The University of Birmingham
School of Computer Science

Assignment 01

1 Introduction

This assignment will be marked out of 11 and **DOES NOT** contribute to your module mark. As such it is an opportunity to practice preparing a submission, using `Eclipse` and the `JUnit` and `log4j` libraries in earnest as well as practicing applying your knowledge to the linked list data structures that you have been studying in the theory part of the module.

2 Background

A linked list data structure is well defined, for example in your theory handouts for this module, and generally very useful. However one often comes across situations when one needs a datastructure that has very similar behavior to a standard one such as a linked list, but has some differences from the standard representation. In such cases, it may not be possible to use the linked list class from the standard Java library but you should be able to design your own data structure for the problem you need to solve, drawing on your understanding of linked lists and how they work. This assignment explores such a scenario.

We consider a situation in which we need to maintain a collection of elements on two lists at the same time. In our case, an ordered list, where the elements are always kept in order, and a *Most Recently Used (MRO)* list, where the order of the elements is updated every time an element is *touched* to be at the front of the list. Thus each element will be on both lists but in different orders. In particular, a key requirement we have is to be able to find an element on one list by following a chain of *next* pointers, and when we have found the one we want, to be very able to remove it, very efficiently from the other list.

We could just use two separate linked lists from the standard Java library to accomplish this, but it does not support our key requirement well: Once we have found the element we want on the first list, we would then have to do a linear search on the second list to find the node on the second list that we want to delete. This linear search is (we assume for the purposes of this assignment) too slow.

Our solution is to develop a bespoke data structure to handle a node being on two separate doubly-linked lists at the same time. Thus each node has two forward pointers and two backward pointers. Your job is to fill in the missing bits of the code to make this work.

3 Setup and Specification

To make things a bit simpler, you have been provided with a Java project folder that contains sources and tests for this assignment: see the `dsa_assignment1` folder in the `dsa_2019` GIT repository (don't forget to do a "`git pull`" inside your copy of the repository to update it). You have been provided with the following files and packages:

```
dsa_assignment1/src/main/java/dsa_assignment1:
    MLNode.java
    MLNodeInterface.java
    OrderedMruList.java
    OrderedMruListInterface.java

dsa_assignment1/src/test/java/dsa_assignment1:
    MLNodeTest.java
    OrderedMruListTest.java
```

If you set the top level `dsa_assignment1` as an Eclipse project, with `src/main/java` and `src/test/java` as the source folders on your build path, and add the user libraries for hamcrest-all (or hamcrest-core), JUnit and Log4J to your project, you should be able to compile and run the tests (nearly all of which should fail!)

Your task is to add missing code to the methods in `MLNode.java` and `OrderedMruList.java` to make all the tests pass.

- The precise information about the behaviour required of the missing code is detailed in the Javadoc comments in the two interface files: `MLNodeInterface.java` and `OrderedMruListInterface.java`

- You should **NOT** use any of the Standard Java Library collection classes in your code
- You should not modify any code outside of these two files
- You should not use any print statements to any files or to standard out or standard error streams: if you want to have some debug output, use the logging calls for Log4j
- For marking purposes, your code will be compiled and executed against a test set in a secure sandbox environment. Any attempts to break the security of the sandbox will cause the execution of your program to fail. This includes infinite loops in your code.
- When you have completed your code changes to your satisfaction, make a zip archive of the `dsa_assigmnment1` package under `dsa_assignment1/src/main/java` **ONLY**.

  That is, change directory to `.../dsa_assignment1/src/main/java`, then run the command

  ```
  zip -r 1234567.zip dsa_assignment1
  ```

  where 1234567 should be replaced by your student id number, and submit the resulting .zip file to Assignment1 on Canvas.

  Note that it is **ONLY** the `dsa_assignment1` package under your `.../dsa_assignment1/src/main/java` folder that should be put in the zip archive. you should **NOT** include anything from the testing part of the project, or your compiled class files etc.

4 Marking

Note that marks for this assignment are not counted towards your final module mark. However, for feedback purposes, there are 10 tests in the testing part of the assignment, you will earn one mark for each test passed. You will also earn a further mark if your submission is structured as required and compiles correctly. If your submission is not structured correctly or does not compile, I make no promises that you will earn ANY marks.

5 Notes and Tips

- Start with the `MLNode.java` changes first: these have to work anyway before you will be able to get any of the (non-initialization) tests working in `OrderedMruList.java`.
- Even if you don't get much working, please do submit so that you can see how the process works and what kind of feedback the system will give you.
- The longest method in my solution had 8 lines in its body: and that includes one blank line. Most of the methods have between 1 and 5 lines.
- You only need to modify the methods which have the `WRITE THIS CODE` comments. Note that, in those methods, I typically added dummy return statements to make the code compile. Many of these return statements will need to be modified.

6 Deadline

The deadline for this assignment is:

- Monday 28th January at 10:00

Your submission will be marked and feedback will be returned to you. A solution will also be made available after the deadline.