CSE 222/505 - Spring 2023
Homework 5 – Report
Erkut Dere 1801042646

Code Structure and Design:

class Lecture{

void main(){
//Reading file and parsing information
//After parsing storing into 2d string array
//Creating tree with DefaultMutableTreeNode class
//Displaying tree with JTree and Jframe

//Getting user input and calling BFS function
//Getting user input and calling DFS function
//Getting user input and calling post-order traversal function
}
//Helper functions

```
private static DefaultMutableTreeNode findChild(DefaultMutableTreeNode node, String value)
private static String[][] resizeArray(String[][] array, int newSize)
```

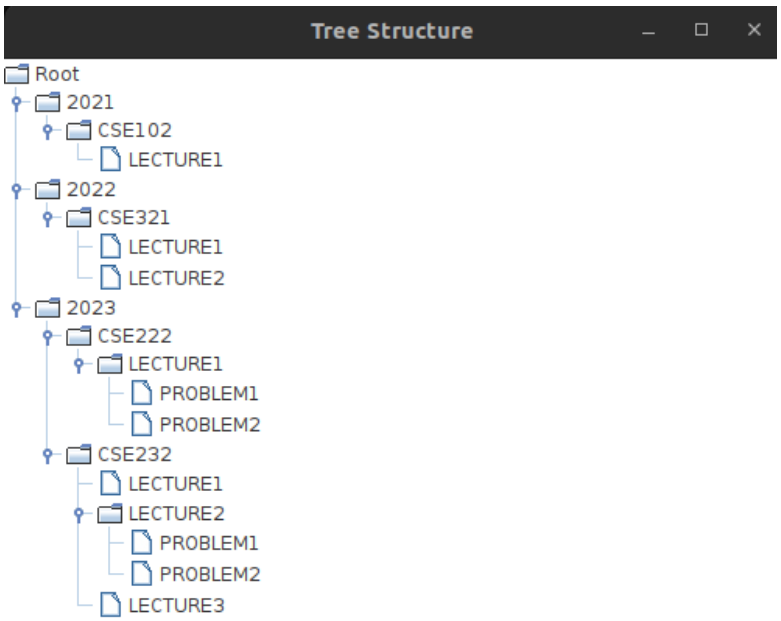
//Implementing three searching algorithm

```
private static void BFSSearch(DefaultMutableTreeNode root, String search)
private static void DFSSearch(DefaultMutableTreeNode root, String search)
private static void postOrderTraversal(DefaultMutableTreeNode node, String search)
```

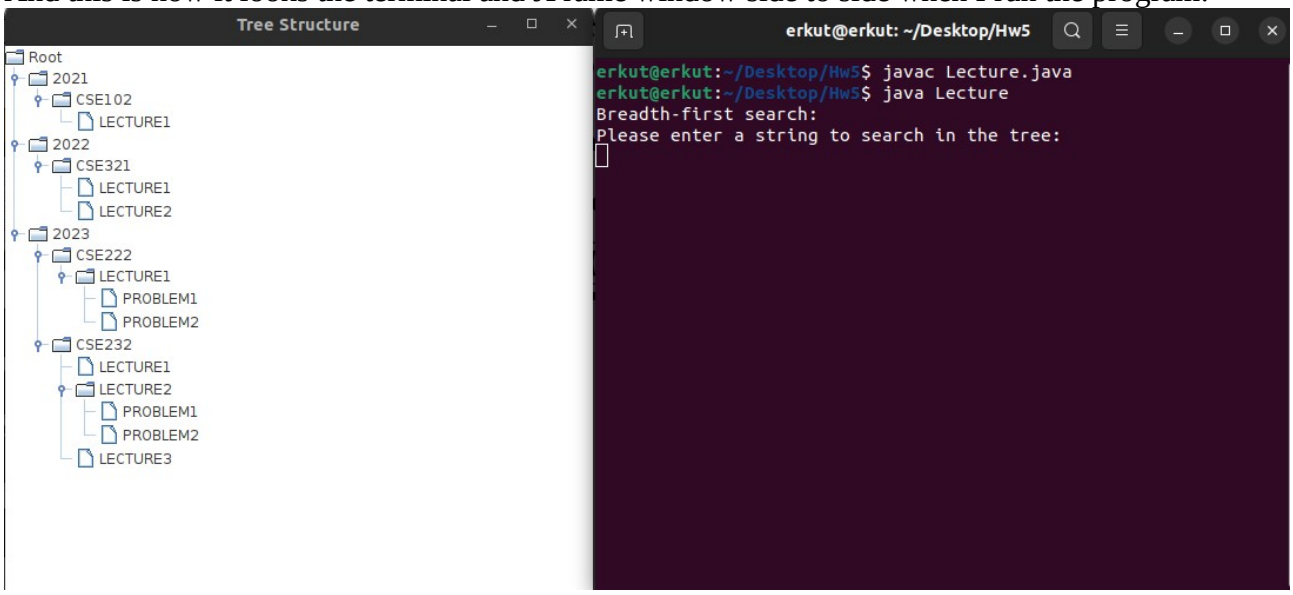
}

Part A:

Here is the output of the tree structure that data is got from tree.txt:



And this is how it looks the terminal and JFrame window side to side when I run the program:



Part B:

BFSSearch method():

In this method I put my tree into a queue. Queue works as a first in first out mechanism so the first element will be the root in the tree. After that we get the children of the root and put them in a queue and their children and so on. We are doing this process from left side to the right side of the tree. While we are doing that we also search for the target string too. If we found it or not found it we are printing the necessary output.

Part C:

DFSSearch method():

This method is similar to BFSSearch method instead we are storing nodes in the stack. Stack works in last in first out mechanism so if we add children of the root to the stack and if we pop the node from the stack we will search from right side of the tree to the left side of the tree (root ,right, left). That is why I use stack to implement Depth-first search algorithm.

Part D:

PostOrderTraversal method():

In this method I used postorderEnumeration method in DefaultMutableTreeNode class. This method creates and returns an enumeration that traverses the subtree rooted at this node in postorder. So my first node in this enumeration will be the leftmost leaf of the tree. I know that I could do this method with stack too but I saw this postorderEnumeration method when I was looking at the DefaultMutableTreeNode class documentation and it makes the work really easy so I wanted use it. After that I use a while loop to search through the nodes with nextElement method and search for the target.

I couldn’t implement the part E because I started this homework late so I am so sorry for this.