# CSE 222/CSE505 SPRING 2022

# HOMEWORK 7 REPORT

Burcu Sultan Orhan

1901042667

# 1- Detailed System Requirements

```
BinaryTree<String> subTree6 = new BinaryTree<>("sdf", null, null);
BinaryTree<String> subTree5 = new BinaryTree<>("ghhgf", null, null);
BinaryTree<String> subTree4 = new BinaryTree<>("xyz", null, null);
BinaryTree<String> subTree3 = new BinaryTree<>("hgfss", null,  null);
BinaryTree<String> subTree2 = new BinaryTree<>("hgfv", subTree5, subTree6);
BinaryTree<String> subTree1 = new BinaryTree<>("asdfgf", subTree3, subTree4);
BinaryTree<String> myTree = new BinaryTree<>("fggfd", subTree1, subTree2);


String[] inputs = {"burcu", "sultan", "orhan", "gizem", "alper", "elif", "melissa"};
```

First there needs to be a BinaryTree class to implement binary tree and rearrange it.

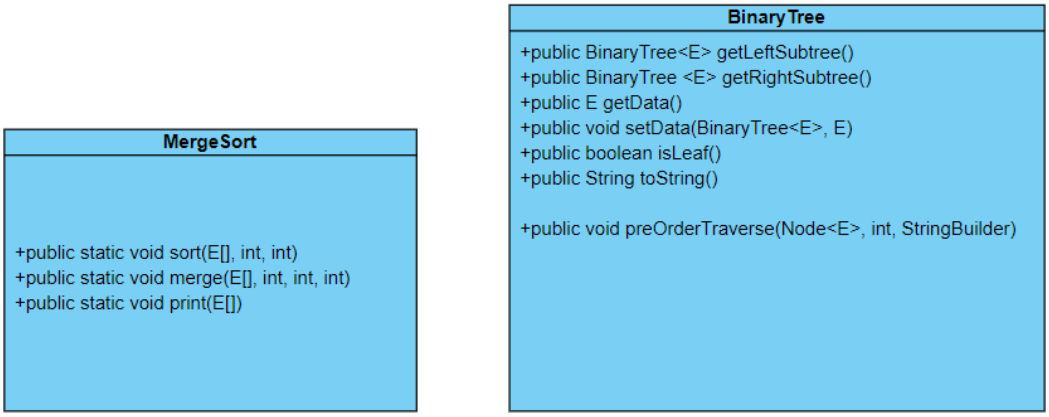
```
public static void binaryToBinarySearch(BinaryTree<String> localTree, String[] data){

    System.out.println("Given binary tree structure:");
    System.out.println(localTree.toString());
    System.out.print("Given array of inputs: ");
    for(int i=0; i<data.length; i++){
        System.out.print(data[i] + " ");
    }
    MergeSort.sort(data, 0, data.length-1);
    fillTree(localTree, data, 0, data.length-1);
    System.out.println("\n\n\nBinary search tree with given inputs according to given tree structure: ");
    System.out.println(localTree.toString());

}

public static void fillTree(BinaryTree<String> thisTree, String[] data, int start, int end){
```

Then there needs to be binaryToBinarySearch method which takes a binary tree and an array of inputs as parameter and rearranges the tree to be binary search tree.

And fillTree method which helps the binaryToBinarySearch method.

As I pointed it out, program needs MergeSort class which sorts the given array.

# 2- Class Diagram



# 3- Problem Solving Approach

For this method's algorithm, I used array sorting algorithm Merge Sort, which sorts the given array. After that, starting with BST's root, I inserted array's middle element to the tree. After handling root node, I inserted the array's left half's middle element to the root node's left child. And finally, I inserted the array's right half's middle element to the root's right child. Recursively I've done that every single node and it became a binary search tree with given structure.

# 4- Test Cases

```
public static void main(String[] args) throws Exception {

    BinaryTree<String> subTree6 = new BinaryTree<>("sdf", null, null);
    BinaryTree<String> subTree5 = new BinaryTree<>("ghhgf", null, null);
    BinaryTree<String> subTree4 = new BinaryTree<>("xyz", null, null);
    BinaryTree<String> subTree3 = new BinaryTree<>("hgfss", null,  null);
    BinaryTree<String> subTree2 = new BinaryTree<>("hgfv", subTree5, subTree6);
    BinaryTree<String> subTree1 = new BinaryTree<>("asdfgf", subTree3, subTree4);
    BinaryTree<String> myTree = new BinaryTree<>("fggfd", subTree1, subTree2);


    String[] inputs = {"burcu", "sultan", "orhan", "gizem", "alper", "elif", "melissa"};

    binaryToBinarySearch(myTree, inputs);
```

# 6- Running Commands and Results

```
Given binary tree structure:
fggfd
  asdfgf
    hgfss
      null
      null
    xyz
      null
      null
  hgfv
    ghhgf
      null
      null
    sdf
      null
      null

Given array of inputs: burcu sultan orhan gizem alper elif melissa


Binary search tree with given inputs according to given tree structure:
gizem
  burcu
    alper
      null
      null
    elif
      null
      null
  orhan
    melissa
      null
      null
    sultan
      null
      null
```