GIT Department of Computer Engineering
CSE 222/505 - Spring 2022
Homework #07 Report

Muhammed Sefa Cahyir
1801042686

1. SYSTEM REQUIREMENTS

No need expensive system for this project. If don't use with big datas.

It is enough to have a Java and a machine to run JVM

The minimum system requirements for Java Virtual Machine are as follows:

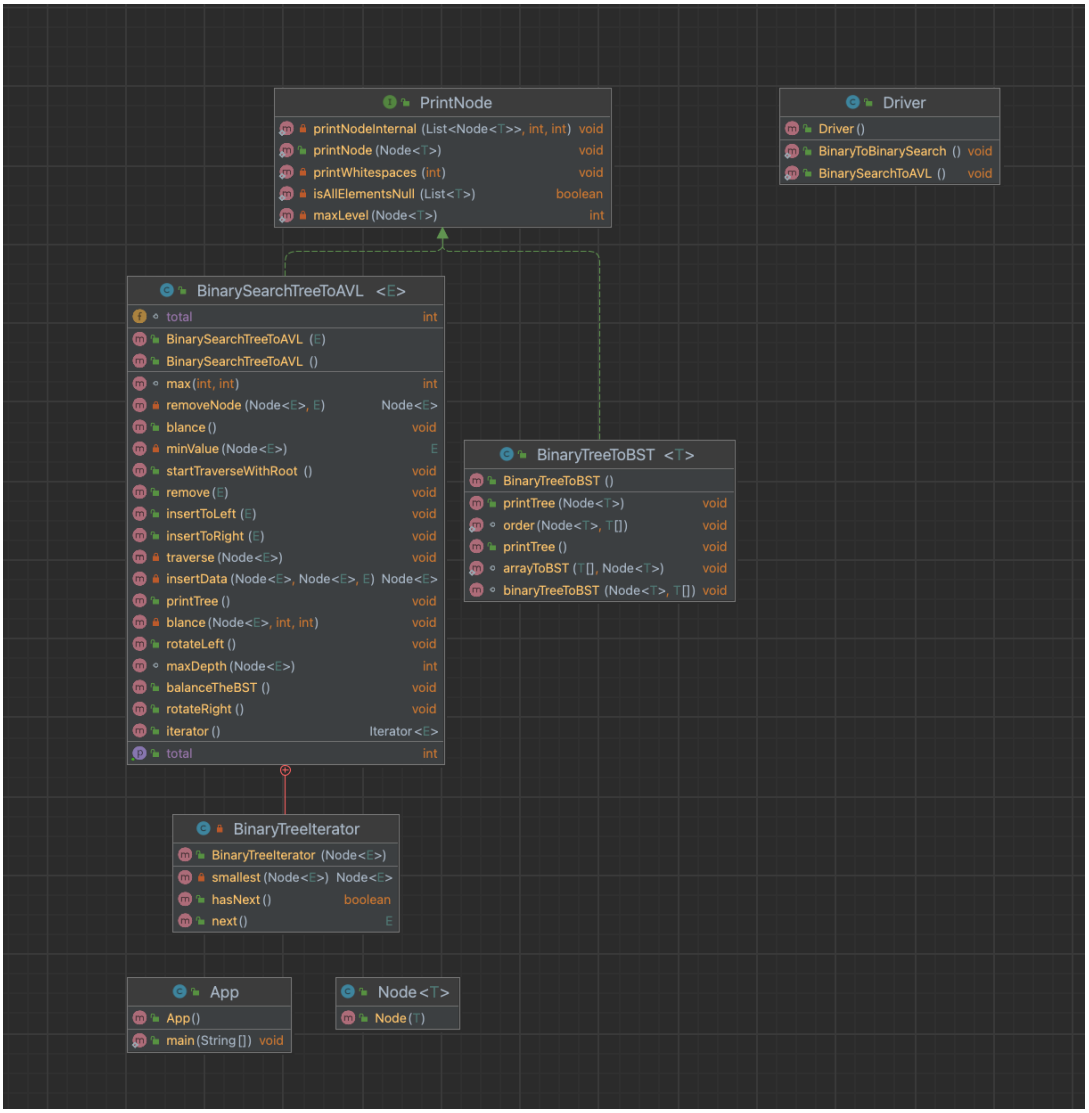
Windows 8/7/Vista/XP/2000.

Windows Server 2008/2003.

Intel and 100% compatible processors are supported.

Pentium 166 MHz or faster processor with at least 64 MB of physical RAM. 98 MB of free disk space.

2. USE CASE AND CLASS DIAGRAMS



3. OTHER DIAGRAMS

No other diagram

4. PROBLEM SOLUTION APPROACH

In this homework Q1 problem was the convert the Binary Tree to Binary Search Tree without change the shape, for this problem I was the find the left and right sides and after that find the root node. After the find the root node implement others one by one. And this is create a new tree without change the shape.

```
void binaryTreeToBST(Node<T> root, T arr[]) {
   order(root, arr);

   Arrays.sort(arr);

   arrayToBST(arr, root);
}
```

Is the main method (deleted the O(1) lines, just methods is left) , and in there
Order(root,arr) is the -> O(log n)
Array.sort(arr) is using the quicksort and -> O(n log n)
arrayTheBST (arr,root) is the -> O(log n)

In the homework Q2 problem was the convert the BST to AVL tree, to making this we need to make this balance method first. For balance method we need to rotateRight and rotateLeft method. When write all these method we need to find the depth between left and right trees. For this created a method with name maxDepth. This method is finding the left and right side depth, and with these we can find the differences between left and right tree depth. If left and right depth difference is more then one we are using rotateLeft or rotateRight until difference lower or equal than one.

```
private void blance(Node<E> root, int rootLeftSize, int rootRightSize) {
    if ((rootLeftSize - rootRightSize) > 1) {
        rotateRight();
        blance(root, rootLeftSize - 1, rootRightSize + 1);
    }
    else if ((rootLeftSize - rootRightSize) < -1) {
        rotateLeft();
        blance(root, rootLeftSize + 1, rootRightSize - 1);
    }
}
```

Is the main method (deleted the O(1) lines, just methods is left) , and in there Blance using rotate right or rotate left and using maxDepth to find depth

```
blance(root, rootLeftSize - 1, rootRightSize + 1);
        rotateLeft, rotateRight -> O(log n)
        maxDepth -> O(log n)
```

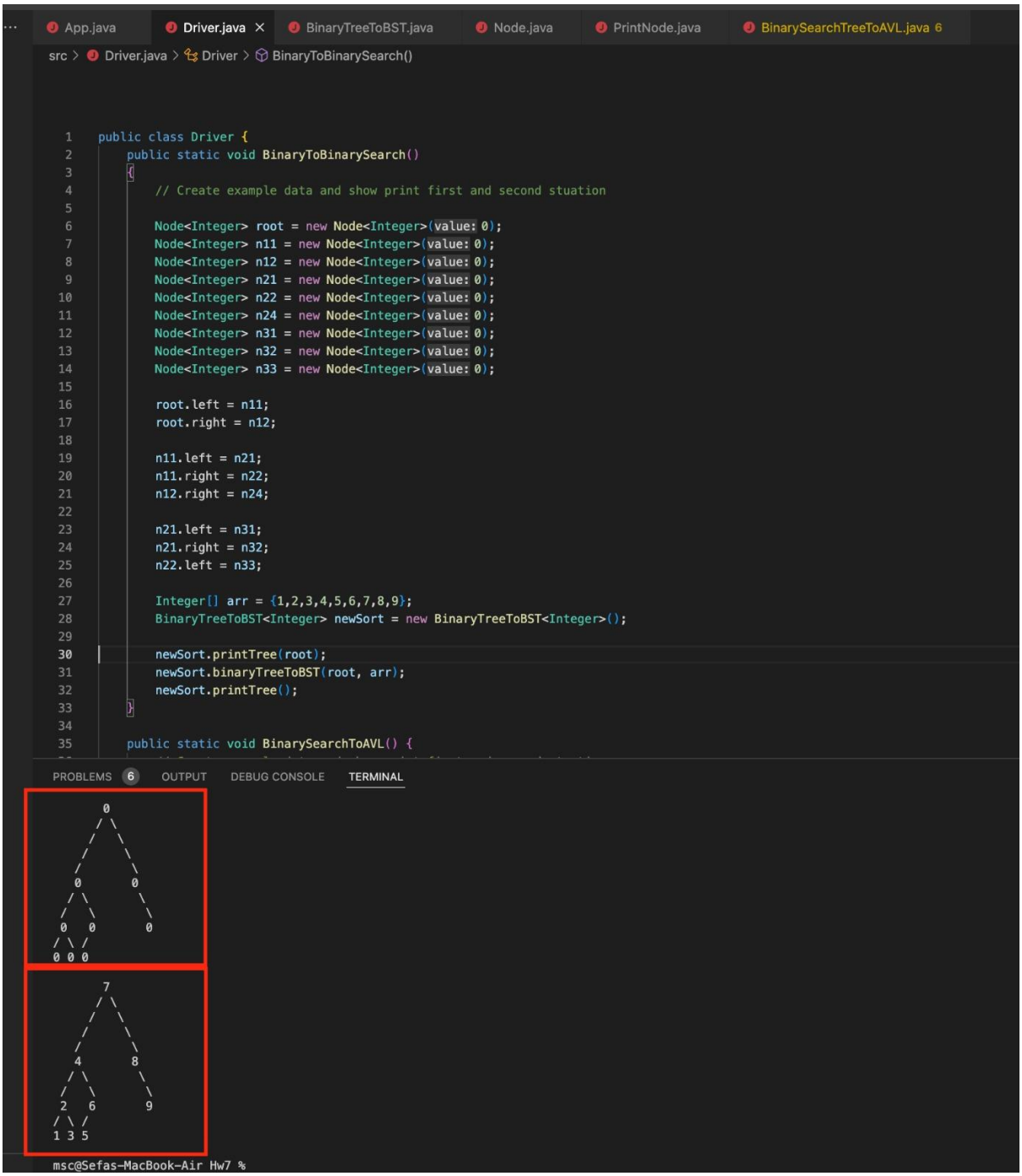
5. **TEST CASES**

Q1 is tested with diffirent variations like sorted array, unsorted array , balanced tree, unbalanced tree, and showed the example in downside.

Q2 is tested with diffirent variations like sorted array, unsorted array , balanced tree, unbalanced tree, and showed the example in downside.

6. RUNNING AND RESULTS

For question 1 example and result



For question 2 example and result

```java
    }

    public static void BinarySearchToAVL() {
        // Create example data and show print first and second stuation

        BinarySearchTreeToAVL<Integer> bst = new BinarySearchTreeToAVL<Integer>();

        Node<Integer> root = new Node<Integer>(value: 20);
        Node<Integer> n11 = new Node<Integer>(value: 10);
        Node<Integer> n12 = new Node<Integer>(value: 40);
        Node<Integer> n21 = new Node<Integer>(value: 5);
        Node<Integer> n22 = new Node<Integer>(value: 15);
        Node<Integer> n32 = new Node<Integer>(value: 7);

        bst.root = root;
        bst.root.left = n11;

        n11.left = n21;
        n11.right = n22;

        n21.right = n32;
        n22.left = n12;

        bst.printTree();
        bst.balanceTheBST();
        bst.printTree();
    }
}
```

PROBLEMS 6 OUTPUT DEBUG CONSOLE TERMINAL

```
      20
     /
    /
   /
  /
 10
/ \
5  15
 \ /
 7 40
```

```
      10
     / \
    /   \
   /     \
  /       \
 5        20
  \      /
   \    /
   7   15
      /
     40
```

```
msc@Sefas-MacBook-Air Hw7 %
```