COMP 182
PP 2: LinkedLists
Due Tuesday, Feb 28 by 11:59 pm

What to submit in Canvas?

Updated version of your COMP182LinkedList.java and ListTest.Java, both in a zip file. You also need to upload the PDF version of COMP182LinkedList.java file separately, not in the zip file (Copy everything in a word file and convert it to PDF).

If you encounter any problems in completing the assignment or in the submission process, please don't hesitate to ask for help. The sooner, the better!

Instructions

There are two files with the project, that are all the files you will need for this project. Open the file COMP182LinkedList.java. **All the methods that you need to write for this project must be added to this file.**
ListTest.java is where you will find a main method that you will use to test the new linked list methods that you have created in COMP182LinkedList.java. For most of the methods proper testing is provided. You need to add testing for two more methods. More details below.

You are not to use any other java classes than the two classes provided. If you used any other classes on your code, your project will not be graded.

MAKE SURE TO COMMNET YOUR CODE!

The Linked List

In this project, you will work with a similar LinkedList class that we practiced during the lectures. Few of the methods in COMP182LinkedList.java are already implemented and ready to use. Your job is to add more methods to the COMP182LinkedList class that you downloaded from Canvas. Once finished, you'll have a pretty good start on a fully-fledged generic linked list class!
During the last few lectures, we discussed the generic (parameterization). When you want to create a class that can accept any data type you want, you need to parameterize your solution. In our implementation of LinkedList we used <T> to denote this.

The Methods

There are a couple of new methods that you will be writing for this project. Some are easy, some are hard. I have provided code that tests most of the methods by calling them from the main method in the file ListTest.java. The test file will only work when you have done all the methods. In COMP182LinkedList.java I have provided the signature (definition and input parameters) for these methods. At this point to make sure you can run the file (COMP182LinkedList.java that you downloaded from Cavas) I have added template values as return types of the methods. For example, if the method returns an int then I added return 0 as the return value.

Here are the methods are as follows:

```
Anything getFirst()
Anything getLast()
void add(Anything value)
void addAfter(int index, Anything value)
Anything set(int index, Anything newValue)
void removeAll(Anything value)
lastIndex(Anything value)
COMP182LinkedList<Anything> clone
boolean equals(Object o)
COMP182LinkedList <Anything> split()
boolean hasCycle()
Node reverseList()
Node oddEvenList()
```

`Node deleteDuplicates ()`
`boolean isPalindrome()`

*For the last two methods, you need to add proper testing to ListTest.Java

---

Method Specifications

Here are more details for implementing each of these methods. You need to complete all the following methods in the COMP182LinkedList class. You can write the methods in any order you wish, but I suggest you do them in the order they are listed.
In particular, make sure all your methods work for empty lists, lists of different sizes, etc. as you can see from the sample test cases provided in ListTest.java. While grading your project, we will make changes to these values! Also, make sure that your solutions do not modify the original lists (unless you are specifically instructed to do so).

void main(String[] args)

You can use this method in the ListTest class to create lists as needed and test each of the methods listed above. I have provided a few test cases, but you may want to add more to some of the method tests. Until you get add working, you will need to use addFirst to create lists to test your other methods.

Anything getFirst()
This method returns the value stored in the first node in the list. It should print an error message and return null if the list is empty.

Anything getLast()
This method returns the value stored in the last node in the list. It should print an error message and return null if the list is empty.

void add(Anything value)
This method adds a node containing new value to the end of the list.

void addAfter(int index, Anything value)
This method adds a node containing new value after the given index.

Anything set(int index, Anything newValue)
This method replaces the data in the node at position index with newValue, if such a position is in the list and returns the previous (old) value that was at that location. Prints an error message and returns null if index is out of range.

void removeAll(Anything value)
This method removes every node that contains value in the list. Realize that the value can occur multiple times and anywhere in the list! This method does nothing if value is not in the list.

int lastIndex(Anything value)
This method returns the last index in which value id found. Realize that the value can occur multiple times and anywhere in the list!

COMP182LinkedList<Anything> clone()
This method returns a new copy of the LinkedList. It creates a copy of COMP182LinkedList.java. It creates a new instance of the class of the current object and initializes all its fields with exactly the contents of the corresponding fields of this object.

boolean equals(Object o)
This method overrides the equals method (found in the Object class). Since you are overriding the equals method, it must have the same signature as the one found in the Object class! As a result, you will need to cast the Objectparameter to a variable of appropriate type (COMP182LinkedList <Anything>). I suggest that you use the following

line of code as the first line in your solution to accomplish the cast (it will generate warning, but that's OK):

```
COMP182LinkedList <Anything> list = (COMP182LinkedList <Anything>)o;
```

Once you've made the cast, the method should then compare list with this list for equality. The method returns true if and only if both lists have the same size and all corresponding pairs of elements in the two lists are equal. You must not create any other list in this method, and your solution must not modify the original lists.

COMP182LinkedList <Anything> split()
This method splits the original list in half. The original list will continue to reference the front half of the original list and the method returns a reference to a new list that stores the back half of the original list. If the number of elements is odd, the extra element should remain with the front half of the list.

boolean hasCycle()
This method checks if a given linkedList has cycle in it.

Node reverseList()
This method reverse a given linkedList.

Node oddEvenList()
For a given linkedList this method groups all the nodes with odd indices together followed by the nodes with even indices, and return the reordered list.

Node deleteDuplicates ()
For a given **sorted** linkedList, this method deletes all duplicates such that each element appears only once. Return the linked list sorted as well. * For this method you must add proper testing to ListTest.java

boolean isPalindrome()

For a given linkedList, this method returns true if it is a palindrome or false otherwise.
A palindrome is a sequence that reads the same forward and backward. * For this method you must add proper testing to ListTest.java

Other methods
It is OK to add other methods (helper methods) to complete the methods described above. But inside the same class (same file).