List Implementation Assignment: Part #1

Re-submit Assignment

Due Apr 13, 2017 by 11:59pm **Points** 100 **Submitting** a file upload
File Types zip

Overview

We will be implementing a list data structure for this assignment, which does not store duplicate values. The goal is to understand the List<T> API in Java, as well as exploring the amount of work that each operation in the List<T> API requires.

Setup

First download the following starter files and add them to Eclipse: **files**.

As seen below, each student has been randomly assigned one of the following lists implementations: ArrayList<T> or LinkedList<T>. Your job is to implement your assigned list.

Name	List Type
ANTONYUK SOFIYA	linked list
BACK JAMES	array list
BENNER ELIZABETH	array list
BOURQUE MICHELINE	linked list
BRADEN KYLE	array list
CALLAHAN PATRICK	array list
CHURCH JEFFREY	linked list
GINN THEODORE	array list
HASCUP NATHANAEL	array list
HAWKS JOSHUA	linked list
LEANO MARLENE	array list
LOCKE KEVIN	array list
MANALO JEREMY	array list
MCCOY JONNATHON	linked list
MOON SSONNI	array list
NGUYEN DUC	array list
NGUYEN KEVIN	linked list
NOBLE NEAL	linked list
OSTRANDER CALEB	array list
PADILLA ELENA	linked list
PHAM CYNTHIA	linked list
PRATT JEFFREY	array list
ROUSH TIMOTHY	linked list
SAYLOR BRIAN	linked list
STRAND NATHAN	array list
TAYLOR BRENT	array list

Note: Developing the wrong list will result in an immediate zero. No exceptions! So make sure you know which list you are implementing.

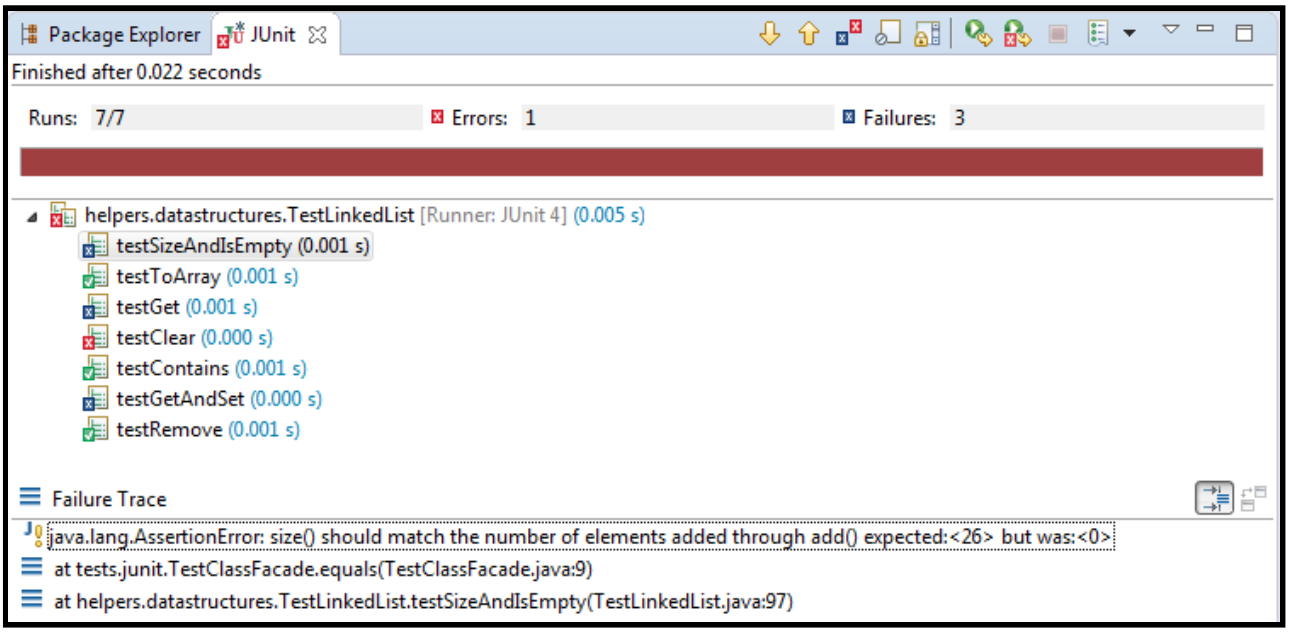
Note: A primary requirement of this assignment is to maintain the list with no duplicate values (as a mathematical set). Any student ignoring this requirement will also receive a zero.

JUnit & Unit Tests

Part of this assignment is to formally verify your list's functionality using a well-known testing regime called unit-testing. With unit testing we separate our code into small executable units (usually the contents of a method) and then test that unit (method). I have already written several unit tests in the project above with a framework called JUnit.

It is quite easy to add JUnit to any new Eclipse project and to begin writing tests. How to write effective tests that provide coverage of your application is one of the difficulties of unit testing and requires a certain mindset.

For now, I have already written tests for each of the methods in your list class. You only need to download the project above and run the file called ListTests.java located in the part1 package. Your results will look something like the screenshot below:



When running the tests, each test will result in one of the following colors:

- **Green - Success:** This means that my test has invoked several of the methods in your list class to verify that a single unit (method) is working correctly. For example, I may call the add() method to add a single element to your list and then call get() & size() to verify that the element is actually present in your list.
- **Blue - Failure:** This means that my test failed, which means your method is written incorrectly. In the example above this might mean that the size() method returns zero after an element has been added to the list (this should not be the case)
- **Red - Exception:** This means that an uncaught exception was thrown during my tests. This means that either your method that is being tested is throwing an exception, or my test is due to the results I am receiving from your list.

Note: Pay particular attention to the test failure-text that is shown with each failed test (red or blue).

Methods

This first assignment focuses on (roughly) one-third of the methods in the List<T> interface. You should implement each of the methods below.

public boolean add(T element)

Appends the specified element to the end of this list, but only if the element is not already present in the list.

Parameters:

`element` - element whose presence in this collection is to be ensured

Returns:

true if this collection changed as a result of the call

public T get(int index)

Returns the element at the specified position in this list.

Parameters:
`index` - index of the element to return
Returns:
the element at the specified position in this list
Throws:
`IndexOutOfBoundsException (https://docs.oracle.com/javase/7/docs/api/java/lang/IndexOutOfBoundsException.html)` - if the index is out of range (index < 0 || index >= size())

---

public boolean contains(Object element)

Returns true if this collection contains the specified element.

Parameters:
`element` - element whose presence in this collection is to be tested
Returns:
true if this collection contains the specified element

---

public boolean isEmpty()

Returns true if this collection contains no elements.

Returns:
true if this collection contains no elements

---

public int size()

Returns the number of elements in this collection.

Returns:
the number of elements in this collection

---

public void clear()

Removes all of the elements from this collection.

---

public boolean remove(Object element)

Removes the first instance of the specified element from this collection, if it is present.

Parameters:

`element` - element to be removed from this collection, if present

Returns:

true if an element was removed as a result of this call

public Object[] toArray()

Returns an array containing all of the elements in this collection.

Returns:

an array containing all of the elements in this collection

*Note: the remaining methods will be implemented in part #2 of this assignment which can be found here: **List Implementation Assignment: Part #2***

Submission

- Each of your methods should pass the tests given in the sample project (i.e. they should be green when the tests are ran).
- Submit your entire project to the Canvas drop-box zipped.
- Remember to follow our style guide and include Javadocs for your list class!

List Implementation Part #1 Rubric

Criteria	Ratings		Pts
List structure uses generics to store elements.	10.0 pts Full Marks	0.0 pts No Marks	10.0 pts
add() method adds only unique elements to the list and returns true if successful.	10.0 pts Full Marks	0.0 pts No Marks	10.0 pts
size(), clear() & isEmpty() methods correctly track the number of elements in the list.	10.0 pts Full Marks	0.0 pts No Marks	10.0 pts
remove() method removes elements and returns true if the element was found.	15.0 pts Full Marks	0.0 pts No Marks	15.0 pts
toArray() method returns a new array without any null elements.	10.0 pts Full Marks	0.0 pts No Marks	10.0 pts
get() method returns the element at a given index. get() throws an IndexOutOfBoundsException if the index given is invalid.	10.0 pts Full Marks	0.0 pts No Marks	10.0 pts
contains() returns true for elements in the list and false otherwise.	10.0 pts Full Marks	0.0 pts No Marks	10.0 pts
Styles: naming conventions, brace placement, spacing, commented code, redundancy, packages and magic numbers.	15.0 pts Full Marks	0.0 pts No Marks	15.0 pts
Formal documentation: full Javadocs & comment block header.	10.0 pts Full Marks	0.0 pts No Marks	10.0 pts
			Total Points: 100.0