Chris Piech
CS 106A

Section #5
February 9, 2019

Section Handout #5: Files, ArrayLists, and Arrays

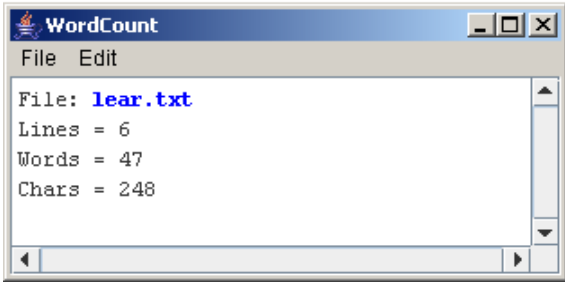
Portions of this handout by Eric Roberts and Marty Stepp

1. Word Count

Write a program **`WordCount`** that reads a file and reports how many lines, words, and characters appear in it. Suppose, for example, that the file **`lear.txt`** contains the following passage from Shakespeare's *King Lear*:

```
Poor naked wretches, wheresoe'er you are,
That bide the pelting of this pitiless storm,
How shall your houseless heads and unfed sides,
Your loop'd and window'd raggedness, defend you
From seasons such as these?  O, I have ta'en
Too little care of this!
```

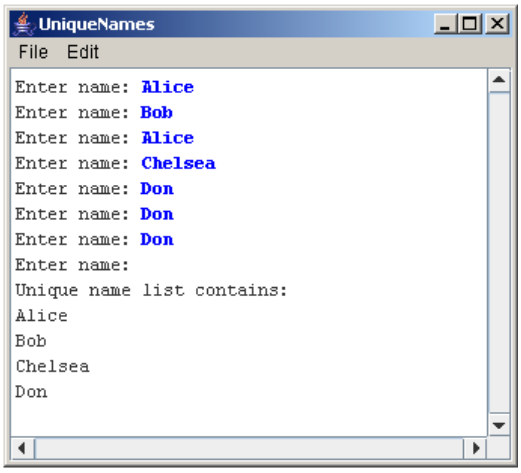
Given this file, your program should be able to generate the following sample run:



For the purposes of this program, a word consists of a consecutive sequence of letters and/or digits, which you can test using the static method **`Character.isLetterOrDigit`**. You should reprompt the user until they enter a valid filename.

2. How Unique!

Write a program that asks the user for a list of names (one per line) until the user enters a blank line (i.e., just hits return when asked for a name). At that point the program should print out the list of names entered, where each name is listed only once (i.e., uniquely) no matter how many times the user entered the name in the program. You may find that using an **`ArrayList`** to keep track of the names entered by the user may greatly simplify this problem. A sample run of this program is shown at right.



3. Mirror

Write a method named **`mirror`** that accepts an **`ArrayList`** of strings as a parameter and modifies it to also contain a mirrored copy of the list. It should modify the parameter itself, and *not* return any value. For example, it should be able to be used as follows:

```
ArrayList myList = ... // ["how", "are", "you"]
mirror(myList);
println(myList);       // ["how", "are", "you", "you", "are", "how"]
```

4. Index Of

Write a method named **`indexOf`** that returns the index of a particular value in an array of integers. The method should return the index of the first occurrence of the target value in the array. If the value is not in the array, it should return **`-1`**. For example, if an array stores the following values:

```
int[] a = {42, 7, -9, 14, 8, 39, 42, 8, 19, 0};
```

Then the call **`indexOf(a, 8)`** should return **`4`** because the index of the first occurrence of value **`8`** in the array is at index **`4`**. The call **`indexOf(a, 2)`** should return **`-1`** because value **`2`** is not in the array.

5. Unique Numbers

Write a method named **`numUnique`** that accepts an array of integers as a parameter and returns the number of unique values in the array. The array is guaranteed to be in sorted order, which means that duplicates will be grouped together. For example, if an array stores the following values:

```
int[] list = {5, 7, 7, 7, 22, 22, 23, 35, 35, 40, 40, 40}
```

then the call **`numUnique(list)`** should return 6 because this list has 6 unique values **`(5, 7, 22, 23, 35, 40)`**. It is possible that the list might not have any duplicates. If **`list`** instead stored:

```
int[] list = {1, 2, 11, 17, 24, 25, 26, 31, 34, 37, 40, 41}
```

then a call on the method would return 12 because this list contains 12 different values. If passed an empty list, your method should return 0.

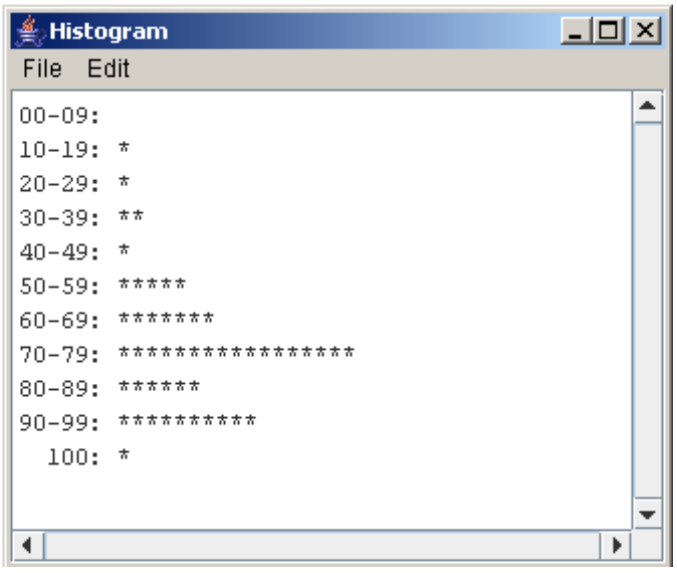
6. Collapse

Write a method named **`collapse`** that accepts an array of integers as a parameter and returns a new array containing the result of replacing each pair of integers with the sum of that pair. For example, if an array called list stores the values **`{7, 2, 8, 9, 4, 13, 7, 1, 9, 10}`**, then the call of **`collapse(list)`** should return a new array containing **`{9, 17, 17, 8, 19}`**. The first pair from the original list is collapsed into 9 (7 + 2), the second pair is collapsed into 17 (8 + 9), and so on. If the list stores an odd number of

elements, the final element is not collapsed. For example, if the list had been `{1, 2, 3, 4, 5}`, then the call would return `{3, 7, 5}`. Your method should not change the array that is passed as a parameter.

7. Histograms

Write a program that reads a list of exam scores from the file MidtermScores.txt (which contains one score per line) and then displays a histogram of those numbers, divided into the ranges 0-9, 10-19, 20-29, and so forth, up to the range containing only the value 100. If, for example, MidtermScores.txt contains the data shown in the right margin, your program should then be able to generate a histogram that looks as much as possible like the following sample run:



MidtermScores.txt

```
73
58
73
93
82
62
80
53
93
52
92
75
65
95
23
100
75
38
80
77
92
60
98
95
62
87
97
73
78
72
55
58
42
31
78
70
78
74
70
60
72
```