Penn Engineering
ONLINE LEARNING

CIT 594 Module 1 Programming Assignment

In this module, we have seen how an ArrayList works by performing operations using an underlying array. In this assignment, you will modify the implementation of an ArrayList so that it uses Java Generics and has additional functionality.

Learning Objectives

In completing this assignment, you will:

- Understand the implementation of an ArrayList in Java
- Gain familiarity with reading and modifying existing code
- Be sure that you can submit code to the course's autograding platform

Getting Started

Download the **MyArrayList.java** file, which contains the unimplemented methods for the code that you will write in this assignment. Be sure that you are able to compile this code, and review the implementation before proceeding.

As you work on this assignment, please do not change the signatures of any of the methods (their parameter lists, names, and return value types) and do not create any additional .java files for your solution. Additionally, please be sure that your *MyArrayList* class is in the default package, i.e. there is no "package" declaration at the top of the source code.

Development Environment

For this and all other assignments in this course, we recommend that you work on your local computer using a development environment such as Eclipse, or whichever IDE you used for any prior Java courses.

Although you will need to upload your solution to Codio for grading (see "How to Submit" below) and could develop your solution on that platform, we recommend using industry standard tools such as Eclipse so that you become more used to them and can take advantage of their features.

If you have trouble setting up your IDE or cannot get the starter code to compile, please post a note in the discussion forum and a member of the instruction staff will try to help you get started.

Penn Engineering
ONLINE LEARNING

Activity

In this assignment, you will modify the *MyArrayList* class to give it some extra functionality.

Step 1. Use Java Generics

Modify the *MyArrayList* class so that it uses Java Generics instead of Strings. That is, it should be possible to create an instance of the class like this:

```
MyArrayList<Integer> list = new MyArrayList<>();
```

Note that you will need to make changes throughout the class definition. Make sure that all methods accept or return objects of the parameterized type, and not just Strings.

Step 2. Implement `remove` method

Implement the `remove` method so that it takes an object of the parameterized type (from Step 1) as its parameter, and removes the first instance of an equivalent element in the underlying array, i.e. for which the `.equals` method returns true.

Once the object has been removed, the elements following it should be shifted over to fill the empty space, and the method should return true; if the object does not exist in the underlying array, the method should simply return false.

For instance, assume the underlying array contained these values in this order:

“Cat”	“Dog”	“Banana”	“Aardvark”	null	null

If the method `remove("Banana")` were called, then the array should look like this:

“Cat”	“Dog”	“Aardvark”	null	null	null

And the method should return true.

Step 3. Shrink array when it is too large

Modify the two `remove` methods (the one we provided and the one from Step 2) so that they shrink the length of the underlying array when, after removing an element, the number of elements in the array is less than or equal to 25% of the array’s length. In this case, the underlying array should be shrunk to half its current length.

For instance, assume the underlying array contains these values in this order:

“Ant”	“Bat”	“Cow”	null	null	null	null	null

And assume that “Bat” was removed, either by its index or its value. Then the array should look like this:

“Ant”	“Cow”	null	null	null	null	null	null

Because the number of elements (2) is now less than or equal to 25% of the length of the array (8), the array should be cut in half, and should then look like this:

“Ant”	“Cow”	null	null

That is, the elements should stay the same, but the underlying array should now have a length of 4 instead of 8.

Hint: Review the code that is used to grow the underlying array and think about how you can write similar code to shrink it.

Step 4. Implement `set(index, value)` method

Implement the `set` method so that it takes an integer `index` and a `value` of the parameterized type (from Step 1) as its parameters and replaces the element in the underlying array at that index with the new value. The method should then return the old value, i.e. the one that was replaced.

For instance, assume the underlying array contained these values in this order:

“Cat”	“Dog”	“Banana”	“Aardvark”	null	null

If the method `set(1, "Monkey")` were called, then the array should look like this:

“Cat”	“Monkey”	“Banana”	“Aardvark”	null	null

And the method should return “Dog” since that is what was replaced.

If the index is out of range, then the method should throw an IndexOutOfBoundsException.

Step 5. Create constructor to initialize underlying array

Create a constructor for this class that takes an array of the parameterized type (from Step 1) as its parameter and initializes the values in the *MyArrayList*’s underlying array.

If the argument to the constructor is null, then the *MyArrayList* constructor should behave as the default (no-argument) constructor does.

Initializing the *MyArrayList* elements sounds relatively straightforward, but note that it should be possible for the caller of this constructor to modify the elements of the input array argument without changing what's in the *MyArrayList* and vice-versa.

For instance, consider the following code:

```
String[] data = {"cat", "dog", "elephant"};
MyArrayList<String> list = new MyArrayList<>(data);
```

If the caller then invokes

```
data[1] = "bat";
```

this should not change what's in `list`.

Likewise, if the caller invokes

```
list.set(0, "mouse");
```

this should not change what's in `data`.

Before You Submit

Please be sure that:

- your *MyArrayList* class is in the default package, i.e. there is no "package" declaration at the top of the source code
- your *MyArrayList* class compiles and you have not changed the signature of the methods we have provided
- you have not created any additional .java files

How to Submit

After you have finished implementing the *MyArrayList* class, go to the "Module 1 Programming Assignment Submission" item and click the "Open Tool" button to go to the Codio platform.

Once you are logged into Codio, read the submission instructions in the README file. Be sure you upload your code to the "submit" folder.

To test your code before submitting, click the "Run Test Cases" button in the Codio toolbar; this will run the tests that are used to grade your code.

You'll see quite a bit of output, even if all tests pass, but at the bottom of the output you will see the number of successful test cases and the number of failed test cases.

You can see the name and error messages of any failing test cases by scrolling up a little to the "Failures" section.

If all tests are successful, then you would earn 100% on this assignment; there are no “hidden tests.” Note, though, that this will **not** be the case in subsequent assignments, which **will** have “hidden tests.”

Assessment

This assignment is scored out of a total of 34 points.

Step 1 is not graded; however, the test cases will not compile if you have not correctly used generics and you will receive a 0 on this assignment if this part is not completed.

Step 2 is worth a total of 12 points, based on whether your implementation correctly removes the element, updates the size, and moves other elements as a result.

Step 3 is worth a total of 6 points, based on whether your implementation correctly modifies the size of the underlying array.

Step 4 is worth a total of 6 points, based on whether your implementation correctly sets the element and handles error cases.

Step 5 is worth a total of 10 points, based on whether your implementation correctly initializes the underlying array and handles error cases.