Linear Structures in the Wild

Due date: by the end of the recitation session

Introduction

We have been studying linear structures for a while now. It is time to look at some of the applications that use some of those structures and to look at their implementation in Java API.
We will be looking at the source code of the OpenMRS package https://github.com/openmrs/openmrs-core

Worksheet: **https://goo.gl/GTcHuW**

Part 1 `Deque<E>` interface

Deque is a data structure that provides functionality related to stacks and queues. It is referred to as a double ended queue. In Java `Deque<E>` is an interface that is implemented by multiple classes.
You can find a single source code file in OpenMRS[1] that uses the `Deque<E>` interface. To answer the questions in this part you will need to

- locate the file in OpenMRS that uses the `Deque<E>` interface
- study the Java API documentation for the `Deque<E>` interface: https://docs.oracle.com/javase/8/docs/api/java/util/Deque.html

Questions (answer them in the worksheet):

1. In the source code file in OpenMRS that uses the `Deque<E>` interface, what is the actual type of the object used? Copy the entire line in which the variable of type `Deque` is instantiated.
2. What is the scope of the variable (i.e., local variable, instance variable / data field)?
3. List **all** methods that are called on the deque instance. Use the Java API documentation to provide brief (one sentence) descriptions of each of the methods.
4. Study the description of the `Deque<E>` interface. What are other classes in Java that implement this interface? Look at the description of these classes, pick two different ones and describe in what ways they are similar and in what ways they are different.
5. Look back at the answer you specified for question 1. Write an alternative line that could replace the line in OpenMRS (i.e., use a different class to instantiate the deque) so that no other lines in in the OpenMRS code need to change.

Part 2 `LinkedList<E>` class

LinkedList<E> is a generic class in Java. It is used in the source code of OpenMRS in many places. If you search for *linkedlist* in the GitHub repository, you should get ten results[2]. To answer the questions in this part you will need to

- open some of the source code files in OpenMRS that use a `LinkedList<E>` object
- study the Java API documentation for the `LinkedList<E>` class
- study the Java API implementation (the source code) for the `LinkedList<E>` class (you should have it easily available on your computer at this point, but if you do not, download it from the course website)

[1] Based on the access on Apr. 2, 2017.
[2] Based on the access on Apr. 2, 2017.

Questions (answer them in the worksheet):

1. Find three different line in OpenMRS that instantiate an object of type `LinkedList<E>` (the generic type `E` will be replaced with an actual type in the code). Pick those three lines from three different files. For each state
   - the name of the file from which it came from (and the line number),
   - the type of the reference that points to the `LinkedList<E>` object ,
   - the actual type that is used in place of the generic type.

   In your own words
   - explain why is it possible to have different reference types all point to an instance of type `LinkedList`
   - explain why different actual types can be used to replace the generic type `E`
2. Read the class description for the `LinkedList<E>.` Does the description suggest how the class is actually implemented (i.e., how the data is stored)?
3. Find which package the `LinkedList<E>` class is located in. Explain how you figured it out. Open the source code file for `LinkedList<E>` class.
4. List all data fields used in the `LinkedList<E>` class. Explain their purpose.
5. What is the type of the building blocks (i.e., nodes in which the data is stored) for the `LinkedList<E>` class? Find and copy the source code for the entire class. Explain the meaning of each of the data fields in this class.
6. Find the `boolean add(E e)` method of the `LinkedList<E>` class. Explain how it works. (Note, you may need to look at other methods as well.)
7. Find the `void add(int index, E element)` method of the `LinkedList<E>` class. Explain how it works. (Note, you may need to look at other methods as well.)
8. Find the `boolean contains(Object o)` method of the `LinkedList<E>` class. Explain how it works. (Note, you may need to look at other methods as well.)
9. Find the `boolean remove(Object o)` method of the `LinkedList<E>` class. Explain how it works. (Note, you may need to look at other methods as well.)
10. Find the `boolean remove(Object o)` method of the `LinkedList<E>` class. Explain how it works. (Note, you may need to look at other methods as well.)