BIL 212
Summer 2021

Lab 4

Part 1

Download Tree, AbstractTree, BinaryTree, AbstractBinaryTree and LinkedBinaryTree classes from Piazza. Modify the LinkedBinaryTree as such that it takes only a Comparable class as the template parameter. You will use compareTo method for the comparisons. Then implement a member method with the signature of:

```
public <E> void addOrdered(E element)
```

This method add the element to the tree following these rules:

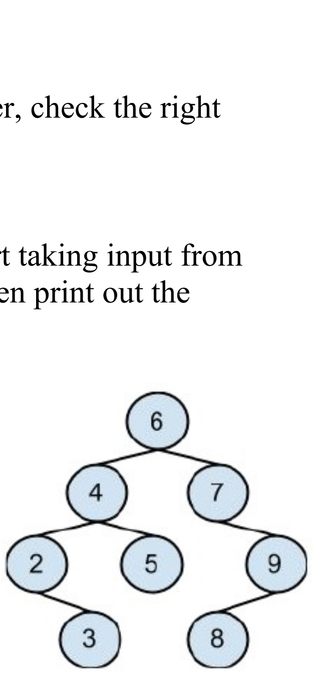
- If the tree is empty, add the element as root
- If not, start looking for the suitable position starting from root
    - If the element is smaller than the current node's value, check the left node
    - If the element is bigger, check the right node
- Continue checking recursively until encountered with an empty node
- Add the element as the leaf node to this position

Starting with an empty integer tree, an example sequence goes like this:

1. Add 6:
    1. Root is empty, add 6 as root node
2. Add 4:
    1. Root contains value 6, compare 4 to 6, it is smaller, check the left child
    2. Left child is empty, add 4 here
3. Add 5:
    1. Compare 5 to 6, it is smaller, check the left child
    2. Left child contains the value 4, compare 5 to 4, it is bigger, check the right child (of 4, not the right child of 6)
    3. Right child is empty, add 5 here

In your LBTDriver class, create an LinkedBinaryTree<Integer>, start taking input from terminal and add them to tree using addOrdered until -1 is given. Then print out the whole tree using the iterator that you implemented in class.

Example input sequence: 6 4 2 7 3 9 8 5 -1   Example tree structure:



Example output: 6 4 2 3 5 7 9 8

Part 2

1. Download List.java and ArrayList.java from Piazza site under Resources. As we covered in class, the inner class ArrayIterator lets us create iterators over elements of the ArrayList. Study this class before you continue. The method iterator() of the ArrayList class creates and returns an Iterator.
2. Now, you will write another iterator class within the ArrayList class called ArrayListIterator. This class should implement the java.util.ListIterator interface. See https://docs.oracle.com/javase/8/docs/api/java/util/ListIterator.html for the method specifications of the ListIterator.
3. Add in your ArrayList class two methods listIterator() and listIterator(int i) that will make an instance of ArrayListIterator and position it at the beginning of the list, and position it right before the i-th index, respectively. Note that indices start with 0.
4. Now, in your ALDriver, create an ArrayList<Integer> objects, called L. Populate this list with random integers, and then call a method addSumBetween(L) that you write. The method addSumBetween(L) is supposed to add the sum of each subsequent pair BETWEEN them. For instance, if L contains [3, 10, 8, 5] originally, after the addSumBetween call L will contain [3, 13, 10, 18, 8, 13, 5]. You can have more than one iterator. You are allowed to use ONLY the methods of the ListIterator in your code. Note that only n-1 values are added to an n sized list, not adding anything to the start and end of the list.

You can write your own code or use the classes posted on Piazza, but you cannot use the Java's built-in data structures. Make sure to check your section before attending. Please upload all your .java files without compression to uzak.etu.edu.tr before 17.06.2021 Thursday 11:00.