, it returns 0.

We also provide an abstract class NameSearcher in NameSearcher.java that contains the following:

[Provided Methods]

- **protected static ArrayList<String> readNames** : A list of String name of employee read from the input document.
  protected int number_of_compared: A number of total comparison for each search query.
- **NameSearcher(String filename)**: A constructor that loads the text file (specified by filename), cleans the text, tokenizes, and stores the sorted words in readNames. This method is already implemented for you.
- **public int getNumComparisons():** A method that return a number of total comparison for each search query.
- **public void resetCompareCounter ()**: A method that is used to reset the number of total comparison before searching for a new information.

[You have to implement the following method in NameSearcher:]

- **public void sortWord()**: A method for sorting name of employee in a list of String readNames. This method is used for the bonus lab. You can implement this function using any sorting algorithm.

Your task is actually very simple. Besides understanding the provided code (code reading is a necessary skill for a good programmer), you must create class **LinearNameSearcher** (in NameSearcher.java) that extends **NameSearcher**, then implement the necessary constructor, and **find(String query)** that uses ***linear search*** algorithm discussed in class, by linearly scanning through ***readNames*** *and* return String output as presented in the output box. You must use **compareTo()** whenever you need to check for equality or compare two String name. You may use the provided **commons-io-2.6.jar** to facilitate file I/O.

The output from NameSearcherTester should look like:

```
***************** NORMAL****************

[Linear-Case1] Found: 'Zebra' AT_INDEX(0) >>> Number of Comparisons (Linear):1

[Linear-Case2] Found: 'ant' AT_INDEX(3) >>> Number of Comparisons (Linear):4

[Linear-Case3] Not Found Name: 'tiger' >>> Number of Comparisons (Linear):6

[Linear-Case4] Found: 'Monkey D. Luffy' AT_INDEX(500) >>> Number of Comparisons (Linear):501

[Linear-Case5] Not Found Name: 'Monkey' >>> Number of Comparisons (Linear):1177

[Linear-Case6] Found: 'trafalgar d. water law' AT_INDEX(1079) >>> Number of Comparisons (Linear):1080

[Linear-Case7] Not Found Name: 'Yonta Maria Grand Fleet' >>> Number of Comparisons (Linear):1177
```

Challenge Bonus (Optional):

Implement **BinaryNameSearcher** that extends **NameSearcher,** then implement the necessary constructor, and **find(String query)** that uses ***binary search*** algorithm on ***readNames*** and return String output as presented in the output box.. You must use **compareTo()** whenever you want to check for equality or compare two String name.

The output from NameSearcherTester (Bonus) should look like:

```
********* BONUS for CRAZY PEOPLE************

[Binary-Case1] Found: 'Zebra' AT_INDEX(5) >>> Number of Comparisons (Binary):3

[Binary-Case2] Found: 'ant' AT_INDEX(1) >>> Number of Comparisons (Binary):3

[Binary-Case3] Not Found Name: 'tiger' >>> Number of Comparisons (Binary):3

[Binary-Case4] Found: 'Monkey D. Luffy' AT_INDEX(736) >>> Number of Comparisons (Binary):10

[Binary-Case5] Not Found Name: 'Monkey' >>> Number of Comparisons (Binary):11

[Binary-Case6] Found: 'trafalgar d. water law' AT_INDEX(1086) >>> Number of Comparisons (Binary):9

[Binary-Case7] Not Found Name: 'Yonta Maria Grand Fleet' >>> Number of Comparisons (Binary):10
```

hing
ss