

Homework 1

Due: 4pm on November 2, 2017 (Thursday)

Objectives:

- Review basic concepts and limitations of arrays
- Reinforce the concept of Object-Oriented Programming

In this homework, you are to implement a Java class

```
public class MyArray
```

You are NOT allowed to use any of the Java Collections framework. If you have any imports going on in your class, there will be a big deduction. Instead, you must use a `String[]` array as its underlying data structure to store and manipulate words in your class. As we discussed in class, a typical array should never allow any holes. Thus, at any point, this underlying array should never have any holes such as putting nulls in the middle of indices with strings, etc. Similar to Java's `ArrayList` class, 0 can be passed as its initial capacity which must be used as the value of the length of the underlying array that is being instantiated in constructors.

You must provide the following constructors and methods in your `MyArray` class.

```
public MyArray()
public MyArray(int initialCapacity)
public void add(String text)
public boolean search(String key)
public int size()
public int getCapacity()
public void display()
public void removeDups()
```

Step-by-step guide:

- Think carefully to find out what methods you need to implement. Additional information about all of the necessary methods that you are to implement can be found in `HW1Driver.java` file.
 - When inserting new words, duplicates are allowed.
 - And add method should take care of validating words.
 - If `NULL` is passed for add and search methods, do not throw exceptions.
 - The underlying `String[]` array should double its length when necessary.
 - The only time to remove duplicates in the array is when `removeDups()` method is invoked.
 - The `size()` method's worst-case running time complexity must be $O(1)$.
 - You are NOT allowed to use the Java Collections Framework.
 - `Arrays` class is a part of the Java Collections Framework. That means you cannot use `Arrays` class in your code. Once again, there should be no imports in your code.
 - You are NOT allowed to use any other data structures than one `String[]` array.
 - This also means you are not allowed to use two or more `String[]` arrays. Especially, think carefully about how you can implement your `removeDups()` method without having any temporary or auxiliary array(s).

- When increasing, you do need to create another array. But once done increasing, then there should be only one array.
- You are NOT allowed to use any sorting algorithms.
- You are more than welcome to use private helper methods. In fact, we encourage you to do so.
- Your clients may choose to convert words in a file to lowercase before inserting them. Thus, you should not convert strings to lowercases in your implementation.
- Implement the class on Eclipse or any other IDEs you prefer.
 - In your JavaDoc comments, clearly write worst-case running time complexity using Big O notation for each of the public methods.
 - Submit the source file on Autolab.
 - Follow the Java Coding Conventions as described in <http://www.oracle.com/technetwork/java/javase/documentation/codeconvtoc-136057.html>
 - In terms of setting a proper indentation rule in Eclipse, please refer to the attached document called "IndentationConfigurationinEclipse.pdf"
- Test your code extensively!
 - When you run the program with the given file, childrensbible.txt file, you should see the output that is in HW1Driver.java file
 - For display method, your code should print all the words in one line and put a space between words.
 - A word is a sequence of letters [a..zA..Z] that does not include digits [0..9] and the underscore character. If it is not a word defined here, gently ignore them. Do not throw an exception.

Deliverables:

- Your source code file. (Make sure to include your Andrew ID and NAME in the JavaDoc comment using author tag. And, make sure to remove package.) Submit your source code file (MyArray.java) using Autolab (<https://autolab.andrew.cmu.edu>).
- Make sure to remove all the print statements that you put for your own debugging.
- The result output generated by Autolab grader may not have a lot of details and that is intentional. They are good enough for you to narrow down your debugging process to fix any issue(s). Hence, we encourage you to spend some time to think about edge cases you might have missed before resubmitting it.

Grading:

Autolab will grade your assignment as follows.

- Working code: 90 points
- Coding conventions: 10 points
 - We'll deduct one point for each coding convention issue detected.

In case you do not pass test case(s), please spend some time to think about edge cases you may have missed and test thoroughly before asking questions to the TAs and resubmitting.

Autolab will check your coding conventions quite strictly. Make sure to read the Java Coding Conventions document when in doubt.

Autolab will show you the results of its grading within approximately a few minutes of your submission. You may submit multiple times so as to correct any problems with your assignment. Autolab uses the last submission as your grade.

Also, the TAs may take a look at your source code to check correctness and design. The most important criterion is always correctness. Buggy code is useless (even if you may think a found bug is very minor). It is also important that your code be readable and well organized. This includes proper use of clear comments. Points will be deducted for poor design decisions and unreadable code.

As mentioned in the syllabus, we will be using the [Moss](#) system to detect software plagiarism. Any cheating incident will be considered very seriously. The University also places a record of the incident in the student's permanent record.

Late submissions will NOT be accepted and, if you have multiple versions of your code file, make sure you do **submit the correct version. Make sure to submit .java file. Do not submit .class file!**