CSCI 2110 Data Structures and Algorithms
Laboratory No. 5
Week of October 16-20, 2023

Due: Sunday, October 22, 11.59 PM

Linked Lists

The objective of this lab is to familiarize you with Linked Lists. First download the generic Node class (Node.java) and the generic LinkedList class (LinkedList.java) and study the methods. You will then develop a demo program for linked list manipulation.

Marking Scheme
Each exercise carries 10 points.
Your final score will be scaled down to a value out of 10. For example, if there are three exercises and you score 9/10, 10/10 and 8/10 on the three exercises, your total is 27/30 or 9/10.
Error checking: Unless otherwise specified, you may assume that the user enters the correct data types and the correct number of input entries, that is, you need not check for errors on input

Submission Requirements:

- No submission other than a single ZIP file will be accepted.
- You MUST SUBMIT .java files that are readable by the TA. If you submit files that are unreadable such as .class file, the lab will be marked 0.
- Please additionally comment out package specifiers.

What to submit:

- One ZIP file containing all source code (files with .java suffixes) and a text/word document with sample inputs and outputs, and graphs. The complete list of submission items is given at the end of this document.

You MUST SUBMIT .java files that are executable by your TAs. If you submit files that are unreadable such as .class, you will lose points. Please additionally comment out package specifiers.

Late Submission Penalty: The lab is due on Sunday at 11.59 PM. Late submissions up to 5 hours (4.59 AM on Monday) will be accepted without penalty. After that, there will be a 10% late penalty per day on the mark obtained. For example, if you submit the lab on Monday at 12 noon and your score is 8/10, it will be reduced to 7.2/10. Submissions past two days (48 hours) after the grace submission time, that is, submission past 4.59 AM Wednesday will not be accepted.

Exercise

There is only one exercise for this lab. It requires you to write a demo program with a main method that creates and manipulates a linked list of Strings. You will be required to add methods to the demo program. If required, you may also add methods to the LinkedList class.

1. The program should first prompt the user to enter a list of Pokemon names (Strings), create a linked list with the Pokemon names and display the list, as shown in the example below:

```
Enter Pokemon names, one on each line
Enter quit to end
Pikachu
Charizard
Mewtwo
Emboar
Bulbasaur
Eevee
Gengar
Ditto
Charmander
quit

Here is the linked list:
  Charmander-->Ditto-->Gengar-->Eevee-->Bulbasaur-->Emboar-->Mewtwo-->Charizard-->Pikachu-->
```

2. Next add a method to the demo program that accepts a linked list of Strings as the argument and displays the names in the nodes in even-numbered indices in the list. Assume that the first node is at index 0. For example, with the above list, it should display

```
Here is the linked list with even-numbered indices:
Charmander   Gengar          Bulbasaur     Mewtwo          Pikachu
```

Note that your method should work for a linked list of any size.

3. Next add a method to the demo program that that accepts a linked list of Strings as the argument and displays the names in the nodes in odd-numbered indices in the list. Assume that the first node is at index 0. For example, with the above list, it should display

```
Here is the linked list with odd-numbered indices:
Ditto          Eevee          Emboar          Charizard
```

Note that your method should work for a linked list of any size.

4. Next add a method to the demo program that that accepts a linked list of Strings as the argument and creates and returns another linked list with the names in the reverse order. For example, if the linked list that you created above is the argument, it should create and return the following list:

```
Here is the reversed linked list:
  Pikachu-->Charizard-->Mewtwo-->Emboar-->Bulbasaur-->Eevee-->Gengar-->Ditto-->Charmander-->
```

Note that your method should work for a linked list of any size.

5. Next add a method to the demo program that removes the middle node in the linked list. For example, if the list is

Charmander-->Ditto-->Gengar-->Eevee-->Bulbasaur-->Emboar-->Mewtwo-->Charizard-->Pikachu-->
then it should remove the node at index 4 ("Bulbasaur" - midpoint between 0 and 8) and change the list to
Charmander-->Ditto-->Gengar-->Eevee-->Emboar-->Mewtwo-->Charizard-->Pikachu-->

Suppose the list has an even number of nodes. For example, the list below has 8 nodes,
Charmander-->Ditto-->Gengar-->Eevee-->Emboar-->Mewtwo-->Charizard-->Pikachu-->
then the method should remove the node at index 3 ("Eevee", since 7/2 in integer division is 3), and change the list to
Charmander-->Ditto-->Gengar-->Emboar-->Mewtwo-->Charizard-->Pikachu-->

Your method should work for a linked list of any size.

Test your program by running it for at least three different sets of inputs.

Here's the structure of your solution:

```
import java.util.*;
public class LinkedListDemo{
        public static void main(String[] args){
                //continue
        }
        //method to display even-numbered nodes
        public static void displayEven(LinkedList<String> list){
        //continue
        }
        //method to display odd-numbered nodes
        public static void displayOdd(LinkedList<String> list){
        //continue
        }
        //method to create a reversed linked list
        public static LinkedList<String> reverse(LinkedList<String> list){
        //continue
        }

        //method to remove the middle node in the linked list
        public static void removeMiddle(LinkedList<String> list){
        //continue
        }
}
```

What to submit:

1. Node.java
2. LinkedList.java
3. LinkedListDemo.java
4. Document with screen capture of three sample runs of the program.