CSE 274 - Lab 1 - Array Implementation of a List

In this lab, you will implement an array-based list of words (Strings). A *WordList* class has been created, but the constructors and methods need to be implemented. We will implement a few of these together.

Notes about the List abstract data type:

- Order is important
- Duplicates are allowed
- Many methods try to give back useful information in their return statements. Some examples that might not be intuitive:
    - *add("cat")* will return a boolean for whether or not the add was successful
    - *remove("cat")* will return a boolean for whether or not the remove was successful
    - *remove(3)* will return the string in position 3
    - *set(5, "cat")* will change the string in position 5 to "cat" and *return* the word that used to be in that position
- We should expect exceptions to be thrown when a parameter value does not make sense. In this case, the only parameter that might be bad is the index for an operation: this index might be out of bounds.

Notes about an array-based implementation:

- In this lab, the array size is fixed. That means, there is a maximum size, and if we reach that size, we cannot add any additional elements. If the client tries to add to a full list, an exception is not thrown, but the add method should return false to indicate that the add failed.
- There is a difference between the *size* of the list, and the length of the array. The length of the array never changes in our implementation. But the size does. When the size of the list is equal to the length of the array, we know that the list is full.
- If an operation is attempted when the specified index is out of bounds, throw an *IndexOutOfBoundsException*.
- When items are added to the list, it may be necessary to shift other items to the right.
- When items are removed from the list, it is important to set unused locations in the array to null. The reason this is important is for proper memory management. We don't want unneeded references to point to objects that are no longer in the list. This allows for proper "garbage collection" in Java. So…
    - Each *remove()* method, if successful, should set one memory location to *null*.
    - The *clear()* method should set all locations to *null*.
- The purpose of *toArray()* is to give back an array of only the items in the list. It is not supposed to give back the array instance variable.

Assignment:

1. Create an Eclipse project called Lab_01
2. Download Tester.java and WordList.java from the lab assignment
3. Drag and drop them in your src folder of your Lab_01 project
4. Implement and test all constructors and methods. A suggested starting order:
    - Both constructors
    - *add()* - The version that takes just a String parameter
    - *toArray()* - Makes it easy to see how your methods are working
    - *contains()*, *size()*, *clear()* - These should be easy to implement
    - The rest of the methods, testing as you go.
5. Test thoroughly. Pay careful attention to edge cases:
    - What happens when you add to a full array?
    - What happens when you remove from an empty list?
6. When things seem to be working properly, upload *WordList.java* to Canvas