# NumberList Lab

You will be implementing a new class called **NumberList** that will represent a list of integers. A **`NumberList`** object has just one instance variable, which is a reference to an array of **`int`** values. Below are the "stages" in which I suggest you design and test things…

1. Look at the constructor that takes no arguments. Note that it must still instantiate the array "values" as an array of size zero.
2. Implement a constructor that takes an **`int`** array as an argument. You will need to:

   Have the **`values`** reference refer to a new array that is the same size as the parameter.

   Copy the elements from the parameter array into that new array.
3. Implement a copy constructor that takes another **`NumberList`** reference as an argument. You will need to:

   Have the **`values`** reference of the new object refer to a new array that is the same size as the one held by the parameter object.

   Copy the elements from the parameter object's **`values`** array into the new object's **`values`** array.
4. Run the public tests on the submit server to be sure your constructors are working – if they aren't then you won't be able to pass any of the subsequent tests.

---

5. Implement the **`getSize`** method. It simply returns the size of the list.
6. Implement the **`getAt`** method. It returns the element whose position in the list matches the parameter. We are using 0-based indexing. If the parameter is negative or exceeds the last index of the list then throw an **`IndexOutOfBoundsException`** with a simple message inside it.
7. Run the public tests on the submit server and confirm these methods work as expected.

---

8. Implement the **`getSum`** method. It will return the sum of all entries in the list.
9. Implement the **`contains`** method. It will only return true if the parameter is in the list.
10. Implement the **`add`** method. It needs to add the parameter to the end of the list. In order to be able to accomplish this, you will need to**:**
    a. Create another array that is one unit larger than the existing array.
    b. Copy all of the elements from the existing array over to the new array.
    c. Add the parameter to the end of the new array.
    d. Update the instance variable **`values`** so that it refers to the new array.

NOTE: Perhaps write your own JUnit tests and ***then*** "release test" to check these methods. The release tests will not run unless all of the public tests pass.