Due Thursday, March 17th, at Noon

Tutorial 6

W22 – COMP1006/1406 – Introduction to Computer Science II

Objectives

Practice with Linked Lists and the Java Collections Framework

---

Notes

Exercise 1 is asking you to read through some classes and understand the provided code. It then asks you to implement (override) some methods dealing with linked lists.

Exercise 2 is asking you to run some code that uses a set and list (with provided Java classes) to compare search times. The question also serves to give a short example of using some java classes (JCF) to store data. Nothing needs to be submitted with this question.

Exercise 3 is an optional extra problem for anyone interested. It is optional. Where will you see this later? If you are CS major, COMP2804 makes you prove the result. You may see it in COMP2402. If you are in the security stream, you will talk about the birthday paradox in COMP3109.

QUESTION 1

Recall that a **list** is a collection of ordered items $s_0, s_1, s_2, \ldots, s_{n-1}$. The length of the list is n.

Download the **Node**, **LList**, **LinkedList** and **TestLL** classes. Read through the classes and be sure you understand all of the **Node** class. The LinkedList (abstract) class gives a partial implementation of the list ADT using a (singly) linked list. Your task for this question is to make the **LList** class concrete. That is, you need to implement all the abstract methods that it inherits. Also, you will need to build up the **TestLL** class.

A. implement the **addFront(String s)** and **removeFront()** methods that were discussed in class last week. The first adds a string to the front the current list. The second removes s_0, the first string in the list. In both, the list is modifies and the size must be updated. (Be sure to handle any special cases.)

B. implement the **find(String s)** method:
returns the first index k such that $s_k = s$, returns -1 if s is not in the list

C. implement the **set(int k, String s)** method:
sets s_k to be s. (Look at the get(int) method.)

D. **Extra**: implement the **remove(int k)** method:
removes s_k, the list adjusts itself to be length $n - 1$
Hint: to remove (or add) in a linked list, you need to find the node that comes *before* the node you want to remove (or add).

Add code to the **TestLL** program to test your methods. Be sure that you test the methods you have overridden.

Note: Look at the given code in LinkedList for help with your methods.

QUESTION 2

In the Java Collections Framework (JCF), the **Set** interface defines a collection that contains no duplicate elements. Two useful classes that implement this interface are the **HashSet** and the **TreeSet**. The **HashSet** class implements the Set ADT (an unordered collection of unique items) while the **TreeSet** class implements the Ordered-Set ADT (an ordered collection of unique items). Important methods of all **Sets** in Java include **add()**, **contains()**, **remove()**, **isEmpty()** and **size()**.

https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/Set.html
https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/HashSet.html
https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/TreeSet.html

```
import jave.util.HashSet;
import java.util.TreeSet;
...
Set<Integer> set = new HashSet<Integer>();
Set<Integer> orderedset = new TreeSet<Integer>();
```

Note: All classes in the JCF use generics. To use these classes, you specify what type of data you want to store in the container. The example above is a set of Integer objects. The data type you specify when using generics must be a reference data type; no primitives are allowed (you would just use the associated primitive wrapper class if you wanted a primitive).

Run the **SetORList** program. Rerun the program several times with different values of *size* (see the main method). Try 1000, 10000, 20000, 50000, etc. If you are using the command line to run your code, you will notice a short-cut to running your code with different sizes using command line arguments.

What can we say about using list versus an unordered set vs an ordered set when:

(i) building the collection

(ii) checking if an item is an element of the collection

You do not have to submit anything for this problem.

QUESTION 3 [EXTRA]

What is the probability that there are two people in a room that have the same birthday? How many people do we need in the room so that the probability is 0.5? This is a problem that you will solve in COMP2804 (if you take that course). It is called the *Birthday Paradox* (or birthday problem).

https://en.wikipedia.org/wiki/Birthday_problem

In this problem, you will investigate the birthday problem using random numbers. Write a program, called **BirthdayParadox**. Your program will randomly generate numbers in a set range until you have a number that is repeated twice. Do this for several ranges (one of the ranges should be the entire range of non-negative integers [0,2^31-1], another should be range [0,365] to simulate the days in a year). Your program should output the size of the range, the number of numbers needed to find a repeated one, and that number squared.

How do you do this? Keep all the numbers that you generate in a set. Each time you generate a random number, check if the number is in the set already or not. If the number is in the set, then you have a match (and you are done)! If the number is not in the set, then

add it to the set and generate another random number. Use the HashSet (that you have seen in the previous problem for this).

Are your results consistent with the results of the birthday problem?

What would happen if you used a list (ArrayList for example) instead of a set?