Practical Exercise 7 – Binary Search Trees

Overall Objective

To design and implement applications using binary search trees.

Background

You will need to know:

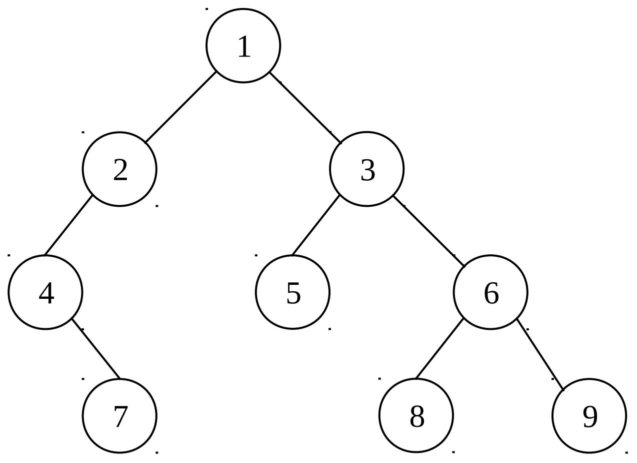
1. basic Java programming knowledge
2. classes and interfaces
3. generics
4. recursion
5. binary search tree concept

Description

Part 1: Discussion

1. For the Binary Tree given below, write the numbers in the order by using the following traversal methods:

    a. Preorder (C**LR)**中左右
    b. Inorder (**LCR)**左中右
    c. Postorder (**LRC)** 左右中



2. Suppose you have a binary tree whose data fields are single characters.

   a. When the data fields of nodes are printed in in-order, the output is ABCDEFGHIJ, and when they are printed in pre-order, the output is BAHCEDGFJI. Draw the binary tree showing the data in each node and the references between nodes. Show the step used to arrive at the result.

Pre-order = Center Left Right (CLR)
In-order = Left Center Right (LCR)

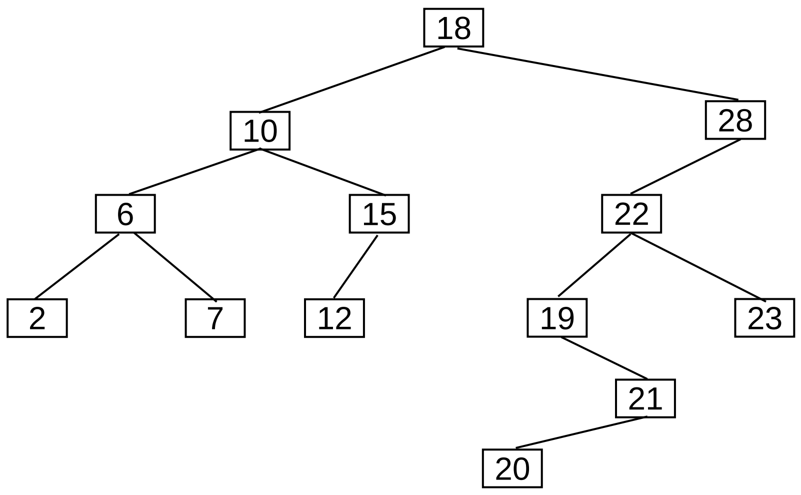
Steps:

1. **In pre-order is BAHCEDGFJI**
2. **In in-order is ABCDEFGHIJ**
3. **B is the centre root node (From 1)**
4. **A is the left node of B (From 2, 3)**
5. **CDEFGHIJ (in-order) is the right part of B (From 2, 3, 4)**
6. **HCEDGFJI (pre-order) is the right part of B (From 1, 3, 4, 5)**
7. **H is the right node of B (From 6)**
8. **CDEFG (in-order) is the left part of H (From 5, 7)**
9. **CEDGF (pre-order) is the left part of H (From 6, 7, 8)**
10. **C is the left node of H (From 9)**
11. **DEFG(in-order) is the right part of C (From 8, 10)**
12. **EDGF (pre-order) is the right part of C (From 9, 11)**
13. **E is the right node of C (From 12)**
14. **D is the left node of E (From 11, 13)**
15. **FG (in-order) is the right part of E (From 11, 13)**
16. **GF (pre-order) is the right part of E (From 12, 15)**
17. **G is the right node of E (From 15)**
18. **F is the left node of G (From 15, 17)**
19. **IJ (in-order) is the right part of H (From 5, 7)**
20. **JI (pre-order) is the right part of H (From 6, 19)**
21. **J is the right node of H (From 20)**
22. **I is the left node of J (From 19, 21)**

Answer:
See 3, 7, 10, 13, 14, 17, 18, 21, 22 to construct the full binary tree.

b. When the data fields of nodes are printed in in-order, the output is ABCDEFGHIJ, and when they are printed in post-order, the output is BEDGFHCJIA. Draw the binary tree showing the data in each node and the references between the nodes. Show the steps used to arrive at the result.

3. Below is a Binary Search Tree (BST). What is the tree obtained after each of the following operations (each on the initial tree)?



a.	`insert(31);`	e.	`remove(15);`
b.	`insert(4);`	f.	`remove(28);`
c.	`insert(16);`	g.	`remove(6);`
d.	`remove(23);`	h.	`remove(18);`

Part 2: Programming Exercise

Refer to lecture slide/main textbook (Ch27, Liang 9th edi.), define interface, abstract class and concrete class for Binary Search Tree (BST) as follows:

1. Define `Tree` interface that extends `java.lang.Iterable` (refer to slide 35).

2. Define `AbstractTree` abstract class that implements `Tree` interface (refer to slide 35).

3. Define `BinaryTree` concrete class that extends `AbstractTree` abstract class (refer to slide 36).
   - Define an inner class `TreeNode`
   - Define an inner class `InorderIterator` that implements `java.util.Iterator`

4. **Find the leaves**
   Add a method in `BinaryTree` class to return the number of the leaves as follows:
   ```
   /** Returns the number of leaf nodes */
   public int getNumberOfLeaves()
   ```

5. Write the test program that puts 20 random integers between -100 and 100 into a BST. The program should print out all the integers in the BST and test the `getNumberOfLeaves()` method above.
   [Note that in Java, there is a method `Math.random()`, which returns a double value between 0.0 and 1.0. And there is another method `Random.nextInt(int n)`, which returns a random value in the range of 0 (inclusive) and n (exclusive).]

6. In addition to the test program you have constructed for Question 5, write and test a method that sums all the integers in the BST.