Assignment 10

Arizona State University - CSE205- Assignment #10

Due Date Friday, April 10th, 5:30pm

Important: This is an individual assignment. Please do not collaborate.

No late assignment will be accepted.

Make sure that you write every line of your code. Using code written by someone else will be considered a violation of the academic integrity and will result in a report to the Dean's office.

It must be submitted on-line (Course website).

Go to "GradeScope" tab on Canvas -> CSE205 -> Assignment10, and upload your files.

Minimal Submitted Files

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

Assignment10.java (https://canvas.asu.edu/courses/44324/files/14414620/download?wrap=1) **(https://canvas.asu.edu/courses/44324/files/14414620/download?wrap=1)** (This file does NOT need to be modified)
LinkedList.java (https://canvas.asu.edu/courses/44324/files/14414621/download?wrap=1) **(https://canvas.asu.edu/courses/44324/files/14414621/download?wrap=1)** (It needs to be modified)
ListIterator.java (https://canvas.asu.edu/courses/44324/files/14414623/download?wrap=1) **(https://canvas.asu.edu/courses/44324/files/14414623/download?wrap=1)** (This file does NOT need to be modified)

Requirements to get full credits in Documentation

1. The assignment number, your name, StudentID, Lecture day/time, 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 methods (especially for a "main" method) to explain code that are hard to follow should be written.

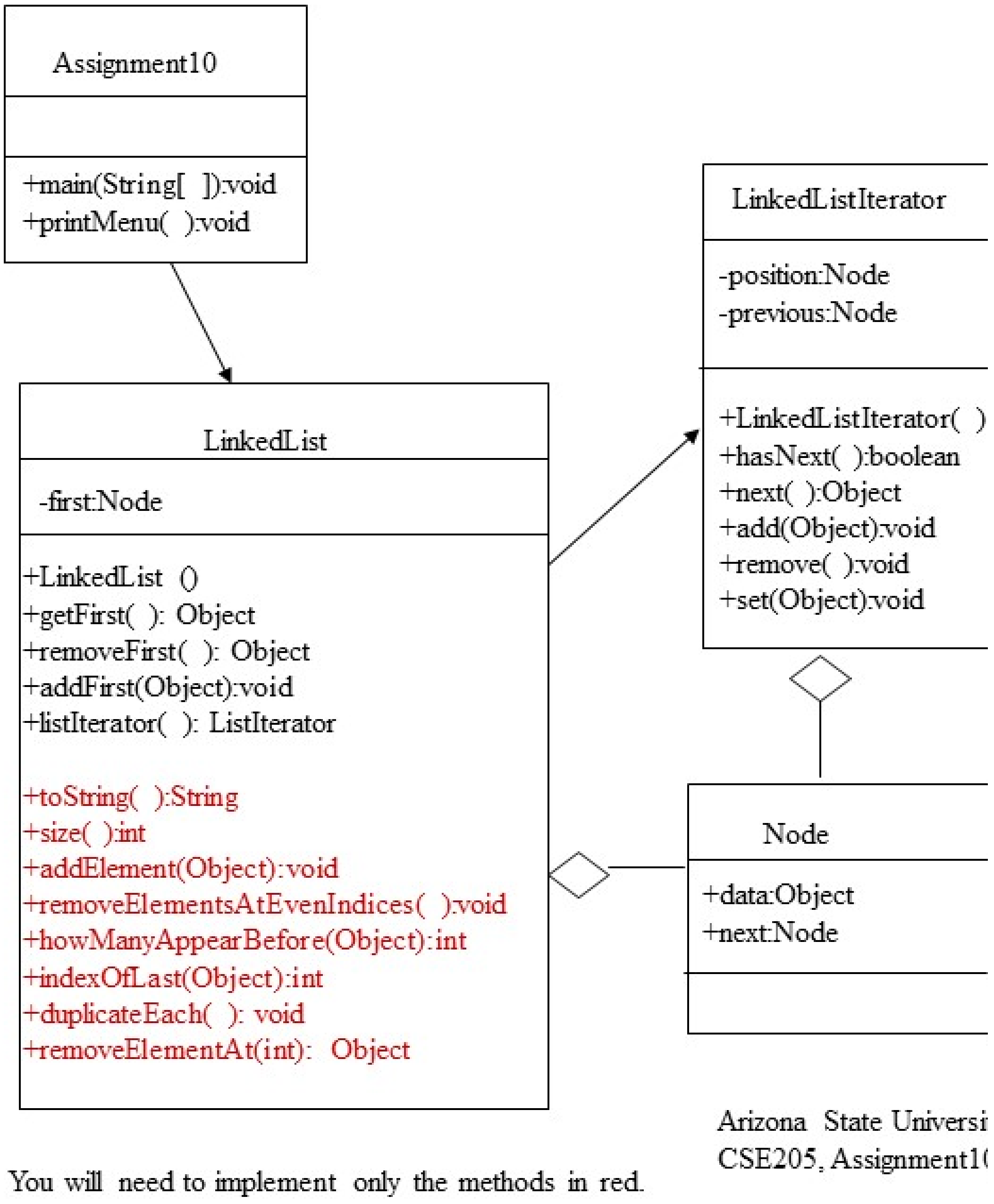
You are not allowed to use the Scanner class in this assignment and any assignment after this one. You will need to use InputStreamReader and BufferedReader (they are in java.io package) to process input and also take care of IOException.

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

Class Diagram:



Program Description

In 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)*

The addElement adds the parameter element into the linked list. The linked list should contain all elements (strings) in alphabetical order. Therefore, in this addElement method, a correct location to insert the parameter element needs to be searched and the element needs to be inserted in that location.

4. *public void removeElementsAtEvenIndices( )*

The removeElementsAtEvenIndices should remove objects at even indices within the linked list. For instance, if the linked list contains { Apple Banana Melon Orange }, then after calling this method, the linked list should contain { Banana Orange } since Applet at the index 0 and Melon at the index 2 need to be removed. Each element within the linked list can be labeled starting at index 0, and based on this assumption, elements should be removed. If the linked list does not contain any element, then the method should not change its content.

5. *public int howManyAppearBefore(Object element)*

The howManyAppearBefore method should search the parameter object (a string in this case) in the linked list, then it should count how many elements appear before the first occurrence of the parameter object/element and return it. If the linked list does not contain the parameter object/element in it, then this method should return -1.

6. *public int indexOfLast(Object element)*

The indexOfLast method should look for the last occurrence of the parameter object (a string in this case) in the linked list, and return the index of its location. If the linked list does not contain the parameter object/elements in it, then this method should return -1.

7. *public void duplicateEach( )*

The duplicateEach method should duplicate each element in the linked list. For instance, if the linked list contains { Apple Banana Melon Orange }, and this method is called, then the linked list will contain { Apple Apple Banana Banana Melon Melon Orange Orange } If the linked list is empty, then it should not change the content of the linked list.

8. *public Object removeElementAt(int)*

The removeElementAt method should look for an object at the parameter index, and it should remove and return the object at the index location. If the parameter index is larger or smaller than the existing indices, it should throw an object of the IndexOutOfBoundsException class.

Test your LinkedList class with the given Assignment10.java file.
It is recommended to test boundary cases such as the cases where 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.

Test Cases

Download the following input files, and the following output files. Save them in the same directory as Assignment10.java is located.

input1.txt (https://canvas.asu.edu/courses/44324/files/14414609/download?wrap=1) (https://canvas.asu.edu/courses/44324/files/14414609/download?wrap=1)

input2.txt (https://canvas.asu.edu/courses/44324/files/14414610/download?wrap=1) (https://canvas.asu.edu/courses/44324/files/14414610/download?wrap=1)

input3.txt (https://canvas.asu.edu/courses/44324/files/14414611/download?wrap=1) (https://canvas.asu.edu/courses/44324/files/14414611/download?wrap=1)

input4.txt (https://canvas.asu.edu/courses/44324/files/14414613/download?wrap=1) (https://canvas.asu.edu/courses/44324/files/14414613/download?wrap=1)

output1.txt (https://canvas.asu.edu/courses/44324/files/14414614/download?wrap=1) **(https://canvas.asu.edu/courses/44324/files/14414614/download?wrap=1)**

output2.txt (https://canvas.asu.edu/courses/44324/files/14414615/download?wrap=1) **(https://canvas.asu.edu/courses/44324/files/14414615/download?wrap=1)**

output3.txt (https://canvas.asu.edu/courses/44324/files/14414616/download?wrap=1) **(https://canvas.asu.edu/courses/44324/files/14414616/download?wrap=1)**

output4.txt (https://canvas.asu.edu/courses/44324/files/14414618/download?wrap=1) **(https://canvas.asu.edu/courses/44324/files/14414618/download?wrap=1)**

Error Handling

Your program should be robust enough to handle all test cases above.

What to turn in:

-Submit your Assignment10.java, LinkedList.java, and ListIterator.java files

using Gradescope-> Assignment10. Make sure that your files are compiling and passing all test cases. You can submit multiple times until the assignment deadline.

Grading Criteria:

____/ 5 Documentation (Each class file needs to have a header with your name, your information, and program description, each method needs its description and comments within your code)

____/ 1 Indentation and spacing (easy to read)

____/ 6 Required classes/methods and functionalities implemented

____/ 8 Produces correct results (test cases – auto graded)

Total points: 20
