GEBZE TECHNICAL UNIVERSITY
CSE222 – HW5 REPORT

Name: Emre OYTUN
Student ID: 200104004099

1) System Design and Method Working Explanations:

CustomTree Class:

- This is a class that constructs a JTree from the .txt file, and makes the tree operations such as breadth-first search, depth-first search, post-order search, and moving of a source node from one year to another.

Fields:

- DefaultMutableTreeNode root: This node keeps the root node of the Jtree.
- JFrame frame: This keeps the frame that the tree is displayed on.
- JTree jtree: This is the Jtree whose root node is kept inside class.

Methods:

- public void constructTreeFromFile():
This method reads the “tree.txt” file dynamically and construct the JTree using the 2D String array that is read from the file.

In reading process; it first reads the txt file line-by-line and finds the total number of non-blank rows. After that it creates a dynamic String array whose row number is determined before but column number of each row is not determined. While reading the file, it parses each line to tokens using “,” as regex and create each row dynamically using the total number of tokens as column number.

After reading process; it calls the helper method constructTreeHelper by sending the 2D dynamic String array as parameter.

- private void constructTreeHelper(String[][]):
It constructs the JTree using the given 2D String array. For each line in the array, it starts from the root and goes to the source node in the line creating a new node if the current path is not in the tree.

- public void showTree():
It creates a new JTree using the root node and embeds it to the Jframe. Then sets Jframe to be visible.

- public void closeTree():
It removes the JTree from the frame and sets Jframe not to be visible. Then disposes the Jframe.

- public void bfs(Object element):
It searches the given object using breadth-first search technique.
To do breadth-first search, it initializes a queue with the root node. Until queue is empty, it pops a node from the queue, checks if the element is equal to the data of this node; if they’re equal then finishes the search. Otherwise, it offers the children of the current node to the queue and goes for the next iteration.

- public void dfs(Object element):
It searches the given object using depth-first search technique.
It calls the recursive helper method dfsRec with the given object, before calling the recursive method it inializes the step number with 1.

- public boolean dfsRec(DefaultMutableTreeNode node, Object element):
Firstly checks if the element and the data in the node are equal, if it is then returns true.
Otherwise, it increments the step number and calls the next recursive methods starting from the right-most child to the left-most child; each time it checks the return value of the recursive call and returns true if the value is true, if it is not true then it continues to call the recursive methods. After all calls are done, then it means the element is not found and returns false.

- public boolean postOrderSearch(Object element):
It searches the given object using the post-order traversal.
It calls the recursive helper method postOrderSearchRec with the given object for each child of the root starting from the left-most to the right-most, before calling the recursive method, it initializes the step number with 1.

- public boolean postOrderSearchRec(DefaultMutableTreeNode node, Object element):
It calls the next recursive methods for the childs starting from the left-most child to the right-most child if any. If one of them returns true, then it returns true. Otherwise, it compares the element and the data inside the node; if they are equal it returns true. Otherwise, it increments the step number and returns false.

- public void moveNode(String source, String destination):
This method moves a node specified in the source path to the destination year.

Firstly, it splits the source using regex “,” to an array of String so that this array contains the paths to the source node. Afterwards it calls deleteNode method to find and delete the source node.
If deleteNode method returns null, then it means there are no such source nodes. Otherwise, it continues to the next step adding the source node to the destination year.

In the second process, it creates a new array which contains the destination path by copying the source path array and changing the first token as the destination year. Afterwards, it starts from the root node and goes until where the source node is being inserted in the way that it creates a new node for the current path if the child node is not found and it continues from the found or newly created child. When it reaches at the end of the path meaning only inserting the source node is remained, it checks if the source node is already one of the childs. If it is not, then it just adds the source node. Otherwise; it removes the node that is being overwritten and puts the other childs that are after this node into a queue. Then it adds the source node at the end of the child list and adds the childs again by popping the queue. The purpose of this process is to keep the order of the children properly.

- private int indexOfChild(DefaultMutableTreeNode node, Object element):
This method tries to find the index of the child which contains the given object as data.
If it founds the child, then it returns its index. Otherwise, it returns false.
To do this, it iterates over all the children of the given node and checks if the data of the child and the element are the same.

- private boolean isBlank(String str):
This method checks if the given string's length is 0 or it only contains space.

Main Class:

This class contains the Main method test the CustomTree class and a helper method to check if a given string can be converted to a number or not.

- public static boolean isNumber(String str):
It checks if the given string can be converted to a number or not by using the Integer.valueOf(String) method.

- public static void main(String[] args):
This is the Main method to test the CustomTree class.
Firstly, it initializes an instance of CustomTree class.

Part A:
- Calls the method constructTreeFromFile for reading the .txt file.
- Calls showTree method to show the tree in the frame.
- Waits input from the user to close the tree.
- Calls closeTree method to close the frame.

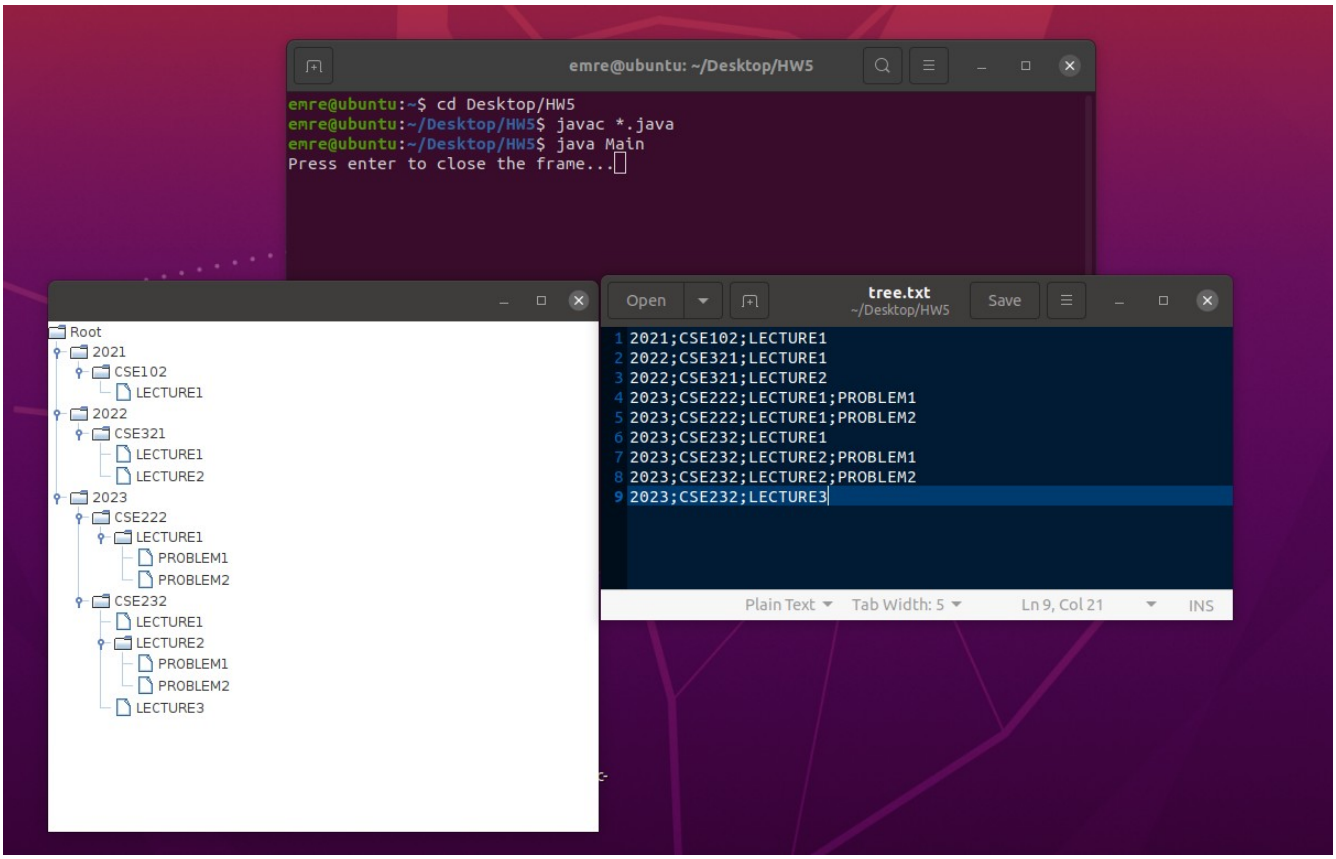
Part B, C, D:
- Takes an input from the user to make the searches.
- Calls bfs method.
- Calls dfs method.
- Calls postOrderSearch method.

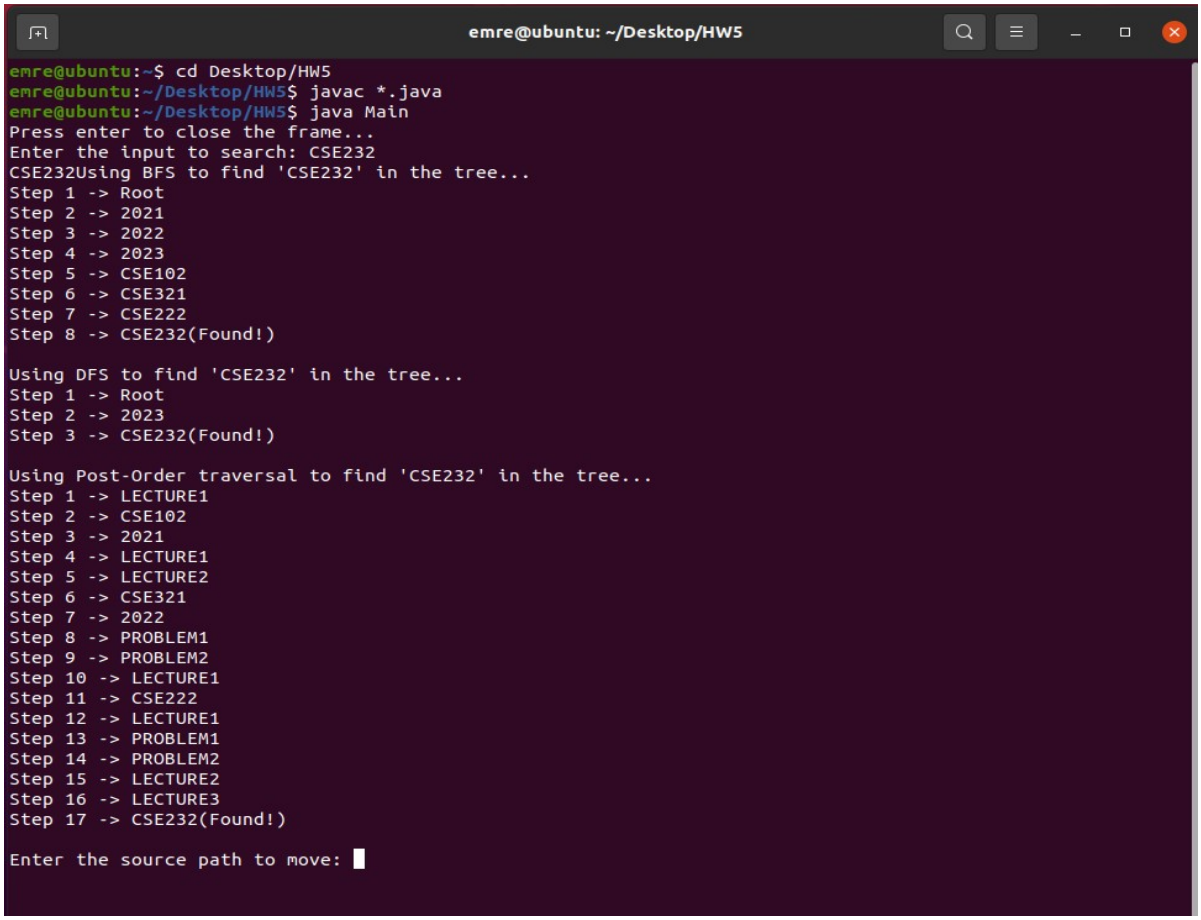
Part E:
- It takes input for the source node by validating the input such that:
* User should enter at least 2 nodes, one for the year and one for the source node.
* The first node in the path should be year.

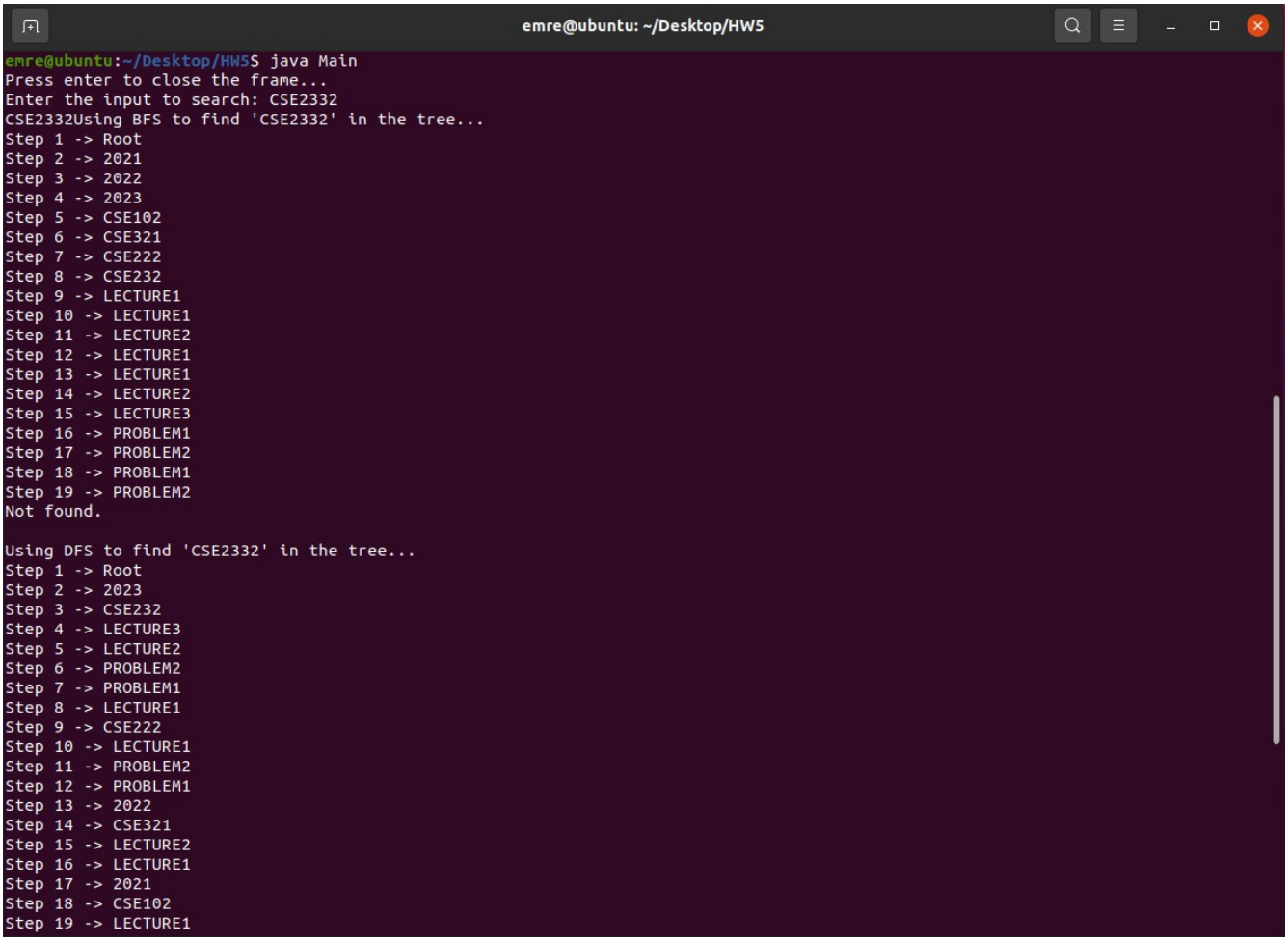
- It takes input for the destination year by validating it such that:
* User should enter the destination such that it can be converted to a year.
* Source year and the destination year should not be the same.

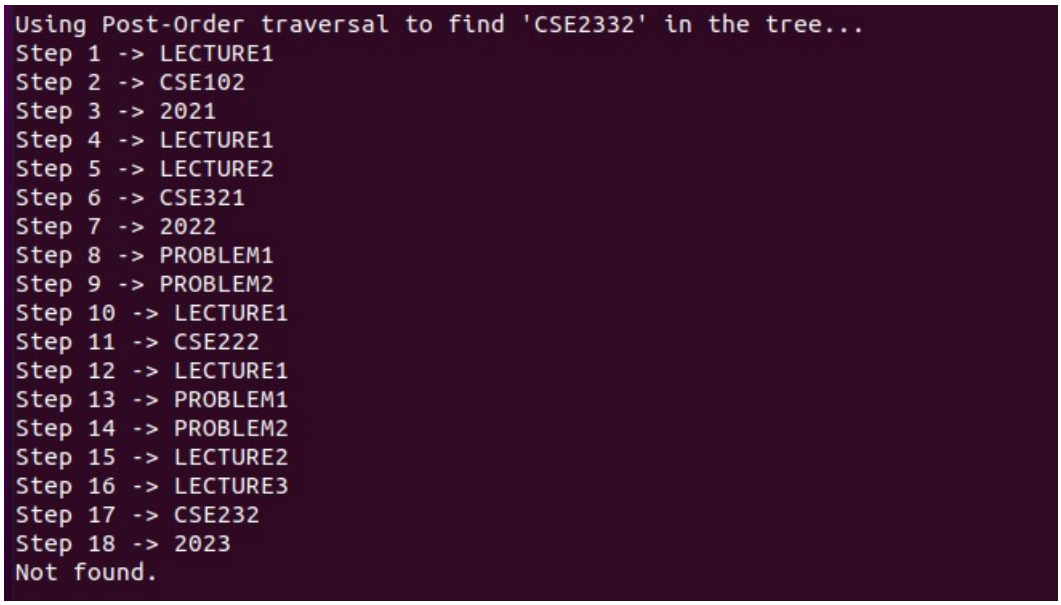
- It calls the moveNode method by passing the source and destination strings as parameters.
- It calls the showTree method.
- Waits input from the user to close the tree.
- It calls the closeTree method.
- It closes the scanner's connection.

2) Screenshots:






emre@ubuntu: ~/Desktop/HW5
emre@ubuntu:~/Desktop/HW5$ java Main
Press enter to close the frame...
Enter the input to search: CSE2332
CSE2332Using BFS to find 'CSE2332' in the tree...
Step 1 -> Root
Step 2 -> 2021
Step 3 -> 2022
Step 4 -> 2023
Step 5 -> CSE102
Step 6 -> CSE321
Step 7 -> CSE222
Step 8 -> CSE232
Step 9 -> LECTURE1
Step 10 -> LECTURE1
Step 11 -> LECTURE2
Step 12 -> LECTURE1
Step 13 -> LECTURE1
Step 14 -> LECTURE2
Step 15 -> LECTURE3
Step 16 -> PROBLEM1
Step 17 -> PROBLEM2
Step 18 -> PROBLEM1
Step 19 -> PROBLEM2
Not found.
Using DFS to find 'CSE2332' in the tree...
Step 1 -> Root
Step 2 -> 2023
Step 3 -> CSE232
Step 4 -> LECTURE3
Step 5 -> LECTURE2
Step 6 -> PROBLEM2
Step 7 -> PROBLEM1
Step 8 -> LECTURE1
Step 9 -> CSE222
Step 10 -> LECTURE1
Step 11 -> PROBLEM2
Step 12 -> PROBLEM1
Step 13 -> 2022
Step 14 -> CSE321
Step 15 -> LECTURE2
Step 16 -> LECTURE1
Step 17 -> 2021
Step 18 -> CSE102
Step 19 -> LECTURE1


Using Post-Order traversal to find 'CSE2332' in the tree...
Step 1 -> LECTURE1
Step 2 -> CSE102
Step 3 -> 2021
Step 4 -> LECTURE1
Step 5 -> LECTURE2
Step 6 -> CSE321
Step 7 -> 2022
Step 8 -> PROBLEM1
Step 9 -> PROBLEM2
Step 10 -> LECTURE1
Step 11 -> CSE222
Step 12 -> LECTURE1
Step 13 -> PROBLEM1
Step 14 -> PROBLEM2
Step 15 -> LECTURE2
Step 16 -> LECTURE3
Step 17 -> CSE232
Step 18 -> 2023
Not found.

```
Enter the source path to move: 2022,CSE321,LECTURE2
Enter the destination: 2023
Press enter to close the frame...
```