# Practice Coding Assignment 12 : Fancy Integer ArrayList

**This assignment is entirely optional and designed to give you practice writing code and applying lessons and topics for the current module.**

This homework deals with the following topics:

- ArrayLists
- Overloading
- Unit Testing

## The Assignment

In this assignment, you will implement a class called *CustomIntegerArrayList*. This class represents a fancy ArrayList that stores integers and supports additional operations not included in Java's built-in ArrayList methods.

For example, the *CustomIntegerArrayList* class has a "splice" method which removes a specified number of elements from the CustomIntegerArrayList, starting at a given index. For a CustomIntegerArrayList that includes: 1, 2, 3, 4, and 5, calling splice(1, 2) will remove 2 items starting at index 1. This will remove 2 and 3 (the 2nd and 3rd items).

The CustomIntegerArrayList class has 2 different (overloaded) constructors. (Remember, an overloaded constructor is a constructor that has the same name, but a different number, type, or sequence of parameters, as another constructor in the class.) Having 2 different constructors means you can create an instance of the CustomIntegerArrayList class in 2 different ways, depending on which constructor you call. Both constructors have been implemented for you. (See below)

Internally, the *CustomIntegerArrayList* class uses a private ArrayList variable named "arr" to store its integer elements. The "CustomIntegerArrayList()" constructor initializes the ArrayList variable "arr" as an empty ArrayList and the "CustomIntegerArrayList(ArrayList<Integer> arr)" constructor initializes the ArrayList variable "arr" with the elements in the given ArrayList. (Again, see below)

```
/**
 * Fancy ArrayList that stores integers and supports additional
operations not included in Java's built-in ArrayList methods.
 */
public class CustomIntegerArrayList {

     // instance variables
```

```
	/**
	 * Internal ArrayList of elements.
	 */
	private ArrayList<Integer> arr;


	// constructors

	/**
	 * Creates a new empty CustomIntegerArrayList.
	 */
	public CustomIntegerArrayList() {
		this.arr = new ArrayList<Integer>();
	}

	/**
	 * Creates a new CustomIntegerArrayList with the elements in
the given ArrayList.
	 * @param arr with elements for the CustomIntegerArrayList
	 */
	public CustomIntegerArrayList(ArrayList<Integer> arr) {
		this.arr = new ArrayList<Integer>(arr);
	}
```

There are 8 (mostly overloaded) methods that need to be implemented in the CustomIntegerArrayList class. (Again, an overloaded method is a method that has the same name, but a different number, type, or sequence of parameters, as another method in the same class.)

- getArrayList() - Returns the elements.
- get(int index) - Returns the element at the specified index from the elements.
- add(int element) - Appends the given element to the end of the elements.
- add(int index, int element) - Inserts the given element at the specified index.
- remove(int index) - Removes the element at the specified index.
- remove(int num, int element) - Removes the specified number of the given element from all elements.
- splice(int index, int num) - Removes the specified number of elements from all elements, starting at the given index.
- splice(int index, int num, int[] otherArray) - Removes the specified number of elements from all elements, starting at the given index, and inserts new elements in the given array at the given index.

Each method has been defined for you, but without the code. See the javadoc for each method for instructions on what the method is supposed to do and how to write the code. It should be

clear enough. In some cases, we have provided hints and example method calls to help you get started.

For example, we have defined a “remove” method for you (see below) which removes a specified number of a given element from the CustomIntegerArrayList. Read the javadoc, which explains what the method is supposed to do. Then write your code where it says “// TODO” to implement the method. You’ll do this for each method in the program.

```
/**
 * Removes the specified number (num) of the given element from the
internal ArrayList of elements.
 * If num <= 0, don't remove anything.
 * If num is too large, remove all instances of the given element
from the internal ArrayList of elements.
 *
 * @param num number of instances of element to remove
 * @param element to remove
 */
public void remove(int num, int element) {

    // TODO Implement method
}
```

In addition, you will write unit tests to test your method implementations. Each unit test method has been defined for you, including some test cases. First make sure you pass all of the provided tests, then write **additional and distinct test cases** for each unit test method.

For example, we have defined a “testRemoveInt” method for you (see below) which tests the “remove” method. Pass the tests provided then write additional tests where it says “// TODO”. You’ll do this for each unit test method in the program.

```
@Test
void testRemoveInt() {

    CustomIntegerArrayList arr1 = new CustomIntegerArrayList();
    arr1.add(0, 2);
    arr1.add(0, 3);
    arr1.add(0, 4);
    arr1.remove(0);
    arr1.remove(1);

    ArrayList<Integer> lst1 = new ArrayList<Integer>();
    lst1.add(0, 2);
    lst1.add(0, 3);
    lst1.add(0, 4);
```

```
    lst1.remove(0);
    lst1.remove(1);

    assertEquals(lst1.get(0), arr1.get(0));

    // TODO write at least 3 additional test cases
}
```

The main method calls have been provided for you (see example below). They are designed purely to call the other methods in the program, and to help you run the program. Nothing needs to be completed in the main method.

```
public static void main(String args[]) {

    //create new empty CustomIntegerArrayList
    CustomIntegerArrayList arr1 = new CustomIntegerArrayList();

    //add element
    arr1.add(2);

    //get internal arraylist of elements
    System.out.println(arr1.getArrayList()); //[2]
}
```

## Submission

You have been provided with *CustomIntegerArrayList.java* and *CustomIntegerArrayListTest.java*. To complete the assignment, implement the methods in *CustomIntegerArrayList.java*, making sure you pass all the tests in *CustomIntegerArrayListTest.java*. Then write **at least 3 additional and distinct test cases** for each unit test method in *CustomIntegerArrayListTest.java*. Do not modify the name of the methods in *CustomIntegerArrayList.java* or the automated testing will not recognize it.

You will submit two files for this assignment: *CustomIntegerArrayList.java* and *CustomIntegerArrayListTest.java*. Make sure your program and the unit testing files run without errors! Submit the completed program using the steps outlined in the assignment in Coursera.

## Evaluation

**This assignment is evaluated, so that you can assess how well you understand the material. However, the evaluation of the assignment will not affect your course grade.**

*Points:*

1. Does your code function correctly? (24 pts)
    - getArrayList() - 3 pts
    - get(int index) - 3 pts
    - add(int element) - 3 pts
    - add(int index, int element) - 3 pts
    - remove(int index) - 3 pts
    - remove(int num, int element) - 3 pts
    - splice(int index, int num) - 3 pts
    - splice(int index, int num, int[] otherArray) - 3 pts
2. Did you include at least 3 additional distinct and valid test cases for each test method? Do all of your tests pass? (16 pts)
    - testGetArrayList() - 2 pts
    - testGet() - 2 pts
    - testAddInt() - 2 pts
    - testAddIntInt() - 2 pts
    - testRemoveInt() - 2 pts
    - testRemoveIntInt() - 2 pts
    - testSpliceIntInt() - 2 pts
    - testSpliceIntIntIntArray() - 2 pts