CMPSCI 187 / Fall 2015
Linked List Recursion

Due on November 6, 2015 4PM EST

Mark Corner and Gerome Miklau

CMPSCI 187

Contents

Overview

In this assignment, you will implement the List abstract data type, as specified in `structures.ListInterface`. Your implementation will be based upon references — that is, you'll be building a singly linked list. You'll implement all methods that must traverse the list using *recursion* rather than iteration. Finally, you'll need to write your own tests for the code; we will not be providing a comprehensive test suite.

Learning Goals

- Show ability to write a generic class that implements a given interface, fulfilling the contract it specifies (including $O()$ behavior and an `Iterable` implementation).
- Additional practice implementing recursive methods.
- Additional practice writing appropriate unit tests.

General Information

Read this entire document. If, after a careful reading, something seems ambiguous or unclear to you, then communicate to the course staff immediately.

Start this assignment as soon as possible. Do not wait until 5pm the night before the assignment is due to tell us you don't understand something, as our ability to help you will be minimal.

Reminder: Copying partial or whole solutions, obtained from other students or elsewhere, is academic dishonesty. Do not share your code with your classmates, and do not use your classmates' code. If you are confused about what constitutes academic dishonesty you should re-read the course syllabus and policies. We assume you have read the course information in detail and by submitting this assignment you have provided your virtual signature in agreement with these policies.

You are responsible for submitting project assignments that compile and are configured correctly. If your project submission does not follow these policies exactly you may receive a grade of zero for this assignment.

Policies

- For some assignments, it will be useful for you to write additional java files. Any java file you write that is used by your solution MUST be in the provided `src` directory you export.
- The course staff are here to help you figure out errors (not solve them for you), but we won't do so for you after you submit your solution. When you submit your solution, **be sure to remove all compilation errors from your project**. Any compilation errors in your project will cause the autograder to fail, and you will receive a zero for your submission. **No Exceptions!**

Test Files

In the `test` directory, we provide several JUnit test cases that will help you keep on track while completing the assignment. We recommend you run the tests often and use them to help create a checklist of things to do next. *But you should be aware that we deliberately do not provide you the full test suite we use when grading.*

We recommend that you think about possible cases and add new `@Test` cases to these files as part of your programming discipline. Simple tests to add will consider questions such as:

- Do your methods handle edge cases such as integer arguments that may be positive, negative, or zero. Many methods only accept arguments that are in a particular range.
- Does your code handle unusual cases, such as empty or maximally-sized data structures?

More complex tests will be assignment-specific. To build good test cases, think about ways to exercise methods. Work out the correct result for a call of a method with a given set of parameters by hand, then add it as a test case. Note that we will not be looking at your test cases (unless otherwise specified by the assignment documentation), they are just for your use.

Before submitting, make sure that your program compiles with and passes all of the original tests. If you have errors in these files, it means the structure of the files found in the `src` directory have been altered in a way that will cause your submission to lose some (or all) points.

Problem 1

Import Project into Eclipse

Begin by downloading the starter project and importing it into your workspace. It is very important that you **do not rename** this project as its name is used during the autograding process. If the project is renamed, your assignment will not be graded, and you will receive a zero.

The imported project may have some errors, but these should not prevent you from getting started. Specifically, we may provide JUnit tests for classes that do not yet exist in your code. You can still run the other JUnit tests.

The project should normally contain the following root items:

`src` This is the source folder where all code you are submitting must go. You can change anything you want in this folder (unless otherwise specified in the problem description and in the code we provide), you can add new files, etc.

`support` This folder contains support code that we encourage you to use (and must be used to pass certain tests). You must not change or add anything in this folder. To help ensure that, we suggest that you set the support folder to be read-only. You can do this by right-clicking on it in the package explorer, choosing Properties from the menu, choosing Resource from the list on the left of the pop-up Properties window, unchecking the Permissions check-box for Owner-Write, and clicking the OK button. A dialog box will show with the title “Confirm recursive changes”, and you should click on the “Yes” button.

Problem 1 continued on next page. . .

`test` The test folder where all of the public unit tests are available.

`JUnit 4` A library that is used to run the test programs.

`JRE System Library` This is what allows Java to run; it is the location of the Java System Libraries.

If you are missing any of the above or if errors are present in the project (other than as specifically described below), seek help immediately so you can get started on the project right away.

Implement the ListInterface

Implement the `ListInterface` provided in a class called `RecursiveList` in the `structures` package. Be sure to place it in the appropriate place under the `src/` directory, and not in the `support/` or `test/` directories.

In your implementation, you may not use explicit iteration (i.e. **`for`**, **`while`**, or **`do while`** loops) of any kind. **If you submit an implementation using explicit iteration, you will receive a zero for the assignment.**

Be sure that your implementation complies with the required big-O runtime bounds in the comments of each method. You will not pass all of the autograder tests if your methods are not within these bounds.

Hints: Recursion is all about reducing some problem slightly to make it easier to solve. Ask yourself, "What method would make this easier?" For example, the `indexOf` method signature is written in such a way that it is not going to be possible to do recursion on a `Node`. However, there is nothing stopping us from writing a private helper method that we can use internally. Perhaps something with a method signature like:

```
private int indexOf(T toFind, Node<T> toCheck, int currentIndex)
```

We can then simply ask, does `toCheck` hold `toFind`? If it does, we can return `currentIndex`. Otherwise, we recurse on the next node in the list, at the next index, with the same `toFind` value. What should we do if `toCheck` is null?

You'll need to come up with other such "helper" methods that lend themselves to recursion. You can't modify the interface, but that doesn't stop you from adding these new helper methods.

Throw an `UnsupportedOperationException` when implementing the `remove` method in the required `Iterator`. You need not attempt to handle the nuances of mid-iteration element deletion.

We have provided you with an absolutely minimal set of tests. You will need to come up with some of your own. Try to think of all of the strange cases that could occur, and write tests to check for each of them. You will not be graded on your tests, but writing them and passing them is the only way that you can be reasonably sure that your code works.

Export and Submit

When you have completed the changes to your code, you should export an archive file containing the entire Java project. To do this, click on the `recursive-list-student` project in the package explorer. Then choose "File → Export" from the menu. In the window that appears, under "General" choose "Archive File". Then choose "Next" and

Problem 1 continued on next page...

enter a destination for the output file. Be sure that the project is named **`recursive-list-student`**. Save the exported file with the `zip` extension (any name is fine).

Once you have the zip file ready, you can submit it to the online autograder. Go to autograder.markdcorner.com create an account with the correct email address and student id. Select the right project from the drop down and submit. It shouldn't take more than a minute to grade your project, but if there is a backlog of projects then it may take longer. If you encounter any errors that look like the autograder failed and not your project, please let the instructors know.