# Vectors - Implementing a Vector

Nada Basit and Mark Floryan

January 20, 2022

## 1 SUMMARY

For this homework, you will be writing a vector data structure in Java. Your summary:

1. Download the starter code and import the project into Eclipse
2. Implement the Vector.java class
3. Verify your implementation using the provided tester class
4. **FILES TO DOWNLOAD:** Vectors.zip
5. **FILES TO SUBMIT:** Vector.java

### 1.1 VECTOR.JAVA

Your only task is to implement all of the methods in Vector.java. This class will implement the provided List interface, which is duplicated for your convenience here. Your task is to implement each of these methods.

```
1 public interface List<T> {
```

```
3       /**
         * Returns the size of this list, i.e., the number
5        * of nodes currently between the head and tail
         * @return
7        */
        public int size();
9
        /**
11       * Clears out the entire list
         */
13      public void clear() ;

15      /**
         * Inserts new data at the end of the
17       * list (i.e., just before the dummy tail node)
         * @param data
19       */
        public void insertAtTail(T data);
21
        /**
23       * Inserts data at the front of the
         * list (i.e., just after the dummy head node
25       * @param data
         */
27      public void insertAtHead(T data);

29      /**
         * Inserts node such that index becomes the
31       * position of the newly inserted data
         * @param data
33       * @param index
         */
35      public void insertAt(int index, T data);

37      public T removeAtTail();

39      public T removeAtHead();

41      /**
         * Returns index of first occurrence of
43       * the data in the list, or -1 if not present
         * @param data
45       * @return
         */
```

```
47      public int find(T data);

49      /**
         * Returns the data at the given index, null if
51       * anything goes wrong (index out of bounds, empty list, etc.)
         * @param index
53       * @return
         */
55      public T get(int index);
}
```

You'll probably want to add a couple of supporting methods. For example, a resize() method is useful for when the underlying array of the vector needs to grow.

## 1.2 Testing Your Implementation

Once you implement these methods, you can test them by running the main method in the provider TestVector.java class. This simple tester will execute the methods in your list and compare them to those in Java's built in List to make sure the results are as expected.

One strategy might be to implement your methods in the order that they are tested. That way, you can see the tester say "this method is correct" before moving on to the next one.

Notice that we expect your Vector to be a generic class, meaning any type of Object can be stored within your Vector.

**Important Note:** The provided tester is NOT perfect. It is possible (not likely, but possible) that the tester will report that a method is correct, not noticing that there is some very minor issue that will appear later. For example, if the Vector insert appears to work but the state of some internal variables you are using are not correct. This might not reveal itself until you start removing. So, it COULD be the case that the error is in insert but the tester doesn't see the error until removing items. Be mindful of this. The tester is a great guide for finding errors in your implementation but it is not perfect.

You should submit one file for this homework, **Vector.java**.

## 1.3 Gradescope

You should submit your code to *Gradescope*. If you are having trouble with your submission, you should double check the following common problems:

1. Make sure you are only submitting one file, and it is called *Vector.java* exactly.

2. Make sure you **keep** the *package vector;* statement at the top of your file. This autograder expects your file to be in that package.

3. Make sure your file *does not* contain a main method.

4. Make sure your Vector class is *Not printing anything to the console.* This will mess up the autograder.