You are required, but not limited, to turn in the following source files:

Assignment10.java (This file does not need to be modified)
LinkedList.java (It needs to be modified)
ListIterator.java (This file does not need to be modified)

Requirements to get full credits in Documentation

1. The assignment number, your name, StudentID, Lecture number, and a class description need to be included at the top of each file/class.
2. A description of each method is also needed.
3. Some additional comments inside of long methods (such as a "main" method) to explain codes that are hard to follow should be written.

You can look at the Java programs in the text book to see how comments are added to programs.

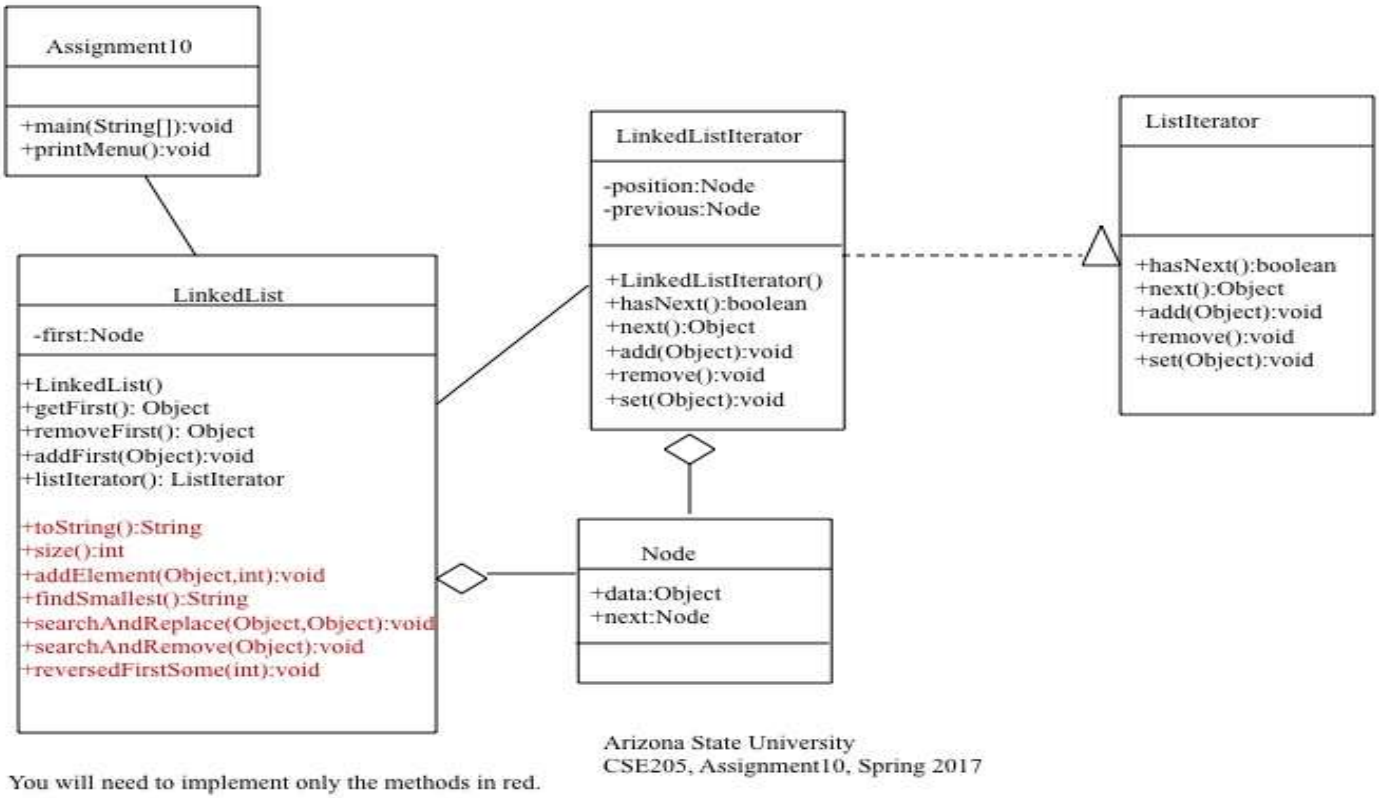
New Skills to be Applied

In addition to what has been covered in previous assignments, the use of the following items, discussed in class, will probably be needed:

Linked Lists

Program Description

Class Diagram:



In the Assignment #10, you are given three files Assignment10.java, LinkedList.java, and ListIterator.java. You will need to add additional methods in the LinkedList class in the LinkedList.java file. The LinkedList will be tested using **strings only.**

Specifically, the following methods must be implemented in the LinkedList class:
(You should utilize listIterator() method already defined in the LinkedList class to obtain its LinkedListIterator object, and use the methods in the LinkedListIterator class to traverse from the first element to the last element of the linked list to define the following methods.)

1.
public String toString()

The toString method should concatenate strings in the linked list, and return a string of the following format:

{ Apple Banana Melon Orange }

Thus it starts with "{" and ends with "}", and **there is a space** between strings and "{" or "}". If the list is empty, it returns "{ }" with a space in between. Note that all elements in the linked list will be added in alphabetical order.

2.
public int size()

The size method returns the number of strings that the linked list contains at the time when this method is called.

3.
public void addElement(Object element, int index)

The addElement adds the parameter element at the parameter specified index. The element at the index and any elements at the later indices will be shifted towards the end of the list. If the parameter index is larger or smaller than the existing indices, it should throw an object of the IndexOutOfBoundsException class.

4.
public String findSmallest( )

The findSmallest method should return the string that is smallest lexically among the strings stored in the linked list. It should return null (null pointer) if the linked list is empty.

5.
public void searchAndReplace(Object first, Object second)

The searchAndReplace method should search and replace all strings that match with the first parameter "first" with the second parameter "second". If the linked list does not contain a string that matches with the first parameter, then the linked list content should not change after calling this method.

6.
public void searchAndRemove(Object toBeRemoved)

The searchAndRemove method should search and remove all strings that match with the first parameter "toBeRemoved". If the linked list does not contain a string that matches with the first parameter, then the linked list content should not change after calling this method.

7.
public void reverseFirstSome(int howMany)

The reverseFirstSome method should reverse the number of strings located at the beginning, specified by the parameter integer. For instance, if the parameter "howMany" is 3, then the first 3 strings in the linked list should be reversed.
If the number "howMany" is 0 or less, then the linked list content will not change, and if it is same or more than the size of the linked list, then the entire linked list content will be reversed.

Test your LinkedList class with the given Assignment10.java file.
It is recommended to test boundary cases such as the cases when the linked list is empty, when it contains only one element, when adding/removing at the beginning or at the end of the linked list.

Input

The following files are the test cases that will be used as input for your program (Right-click and use "Save As"):

Test Case #1
Test Case #2
Test Case #3
Test Case #4

Output

The following files are the expected outputs of the corresponding input files from the previous section (Right-click and use "Save As"):

Test Case #1
Test Case #2
Test Case #3
Test Case #4

Error Handling

Your program is expected to be robust enough to pass all test cases.