Data Structures

Linked Train Cars - 10 Course Points

Overview

The LinkedTrainCars class is meant to implement a Singly Linked list, where each node contains a String.

The only class attribute is a reference to the front of the list, and the completed remove() method is given.

You must fill in the code for the **numCars()** and **insertAt()** methods, to finish the linked list and enable the train car Driver visualization.

Directions

- DO NOT add new import statements.
- DO NOT add any new class attributes
- DO NOT change any of the method's signatures.
- DO NOT modify the given constructor.

To complete this Lab, you must implement the numCars() and insertAt() methods for the singly linked list of strings. Each node in this singly linked list is represented as a TrainCar, with a String name and a next field (which points to another train car).

The method signatures are already given, do not modify these or add any new ones.

There is one class attribute provided for you in LinkedTrainCars.java:

- "front" which is a reference to the front of the singly linked list

The given constructor initializes the list with one starting node, you do NOT need to edit or use this constructor.

numCars()

This method is a counter method, which returns the number of TrainCar nodes that are currently in the list.

Declare a TrainCar variable and assign it to the "front" of the list. This gives you a pointer which you can use to traverse the nodes. Also create a int variable with the value 0, to count the cars.

While that pointer is not null, count the current car and then use ptr = ptr.getNext() to traverse to the next car.

Once you've traversed the whole list, return the number of cars.

insertAt(String add, int index)

This method inserts a new Node containing the given string at the given index.

We count indicies starting at 1 (instead of starting with 0 like arrays). So the front node is index 1, second node is index 2, etc...

We DO allow duplicates in this linked list, so we do NOT need to check for them before inserting.

To complete this method, first create a new TrainCar node and set its name to the given String.

– If "front" is null, then the list is empty. So simply set front equal to your new TrainCar.

Else, the list is not empty, so we need to insert at the given index. Thus, we need to traverse to the node BEFORE that index.

– Grab TWO pointers to the front of the list. One of these will be traversing to the current index (which we want to insert at), and the second pointer will be one behind it (aka the previous node).

– Use a while loop, and while the CURRENT pointer is not null AND the current index is not the target index:

  – Set the "prev" pointer equal to the current pointer.

  – Set the current pointer equal to its next node.

  – Increment the index counter.

This will make the loop stop either when:

1) You reach the end of the list (and the target index is > length of list)

2) Or you reach the target index

Then, simply insert the target node AFTER the previous pointer.

Driver

Once you implement your code, you can run the LinkedTrainCars.java file and interact with the driver. It will show a visualization of an linked list. You can use the add button to add any car at any index. There are two remove buttons: one which removes the specified index, and the other which removes the specified car.

You can use this Driver to test your code, but it is still useful to write JUnit test cases, in case the driver does not catch all cases.

Testing Your Code

We can test this linked list code using a test class, since the TrainCar nodes can contain any String.

In the main project folder, there exists a "tests" folder next to the "src" folder. This contains JUnit tests, which can run and test pieces of your code. To compile and run these tests, you must install the JUnit package. See the Programming FAQ for more info on VScode extensions.

You are provided with a premade JUnit test class for LinkedTrainCars.java. Within this test class there is a COMPLETED test for numCars() and an INCOMPLETE test stub for insertAt().

You should implement the latter using JUnit tests (assertTrue(), assertFalse(), assertEquals()) in order to test your code as you write it. You do NOT need to submit this test file.

See the JUnit Testing Guide to learn more about JUnit tests. See the Debugging Guide to learn more about debugging.

Tests not running?

First, make sure you're in the right folder and the tests are implemented. Next, if you have the "Code Runner for Java" or Oracle "Java" extension, make sure you uninstall those extensions. Remember that you must fill in the tests for push() and resize().

VSCode Extensions

You can install VSCode extension packs for Java. Take a look at this tutorial. We suggest:

- Extension Pack for Java
- Project Manager for Java
- Debugger for Java
- Test Runner for Java

Importing VSCode Project

1. Download LinkedTrainCars.zip from Autolab Attachments.
2. Unzip the file by double-clicking.
3. Open VSCode
    - Import the folder to a workspace through **File** > **Open Folder**

Executing and Debugging

- You can run your program through VSCode or you can use the Terminal to compile and execute. We suggest running through VSCode because it will give you the option to debug.
- How to debug your code
- If you choose the Terminal:
    - first navigate to **LinkedTrainCars** directory/folder
        - to compile: **javac -d bin src/singly/LinkedTrainCars.java**
        - to execute: **java -cp bin singly.LinkedTrainCars**
        - **NOTE: if you have LinkedTrainList (2) -> LinkedTrainList or CS112 -> LinkedTrainList in VS Code, open the INNERMOST LinkedTrainList through File -> Open Folder.**

Before submission

COMMENT all printing statements you have written from LinkedTrainList.java

Collaboration policy. Read our collaboration policy here.

Submitting the assignment. Submit *LinkedTrainList.java* separately via the web submission system called Autolab. To do this, click the *Labs and Assignments* link from the course website; click the *Submit* link for that assignment.

Getting help

If anything is unclear, don't hesitate to drop by office hours or post a question on Piazza.

- Find instructors office hours *here*
- Find tutors office hours on Canvas -> Tutoring
- Find head TAs office hours *here*
- In addition to office hours we have the Coding and Social Lounge (CSL) , a community space staffed with ilab assistants which are undergraduate students further along the CS major to answer questions.

By Colin Sullivan