XJTLU | Computing

CPT108: Data Structures and Algorithms

Semester 2, 2023-24

Laboratory 3: Linked list

1 Purpose

This is an exercise in programming algorithm for list processing. Specifically, you will be required to implement a singly linked list which other programs will then use for purpose that you do have to care about.

2 Background

The linked lists, which was introduced in the lecture, is a widely used data structure with the property that data can be inserted at any point in the list. This list grows and diminishes in size as elements (or items) are added to, or respectively removed from, the list.

3 Tasks: *Sorted linked list*

In this laboratory session, you are required to implement a *sorted linked list*, i.e., a linked list that keeps its elements sorted (either ascending or descending). To keep all elements sorted, instead of applying a sorting algorithm, you are required to insert the element at its correct position so as to keep the list always sorted.

As a starting point, you are being given top-level requirements in the form of a skeleton implementation containing the desired *application programming interface* (API) and certain required details of the implementation approach. Download the package on Learning Mall (LM), and decompress it.

In the package, you can find see two directories: `configuration`, which contains the configuration of Gradle[1] used to build the package and settings of the Integrated Development Environments (IDEs); and `linkedList`, which contains three Java files in the sub-directories `src`▸`main`▸`java`:

`Node.java`	An element inside the `LinkedList`,
`SortedList.java`	Interface of the `SortedLinkedList` class,
`SortedLinkedList.java`	The class you are going to implement,

[1]Gradle: `http://gradle.org`

Below is the code extracted from `Node.java`

```
public class Node {
  private int n;
  private Node next;

  public Node(int n) {
    setN(n);
  }

  // getters and setters are omitted
}
```

As can be seen, there are two variables in the `Node` class, namely: `n`, uses to hold the data; and `next`, uses to store the address of the next element.

In the lab, your teaching assistant (TA) will guide you through the implementation of the functions: `add(int value)`, `get(int ind)`, `first()`, `last()`, and `clear()`, in the `SortedLinkedList.java` class.

Task 1

You are required to complete the implementation for reset of the `SortedLinkedList.java` class. Note that if there is ever any error in the way that your program is used, your program should print the string `ERROR` on standard error, and *show the type of error occurred*.

Before start implementing your code, you should start analysing the problem by identifying:

- Item: What is given?
- Output: What is required?
- Constraints: Under what conditions?
- Abstraction: What information are essential?

ALL must be presented in some form of data that can be processed by computer.

Besides, you also should consider the following questions.

- What are the scenarios that you need to consider when adding a node to the list? and what needs to be updated in each of the scenario?
- Similarly, what are the scenarios that you need to consider when removing a node from the list? and what needs to be updated in each of the scenario?

and

Do not edit the files `SortedList.java` and `Node.java`. The programming assignments are mini-exercises in how multiple programmers are supposed to interact and communicate in the real world; these files are *owned* and *maintained* by the other author(s).

Testing

In addition to this, you will also find two Java classes in the test folder (`src` ▸ `test` ▸ `java`), `NodeTest` and `SortedLinkedListTest`, which contain some test code that you can use to test your implementation.

You will see the test program crash after executing your code, like the one below, if there is any problem with your implementation.

```
./gradlew

> .../teaching-cpt108_2024spring/docs/laboratories/lab03_collections 3/
       ↪ linkedList/src/main/java/xjtlu/cpt108/collections/linkedList/
       ↪ SortedLinkedList.java:3: error: SortedLinkedList is not abstract
       ↪ and does not override abstract method clear() in SortedList
public class SortedLinkedList implements SortedList {
       ^
1 error

FAILURE: Build failed with an exception.

* What went wrong:
Execution failed for task ':compileJava'.
> Compilation failed; see the compiler error output for details.

* Try:
> Run with --info option to get more log output.
> Run with --scan to get full insights.
```

You may want to edit the `NodeTest.java` and `SortedLinkedListTest.java` files if you would like to test some scenarios that have not been covered by the tests.

You may also want to refer to the lecture note when implementing some of the functions.

4 Marking Scheme

You are required to submit the completed `SortedLinkedList.java` file for marking.

#	Item	Weight
1	Class Structure	10
2	`remove(ind)`	10
3	`remove(Node)`	10
4	`contains(int)`	5
5	`remove(Node)`	5
6	`size()`	5
7	`isEmpty()`	5
8	Documentation	10
	Total	60

Final Remark

Note also that documentation is also a critically important part of your software engineering. Your use of Javadoc will be graded.

Your programming style (how clearly and how well you speak a programming language, such as Java, CPP, and Python) is what will be graded. Correct functioning of your program is *necessary* but ***not*** *sufficient*!

You must write your final version of the program on your own. If you worked in study groups, you must also acknowledge your collaborators in the write-up for each problem, whether or not they are classmates. Other cases will be dealt with as plagiarism. Re-read the policy on the course information sheet, and note the University's tougher policy regarding cheating.