

Assessed Coursework

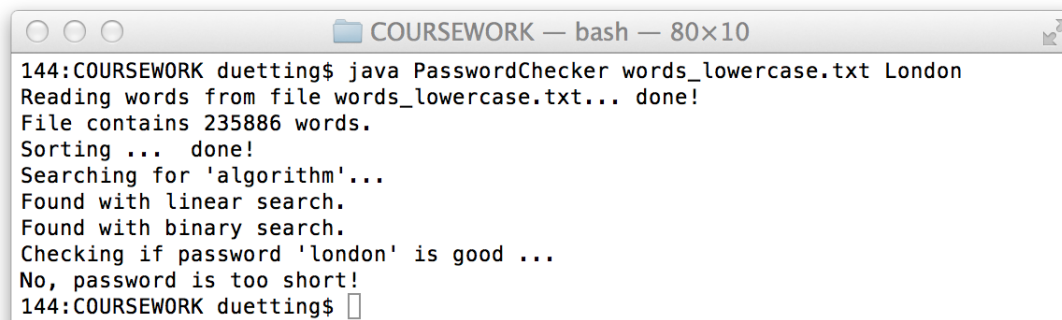
Is one of your commonly used passwords **London**? It shouldn't! Let's see why. Or better, let's write a program that checks if a given password is safe.

What you need: You should download the template **PasswordChecker.java** and the file **words_lowercase.txt** from Moodle. The former contains a skeleton of the program that you are asked to write. The latter contains 235.886 taboo words.

What you should do: Go through the steps below and complete the PasswordChecker program. When you are done hand in the code that you have produced through Moodle. The deadline for handing is **February 26, 4pm UK time**. No late submissions will be accepted.

Overview

Our overall goal is to complete the code in **PasswordChecker.java** so that it takes two arguments from the command line. The first is a path to a file containing a list of taboo words, each on a newline. This will typically be the file **words_lowercase.txt**. The second argument is a password such as **London**. So assuming that both **PasswordChecker.java** and **words_lowercase.txt** reside in the same folder and that you are currently in that folder we would like our program to produce the following output:



```
COURSEWORK — bash — 80x10
144:COURSEWORK duetting$ java PasswordChecker words_lowercase.txt London
Reading words from file words_lowercase.txt... done!
File contains 235886 words.
Sorting ... done!
Searching for 'algorithm'...
Found with linear search.
Found with binary search.
Checking if password 'london' is good ...
No, password is too short!
144:COURSEWORK duetting$
```

As you see the password that we entered **London** was turned into lowercase by the program, and in this case our program told us that the password that we have chosen is not safe as it is too short.

Some background on reading inputs from files

The first thing you will notice when looking at the code provided is that it has two commands right at the beginning that you haven't seen so far.

```
import java.io.BufferedReader;
import java.io.FileReader;
```

These tell java to include the corresponding classes. In this case these classes provide the basic functionality of reading data from files such as text files. We use the functionality that these classes provide in the provided method **readFromFile** which takes as input a filename of type **String** and goes through this file line by line, adding each line to the data structure that we will code up next using **add(line)**.

1 The Class SortedArray

The first thing you will need to do is write to complete the methods in class `SortedArray`. This data structure consists of three fields `String[] arr`, `int max`, and `int size`. The array of strings will hold the words from our list of taboo words. The first integer will be the maximum number of words that we can store in the array of strings. The second integer will reflect the actual number of words that we have stored.

1.1 Method SortedArray, getSize, and add

Let's start by adding some basic functionality to our code:

- Write a constructor method `public SortedArray(int max)` that takes one argument, `max`, and initializes `String[] arr`, `int max`, and `int size` so that the string array `arr` has size `max`, `max` stores the value `max`, and `size` is zero.
- Write a method `public int getSize()` that takes no argument and returns the size `size` of the array of strings `arr`.
- Write a method `public void add(String str)` that takes one argument, `str`, and adds it at the end of the array of strings `arr` and also updates `size` accordingly.

1.2 Method sort

Now, let's do something more useful. Namely, you are asked to write a method that sorts the array of strings `arr`. You can use whichever method you prefer. For example, one of the methods that we have seen in the lecture. To get the full points the method should run in $O(n \log n)$ time (either in the worst case or on average) and should sort the array "in place" (i.e., without copying the array or larger parts of it into a new array). Both, running time and space complexity, will make a difference here as we have more than 200K taboo words!

An important difference to the solutions that we have seen in the exercises is that this time we are sorting strings and not integers. You will find the following method useful for comparing strings

```
str1.compareTo(str2),
```

where both `str1` and `str2` are strings. The above expression will be smaller than 0 if `str1` is lexicographically smaller than `str2`, equal to 0 if they are the same, and larger than 0 otherwise. So for instance `"abc".compareTo("acd") > 0` will evaluate to `false` as `"abc"` is lexicographically smaller than `"acd"`.

1.3 Methods findLinearSearch and findBinarySearch

Let's also add two methods for searching for a string `str`. Both methods should return `true` if the array of strings contains that string, and `false` otherwise. The search methods can be described informally as follows:

Linear search is a method for finding a particular value in an array that checks each element in sequence until the desired element is found or the array is exhausted.

For binary search, the array should be arranged in ascending order. In each step, the algorithm compares the search key value with the key value of the middle element of the array. If the keys match, then a matching element has been found. Otherwise, if the search key is less than the middle element's key, then the algorithm repeats its action on the sub-array to the left of the middle element or, if the search key is greater, on

the sub-array to the right. If the remaining array to be searched is empty, then the key cannot be found in the array.

For binary search it will be the easiest to write two methods

```
public boolean findBinarySearch(String txt), and  
public boolean findBinarySearch(String txt, int low, int high).
```

The former is called once initially and calls the second with `low = 0` and `high = size-1`, and the second then repeatedly calls itself with updated `low` and `high` as described above.

Class PasswordChecker

Now comes the fun part! Let's use our class `SortedArray` to perform some basic checks on a given password, passed to our main method. What the main method does is it reads the filename from the command line and it also reads in a password. It then stores them in `String path` and `String pwd`, respectively. Note the `toLowerCase()` that turns the password that we actually typed in into lowercase. This makes sense as our list of taboo words contains lower words, too. The idea is that we do consider both **London**, **london** as well as **LoNdOn** as unsafe, and like this we do not need to have all of them in our list of taboo words.

Your final task is to write a method `checkPassword` that takes two arguments, `SortedArray a` and `String pwd` and returns a string message. Specifically, we will consider a password unsafe if one of the following holds:

- The password is 7 or less characters long.
- It is one of the words from the list of taboo words.
- It is composed of two words from the list of taboo words.

Your password **London** would fail all tests, while **LondonParis** would fail only the third test. Finally, **LondonParisFrankfurt** would be considered safe. (But please don't rely on this!) For the first check you can use `pwd.length()` to get the length of the string stored in `pwd`. For the second one our find methods from above will be convenient, while for the last you probably want to use

```
String substring = pwd.substring(i,j); ,
```

which stores position *i* to *j* of `pwd` into `substring`. For example, `pwd.substring(0,2)` of **London** would be **Lon**.

The messages that the method returns should reflect at least one of the reasons why the password failed in case we consider it as unsafe. It would be even better to have it reflect all reasons.

Note: Feel free to explore other checks but do not hand them in. Another common “mistake” is to combine words with numbers, such as the name of your significant other followed by the year the two of you first met. Also, noting how quick our program runs, you may re-consider the choice of your password :)