CS2230 Computer Science II: Data Structures
Homework 3
Linked list of web browser tabs

Due February 15, 2017, 11:59pm

Goals for this assignment

- Implement methods for a linked list that involve manipulating references
- Debug your code using JUnit tests

Submission Checklist

You should submit only one file: **LinkedList.java**. Upload it on ICON under Assignments > Homework 3. Physical paper copies or images are not acceptable.

- Do the tests pass?
- Double check the file you submitted to ICON. Is it the version you expected to submit?

Description

In this project, you will implement a LinkedList class (in LinkedList.java) with various methods for manipulating it. There will be two ways to test your implementation. First, we've provided a complete JUnit test file. Second, we've provided an application called Browser, which uses your LinkedList to display and manipulate tabs like a web browser would.

As you implement more methods of LinkedList, you will pass more of the tests and be able to use more of the commands in the Browser application. **You do not need to change any files except for LinkedList.java** (unless you want to add some additional tests to LinkedListTest.java).

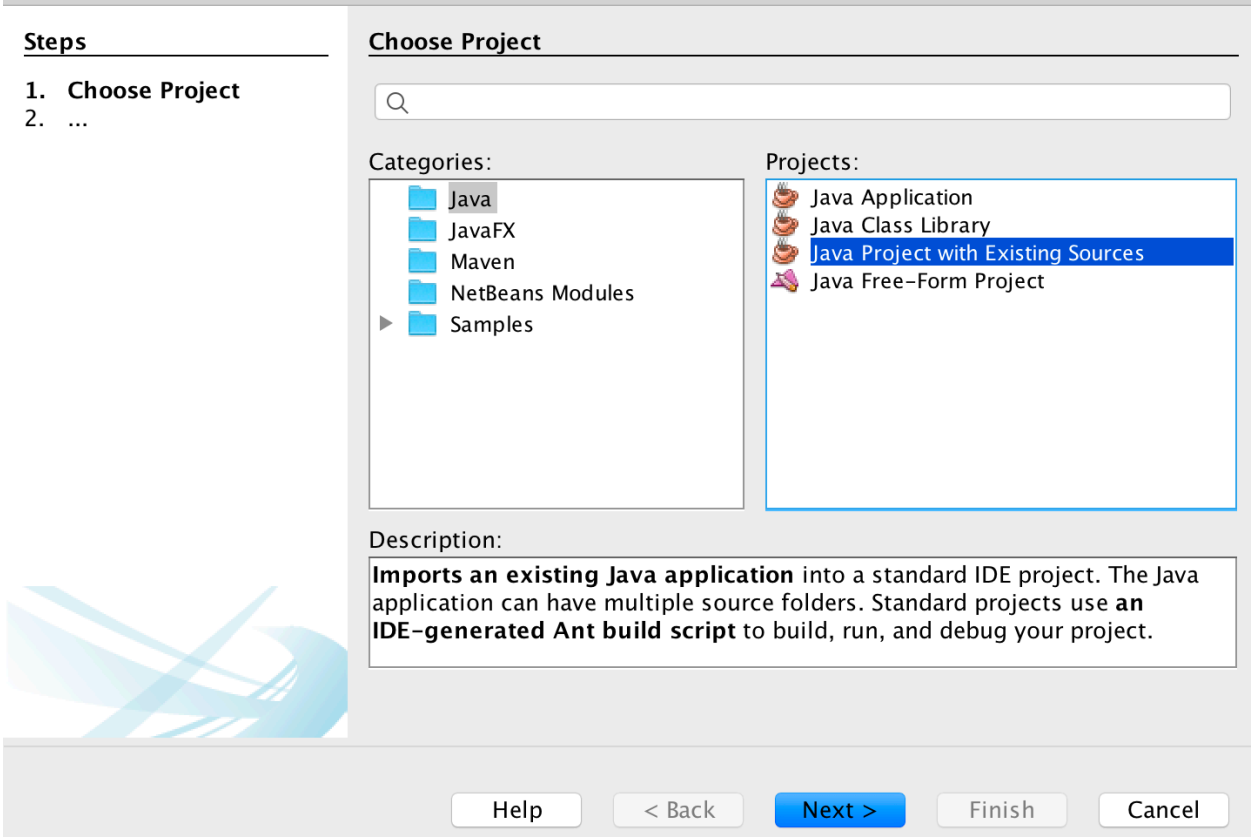
This project has a setup part and 4 parts. The best way to approach the project is in the order of these parts. You should try to pass all the tests specified in Part 1 before continuing to Part 2, and so forth. **A submission that passes the tests in Part1 and doesn't implement anything else will score better than a submission that attempts all 5 parts and passes zero tests.**

Also, see the last section "Helpful Tips" for information that is useful for all the parts.

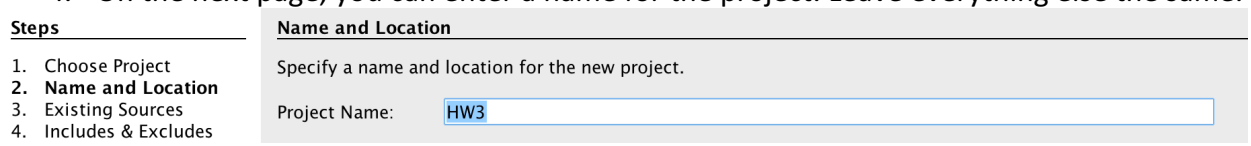
Part 0: Setup the project in NetBeans

In this assignment, we have provided starter code for a whole NetBeans project.

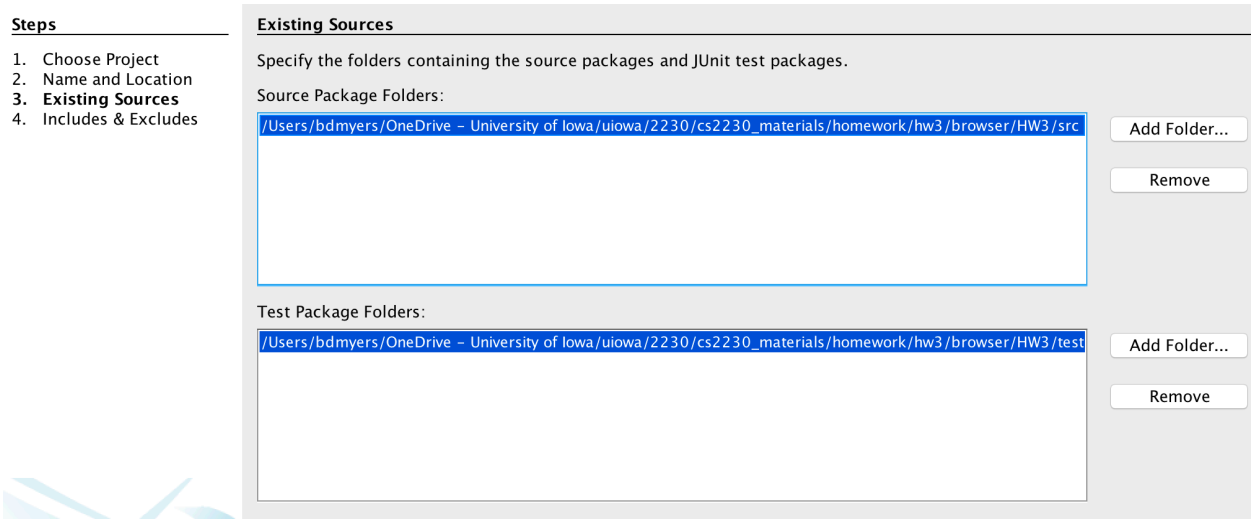
1. Download and unzip hw3.zip
2. Open NetBeans
3. New Project, then pick Java > Java Project with Existing Sources



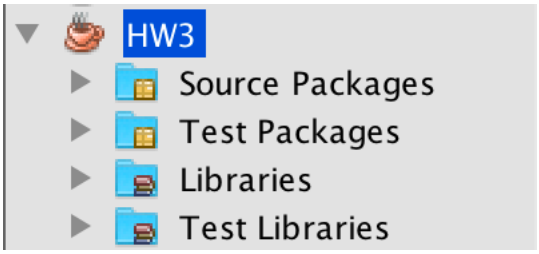
4. On the next page, you can enter a name for the project. Leave everything else the same.



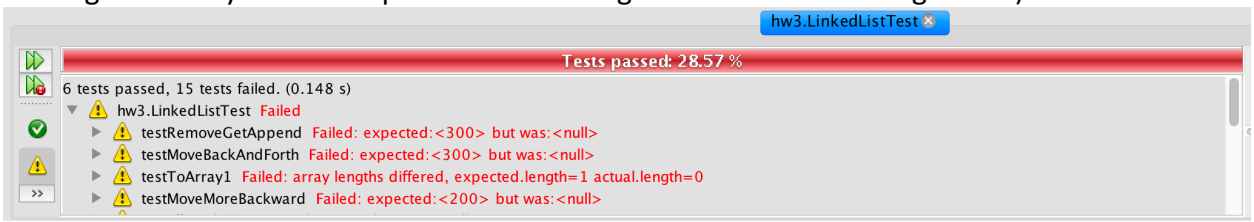
5. One the next page, browse for the src/ folder and test/ folder inside of the HW3 folder that you downloaded and unzipped.



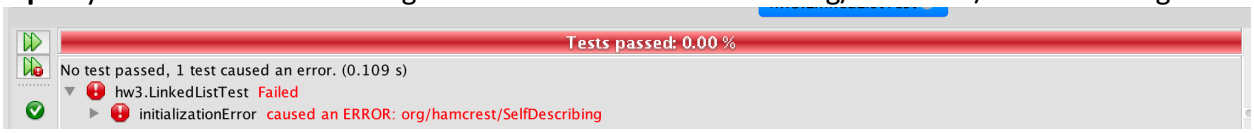
6. If a window pops up about problems, you should click Resolve Problems...
7. You should now see HW3 project under Projects.



8. Run the tests by right clicking on LinkedList.java and choosing Test File. You should see the failed tests. (Exact messages will vary from this screenshot, but you should see messages that say "Failed: expected <something> but was: <something else>")



Tip: If you see this error message instead "caused an ERROR: org/hamcrest/SelfDescribing"



Then you need to right click Test Libraries > Add Library > pick Hamcrest 1.3 > click Add Library. Now try testing LinkedList.java again.

Part 1: constructor, add, and toArray

End result of Part 1:

- pass these tests
  - testToArrayEmpty
  - testToArray1
  - testToArrayMore
  - testSizeEmpty
  - testAddSize
- make these Browser commands work
  - new
  - count

In this part, you must fill in the implementation of the LinkedList constructor, LinkedList.add, and LinkedList.toArray. See the comments above each method in LinkedList.java for details. Notice that we provided the inner class ListNode already. It's data field is type Object, which means that you can store anything in the LinkedList. In the tests, we store integers in the LinkedList, but in the Browser application we store Tabs.

Once you implement these methods, the above tests and functionality should work.

Trying the browser commands (new, count)

To run the Browser application, right click Browser.java > Run file. The application will print out the commands you can run.

```
new <url> / close <num> / display <num> / move <from> <to> / navigate
<num> <url> / count / exit
```

You should be able to type something like

```
new www.uiowa.edu
```

and see the output

```
|0 The University ... |
new <url> / close <num> / display <num> / move <from> <to> / navigate
<num> <url> / count / exit
```

Each time you enter the new command, a new browser tab is added to the end.

```
new stackoverflow.com
|0 The University ... | |1 Stack Overflow |
```

```
new <url> / close <num> / display <num> / move <from> <to> / navigate
<num> <url> / count / exit
```

You can also run the count command.

```
|0 The University ... | |1 Stack Overflow |
new <url> / close <num> / display <num> / move <from> <to> / navigate
<num> <url> / count / exit
count
You currently have 2 tabs open.
|0 The University ... | |1 Stack Overflow |
new <url> / close <num> / display <num> / move <from> <to> / navigate
<num> <url> / count / exit
```

If you try to run the other commands (other than exit), expect to see some kind of error or strange behavior, since you haven't implemented enough of LinkedList yet.

Part 2: get

End result of Part 2:

- pass these tests
  - testGet1
  - testGetBigger
- make these Browser commands work
  - display

In this part, you must fill in the implementation of the LinkedList.get method. See the comments above the method in LinkedList.java for details.

Once you implement this method, the above tests and functionality should work.

Trying the browser commands (display)

In the Browser application, you can use `display` to open up the desired web page. Use the number next to the tab's title.

```
|0 The University ... | |1 Stack Overflow |
new <url> / close <num> / display <num> / move <from> <to> / navigate
<num> <url> / count / exit
display 1
```
(will open up stack overflow in your system's default web browser)

Part 3: replace

End result of Part 3:

- pass these tests
    - testReplace1
    - testReplaceMore
    - testReplaceSize
- make these Browser commands work
    - navigate

In this part, you must fill in the implementation of the LinkedList.replace method. See the comments above the method in LinkedList.java for details.

Once you implement this method, the above tests and functionality should work.

Trying the browser commands (navigate)

In the Browser application, you can use `navigate` to have a Tab go from one website to another. Use the number next to the tab's title and the url of the new website.

```
|0 The University ... | |1 Stack Overflow |
new <url> / close <num> / display <num> / move <from> <to> / navigate
<num> <url> / count / exit
navigate 1 acm.org
|0 The University ... | |1 Association for... |
new <url> / close <num> / display <num> / move <from> <to> / navigate
<num> <url> / count / exit
```

Part 4: remove

End result of Part 4:

- pass these tests
    - testRemove1
    - testRemoveBigger
    - testRemoveAndGet
    - testRemoveGetAdd
    - testSizeRemove
    - testReplace (because it has remove)
- make these Browser commands work
    - close

In this part, you must fill in the implementation of the LinkedList.remove method. See the comments above the method in LinkedList.java for details.

Once you implement this method, the above tests and functionality should work.

Trying the browser commands (close)

In the Browser application, you can use `close` to remove the desired tab. Use the number next to the tab's title.

```
|0 The University ... | |1 Stack Overflow |
new <url> / close <num> / display <num> / move <from> <to> / navigate
<num> <url> / count / exit
close 0
|0 Stack Overflow |
new <url> / close <num> / display <num> / move <from> <to> / navigate
<num> <url> / count / exit
```

Part 5: move

End result of Part 5:

- pass these tests
    - all of the remaining tests
- make these Browser commands work
    - move

In this part, you must fill in the implementation of the LinkedList.move method. See the comments above the method in LinkedList.java for details.

Once you implement this method, the above tests and functionality should work.

Trying the browser commands (close)

In the Browser application, you can use `move` to move the desired tab to the desired location. Move takes two integers. The first integer specifies the tab to move, and the second number specifies the position to move it to.

```
|0 Stack Overflow | |1 The University ... | |2 Wikipedia | |3
Association for... |
new <url> / close <num> / display <num> / move <from> <to> / navigate
<num> <url> / count / exit
move 3 1
|0 Stack Overflow | |1 Association for... | |2 The University ... | |3
Wikipedia |
```

```
new <url> / close <num> / display <num> / move <from> <to> / navigate
<num> <url> / count / exit
move 1 2
|0 Stack Overflow | |1 The University ... | |2 Association for... | |3
Wikipedia |
new <url> / close <num> / display <num> / move <from> <to> / navigate
<num> <url> / count / exit
```

Moving is *not* swapping

As you can see from the above examples, the examples in LinkedList.move comments, and the tests, move does not swap two tabs; rather, it moves a tab to a new location.

TIP: If you are having trouble with implementing move, try to break the problem down into simpler steps that you can solve individually.

Helpful Tips

- **A lot of LinkedList's public methods need similar things**. It may be useful to create private helper methods, such as one that returns a reference to the ith ListNode (like LinkedList.get except it returns the ListNode instead of the Object stored in it).

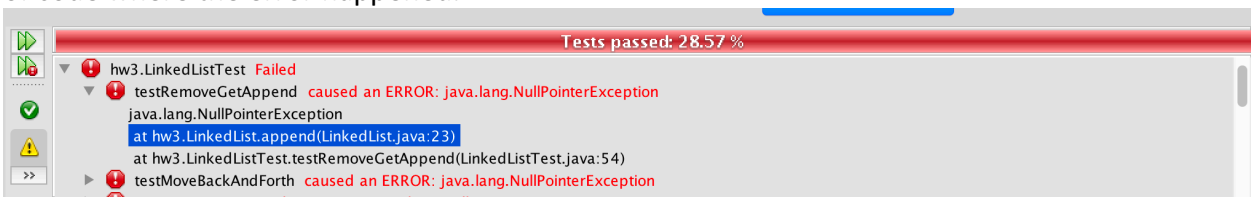
- **For each method, write down your algorithm on paper with boxes and arrows before writing code.** And, try your paper algorithm on different cases. We cannot emphasize this point enough, so it is repeated again below.

- **For each method, write down your algorithm on paper with boxes and arrows before writing code.** Did you do it this time?

- **What you can do when a test fails.**

First, great job. That's not sarcasm. If you made a test fail, you are one step closer to a working program. Almost no program works the first time around.

Pick one test to try to fix. Expand its message. Double-clicking the top line in the *stack trace* (i.e., the list of method calls that your program was in when it crashed) will take you to the line of code where the error happened.



If the error is a NullPointerException, this means you used dot (.) on a reference that is currently null. Either it is okay that the reference is null and you just shouldn't be using dot on it, or the reference became null by some error in your code.

If the error is a JUnit assertion error (actual and expected results are different), then you must reason about the current state of the LinkedList and why the method would return the wrong value. If you need to print the state of your linked list and your toArray method works, then one easy way to print it out is with `System.out.println(Arrays.toString(ll.toArray)).`

Debugger

The debugger is an alternative to inserting print statements into your code. It allows you to stop the program at the line you want and then inspect the current values of variables.

To use the Java debugger in NetBeans, run the tests using Debug Test file instead of Test file. The debugger will only be useful when using Debug Test file if you set a *breakpoint* in your program. A breakpoint is a place where the program will stop running and give control to the debug, where you can step through the program and check variable values.

Set a breakpoint by clicking in the left grey sidebar of the NetBeans editor.