CS392F: Automatic Software Design

P5: Feature House Assignment

Xiaohui Chen

[xhchen0328@utexas.edu](mailto:xhchen0328@utexas.edu)

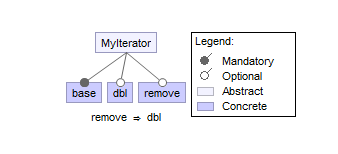
Jianyu Huang

[jianyu@cs.utexas.edu](mailto:jianyu@cs.utexas.edu)

**Part 1**

Description:

The feature model of linked list is shown as follows:



The base feature is a singly linked list (class MyNode only has right pointer) while dbl feature adds the left pointer in MyNode class to make it double linked list. The remove feature adds the remove method in MyIterator class and delete method in MyList class. Also note that there is a constraint “remove implies dbl”, which means the selection of remove feature means the selection of double linked list.

In the base feature, the remove method in MyIterator is an empty method. This is because the MyIterator class implements Iterator interface and remove method is required to be implemented. Here we just leave it blank.

There are some helper methods which do not exist in the original code given, such as getHead() method in MyNode class. Those helper methods are for JUnit testing, which will be described later. There will also be helper methods in all features and all parts of this assignment to aid testing.

The dbl feature adds the global variable left of type MyNode in class MyNode. It serves as the left pointer. Also insert method in MyList class is modified to let left pointer correctly set up during insertion.

Finally the remove feature modify the remove method in MyIterator and add a method called delete in MyList to support linked list deletion.

In part 1, Entity class serves as the Object to be in the linked list. It is indeed a pair of name and age. There are also two static arrays of Entity for usage in Main class. The Main class first inserts array one to the linked list. After printing out array one, it further inserts array two into the linked list. Note that without remove feature, the removal of the second array from the linked list is not executed since there is no remove functionality available.

Therefore the correct output for Main method when remove feature is selected is:

original list

(Chili, 20)

(Beth, 22)

(Scarlett, 7)

(Chief, 3)

(Steve, 90)

(Don, 60)

augmented list

(Kelsey, 25)

(Haggis, 1)

(Chili, 20)

(Beth, 22)

(Scarlett, 7)

(Chief, 3)

(Steve, 90)

(Don, 60)

The correct output for Main class when remove feature is selected is:

original list

(Chili, 20)

(Beth, 22)

(Scarlett, 7)

(Chief, 3)

(Steve, 90)

(Don, 60)

augmented list

(Kelsey, 25)

(Haggis, 1)

(Chili, 20)

(Beth, 22)

(Scarlett, 7)

(Chief, 3)

(Steve, 90)

(Don, 60)

revised list

(Chili, 20)

(Beth, 22)

(Scarlett, 7)

(Chief, 3)

(Steve, 90)

(Don, 60)

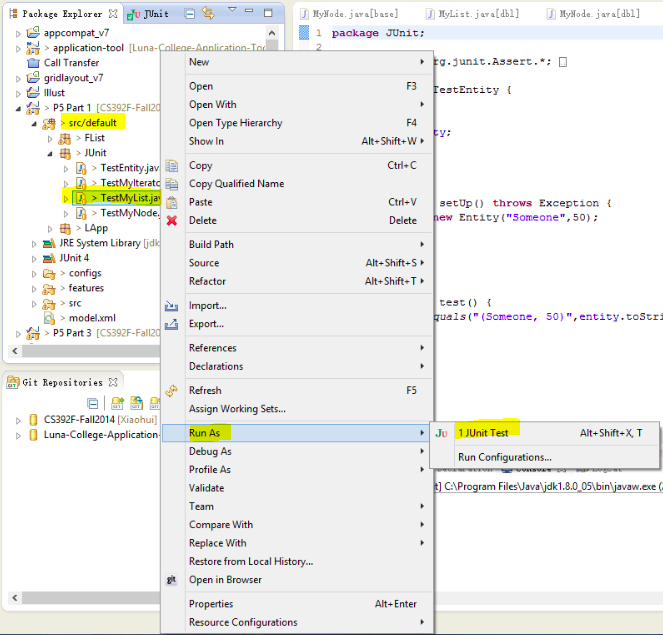
This matches the expected output

JUnit Test:

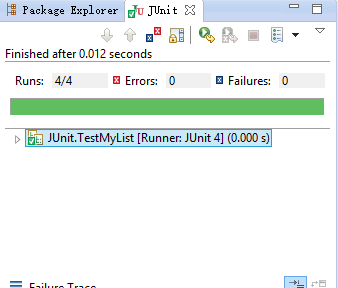
The JUnit test files are stored in JUnit folder in each feature. The JUnit tests mainly test the return values by the methods in MyNode, MyList, Entity and MyIterator classes. Because Main class itself is a testing case, there is no JUnit test for the Main class.

We mainly use three Entity Objects for testing: (Someone, 50), (Somebody, 100) and (ThirdParty, 80). Through insertions, deletions and manipulations (e.g check the left/right pointer of a certain node), we compare the outputs with the expected ones. In fact, the JUnit tests pass without any error. Note that the JUnit tests are slightly different because the JUnit code is modified by FeatureIDE according to the features selected.

To run the JUnit test after selecting the deisired features, go to src/JUnit and find the desired JUnit testing case. Then right click the file and run as JUnit Test, as shown below.



The example output of JUnit Test is shown as follows.



The green bar means all the test cases in the JUnit class file have passed. In fact, our program passes all the JUnit tests.

**Part 2**

(to be finished)

**Part 3**

Description

The feature model in part 3 is the same as that of part 1. However, in the implementations, classes MyNode, MyList and MyIterator all support generic type <T>, in all features. Therefore, in the Main class, those objects have generic type <Entity>. The rest of the implementation is the same as that of part 1. The output of Main class is the same as that of part 1. Therefore, the output matches the expectation.

JUnit Testing

Instead of using Entity objects, we use Integer 80,50, and 100 for the JUnit testing. Undoubtedly, the MyNode, MyList and MyIterator all have generic type <Integer>. The rest of the implementation is the same as that of part 1. In fact all the JUnit tests passed and the program which support generic type linked list is correct.