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

Ali Kaya

1901042618

a.kaya2019@gtu.edu.tr

1.Detailed System Requirements

```
public static <E extends Comparable<E>> BinarySearchTree<E> BSTBuilder(BinaryTree<E> BT , E[] arr){
```

Requirements for *BSTBuilder method* :

* parameter BT , a BinaryTree parameter

* parameter arr , an Array parameter

* parameter <E> this is a generic parameter which extends Comparable<E> interface.

Requirements for *converter method* :

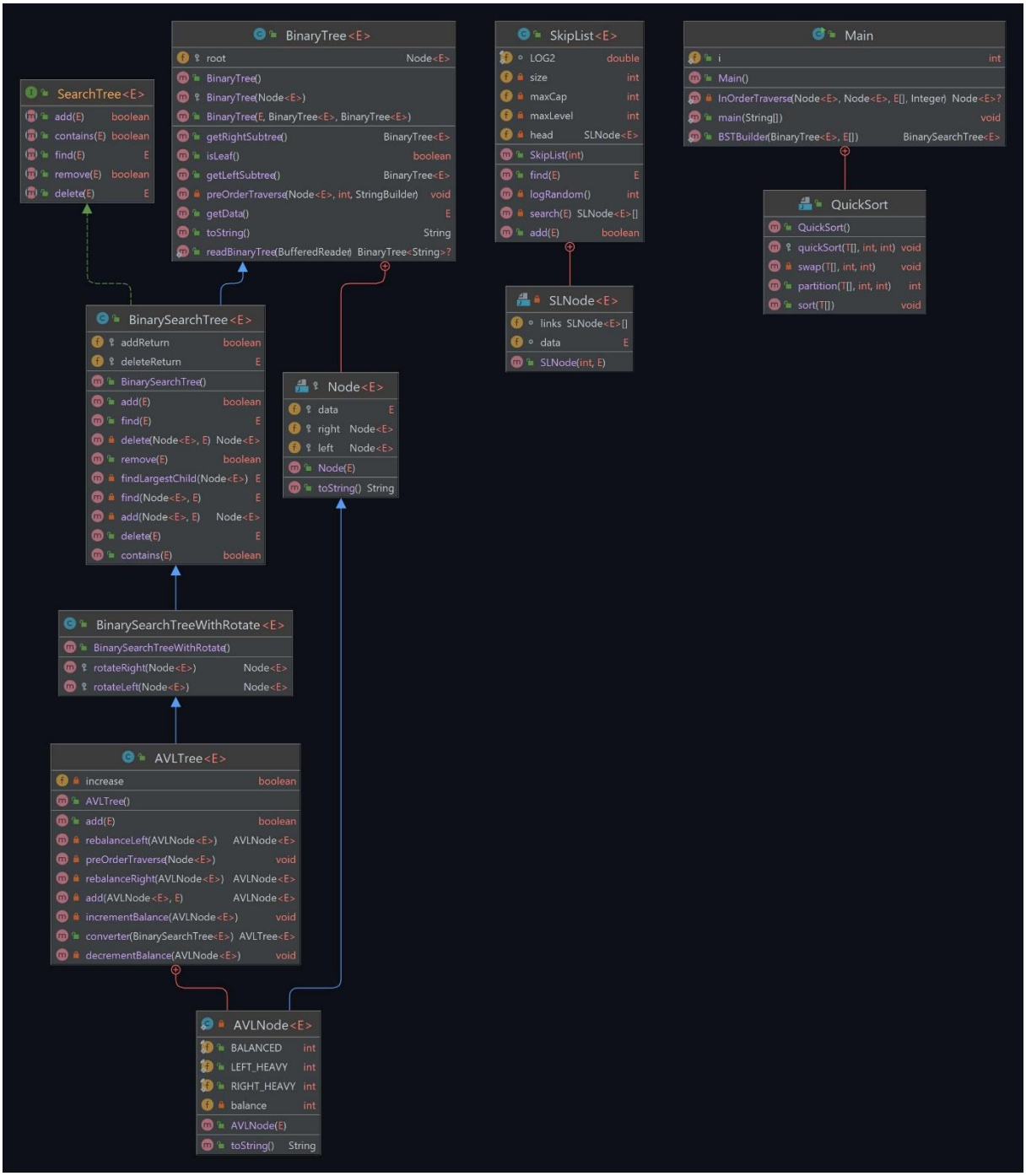
* takes a binary search tree (BST) as a parameter

* parameter <E> this is a generic parameter which extends Comparable<E> interface.

To use above two method properly user must give Comparable type.

(E extends Comparable<E>)

2. Class Diagrams



3. Problem solutions approach

#FOR *BSTBuilder method* :

This method takes a binary tree and an array of items as input , and it returns a binary search tree (BST) as output. Also this method builds a binary search tree of n nodes.

By making this , structure of the binary search tree going to be same as the structure of the given binary tree for example:

```
Let's create a BinaryTree<E> reference
Then let's add some nodes(0) in it manually
Let's print it with toString() method

0
    0
        0
            null
            null
        0
            0
                null
                null
            0
                null
                null
    0
        null
        0
            0
                null
                null
            null
```

Let our initial binary tree structure be like this ,

Then we create an array which includes some integer data,then we call BSTBuilder method create a Binary search tree which has same structure as given BinaryTree structure(above part)

```
Let's create a Integer array with some integer in it
Integer[] myArr = {8 , 3 , 5 , 7 , 2 , 1 , 4 , 6 , 9}
Let's create a BinarySearchTree<E> reference and call
Let's print binary search tree with toString() method

6
    2
        1
            null
            null
        4
            3
                null
                null
            5
                null
                null
    7
        null
        9
            8
                null
                null
            null
```

To make these first our array must be sorted.

Then, while traversing binary search tree, we can add the elements of the array one by one to bst, starting from the smallest node.

We use InOrderTraverse to traverse the trees.

****Theoretical run time complexity of BSTBuilder method=>*

```
public static <E extends Comparable<E>> BinarySearchTree<E> BSTBuilder(BinaryTree<E> BT
    i = 0;
    QuickSort testQuick = new QuickSort();
    testQuick.sort(arr);                                    O(nlogn)

    BinarySearchTree<E> result = new BinarySearchTree<>();
    result.root = new BinaryTree.Node<E>( data: null);
    result.root = InOrderTraverse(result.root , BT.root , arr , i);   theta(N)
    return result;
}
```

O(nlogn)

Traverses always theta(N) because it traverse all nodes in the tree

#FOR *converter method* :

This method simply converts a binary search tree to an AVLTree. For this, I use preOrderTraverse to traverse all the elements in bst one by one and transfer them to AVL tree. AVL tree is already doing balance operations by itself.

(balance operations => rotation operations)

****Theoretical run time complexity of converter method=>*

```
public AVLTree<E> converter(BinarySearchTree<E> BST){
    preOrderTraverse(BST.root);   theta(N)
    return this;
}
```

theta(N)

Traverses always theta(N) because it traverse all nodes in the tree

4)Test Cases

```
System.out.println("Let's create a BinaryTree<E> reference");
BinaryTree<Integer> myBt = new BinaryTree<Integer>();
System.out.println("Then let's add some nodes(0) in it manually");

myBt.root = new BinaryTree.Node<>( data: 0);
myBt.root.left = new BinaryTree.Node<>( data: 0);
myBt.root.left.left = new BinaryTree.Node<>( data: 0);
myBt.root.left.right = new BinaryTree.Node<>( data: 0);
myBt.root.left.right.left = new BinaryTree.Node<>( data: 0);
myBt.root.left.right.right = new BinaryTree.Node<>( data: 0);
myBt.root.right = new BinaryTree.Node<>( data: 0);
myBt.root.right.right = new BinaryTree.Node<>( data: 0);
myBt.root.right.right.left = new BinaryTree.Node<>( data: 0);

System.out.println("Let's print it with toString() method\n");
System.out.println(myBt.toString());

System.out.println("Let's create a Integer array with some integer
Integer[] myArr = {8 , 3 , 5 , 7 , 2 , 1 , 4 , 6 , 9};

System.out.println("Let's create a BinarySearchTree<E> reference an
BinarySearchTree<Integer> bst ;
bst = BSTBuilder(myBt , myArr);
```

```
0
    0
        0
            null
            null
        0
            0
                null
                null
            0
                null
                null
    0
        null
        0
            0
                null
                null
            null
```



```
6
    2
        1
            null
            null
        4
            3
                null
                null
            5
                null
                null
    7
        null
        9
            8
                null
                null
            null
```

```java
System.out.println("Let's create another BinaryTree<E> reference");
BinaryTree<Integer> myBt2 = new BinaryTree<Integer>();
System.out.println("Then let's add some nodes(0) in it manually");

myBt2.root = new BinaryTree.Node<>( data: 0);
myBt2.root.left = new BinaryTree.Node<>( data: 0);
myBt2.root.left.left = new BinaryTree.Node<>( data: 0);
myBt2.root.left.right = new BinaryTree.Node<>( data: 0);
myBt2.root.right = new BinaryTree.Node<>( data: 0);
myBt2.root.right.left = new BinaryTree.Node<>( data: 0);
myBt2.root.right.right = new BinaryTree.Node<>( data: 0);

System.out.println("Let's print it with toString() method\n");
System.out.println(myBt2.toString());

System.out.println("Let's create a Integer array with some integer ir
Integer[] myArr2 = {75 , 35 , 55 , 65 , 25 , 15 , 45};

System.out.println("Let's create another BinarySearchTree<E> referenc
BinarySearchTree<Integer> bst2 ;
bst2 = BSTBuilder(myBt2 , myArr2);
```

```
0
    0
        0
            null
            null
        0
            null
            null
    0
        0
            null
            null
        0
            null
            null
```



```
45
    25
        15
            null
            null
        35
            null
            null
    65
        55
            null
            null
        75
            null
            null
```

```java
System.out.println("Then let's add some data in it with add(E item) method");
myBst.add(0);
myBst.add(1);
myBst.add(2);
myBst.add(3);
myBst.add(4);
myBst.add(5);
myBst.add(6);

System.out.println("Let's print it with toString() method\n");
System.out.println(myBst.toString());

System.out.println("Let's create a AVLTree<E> reference and call converter(myB
BinaryTree<Integer> myAvl = new AVLTree<Integer>();
myAvl = ((AVLTree<Integer>) myAvl).converter(myBst);
System.out.println(myAvl.toString());
```

```
0
    null
    1
        null
        2
            null
            3
                null
                4
                    null
                    5
                        null
                        6
                            null
                            null
```



```
0: 3
    0: 1
        0: 0
            null
            null
        0: 2
            null
            null
    0: 5
        0: 4
            null
            null
        0: 6
            null
            null
```

5) Running command and results

```
System.out.println("Let's create a BinaryTree<E> reference");
BinaryTree<Integer> myBt = new BinaryTree<Integer>();
System.out.println("Then let's add some nodes(0) in it manually");

myBt.root = new BinaryTree.Node<>( data: 0);
myBt.root.left = new BinaryTree.Node<>( data: 0);
myBt.root.left.left = new BinaryTree.Node<>( data: 0);
myBt.root.left.right = new BinaryTree.Node<>( data: 0);
myBt.root.left.right.left = new BinaryTree.Node<>( data: 0);
myBt.root.left.right.right = new BinaryTree.Node<>( data: 0);
myBt.root.right = new BinaryTree.Node<>( data: 0);
myBt.root.right.right = new BinaryTree.Node<>( data: 0);
myBt.root.right.right.left = new BinaryTree.Node<>( data: 0);

System.out.println("Let's print it with toString() method\n");
System.out.println(myBt.toString());
```

```
0
    0
        0
            null
            null
        0
            0
                null
                null
            0
                null
                null
    0
        null
        0
            0
                null
                null
            null
```

```java
Integer[] myArr = {8 , 3 , 5 , 7 , 2 , 1 , 4 , 6 , 9};

System.out.println("Let's create a BinarySearchTree<E> reference and call BSTBuilde
BinarySearchTree<Integer> bst ;
bst = BSTBuilder(myBt , myArr);

System.out.println("Let's print binary search tree with toString() method\n");
System.out.println(bst.toString());
```

```
Let's create a Integer array with some integer in it
Integer[] myArr = {8 , 3 , 5 , 7 , 2 , 1 , 4 , 6 , 9}
Let's create a BinarySearchTree<E> reference and call B
Let's print binary search tree with toString() method

6
    2
        1
            null
            null
        4
            3
                null
                null
            5
                null
                null
    7
        null
        9
            8
                null
                null
            null
```

```java
System.out.println("Let's create another BinaryTree<E> reference");
BinaryTree<Integer> myBt2 = new BinaryTree<Integer>();
System.out.println("Then let's add some nodes(0) in it manually");

myBt2.root = new BinaryTree.Node<>( data: 0);
myBt2.root.left = new BinaryTree.Node<>( data: 0);
myBt2.root.left.left = new BinaryTree.Node<>( data: 0);
myBt2.root.left.right = new BinaryTree.Node<>( data: 0);
myBt2.root.right = new BinaryTree.Node<>( data: 0);
myBt2.root.right.left = new BinaryTree.Node<>( data: 0);
myBt2.root.right.right = new BinaryTree.Node<>( data: 0);

System.out.println("Let's print it with toString() method\n");
System.out.println(myBt2.toString());
```

```
Let's create another BinaryTree<E> reference
Then let's add some nodes(0) in it manually
Let's print it with toString() method

0
    0
        0
            null
            null
        0
            null
            null
    0
        0
            null
            null
        0
            null
            null
```

```java
System.out.println("Let's create a Integer array with some integer in it\nInteger[]
Integer[] myArr2 = {75 , 35 , 55 , 65 , 25 , 15 , 45};

System.out.println("Let's create another BinarySearchTree<E> reference and call BSTB
BinarySearchTree<Integer> bst2 ;
bst2 = BSTBuilder(myBt2 , myArr2);

System.out.println("Let's print binary search tree with toString() method\n");
System.out.println(bst2.toString());
```

```
Let's create a Integer array with some integer in it
Integer[] myArr2 = {75 , 35 , 55 , 65 , 25 , 15 , 45}
Let's create another BinarySearchTree<E> reference and
Let's print binary search tree with toString() method

45
    25
        15
            null
            null
        35
            null
            null
    65
        55
            null
            null
        75
            null
            null
```

```java
System.out.println("Let's create a AVLTree<E> reference");
BinarySearchTree<Integer> myBst = new BinarySearchTree<Integer>();

System.out.println("Then let's add some data in it with add(E item) method");
myBst.add(0);
myBst.add(1);
myBst.add(2);
myBst.add(3);
myBst.add(4);
myBst.add(5);
myBst.add(6);

System.out.println("Let's print it with toString() method\n");
System.out.println(myBst.toString());
```

```
Let's create a AVLTree<E> reference
Then let's add some data in it with add(E item) method
Let's print it with toString() method

0
    null
    1
        null
        2
            null
            3
                null
                4
                    null
                    5
                        null
                        6
                            null
                            null
```

```java
System.out.println("Let's create a AVLTree<E> reference and call converter(myBst)
BinaryTree<Integer> myAvl = new AVLTree<Integer>();
myAvl = ((AVLTree<Integer>) myAvl).converter(myBst);
System.out.println(myAvl.toString());
```

```
Let's create a AVLTree<E> reference and call converter(myBst)
0: 3
    0: 1
        0: 0
            null
            null
        0: 2
            null
            null
    0: 5
        0: 4
            null
            null
        0: 6
            null
            null
```