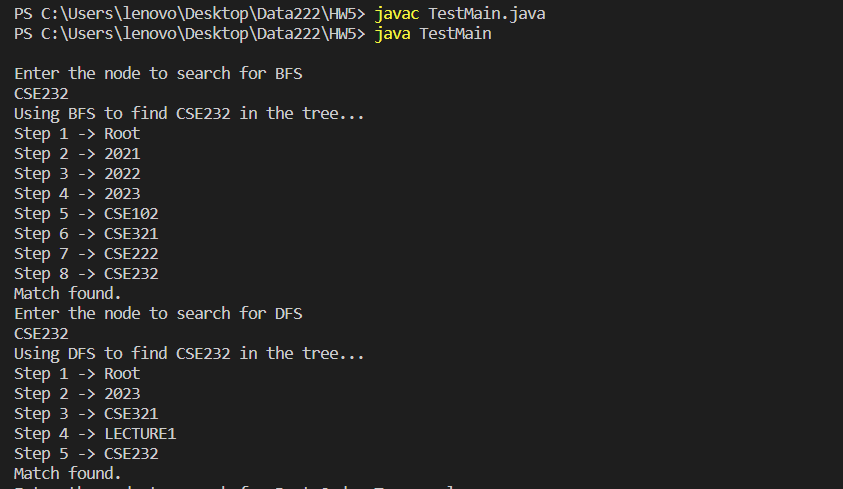
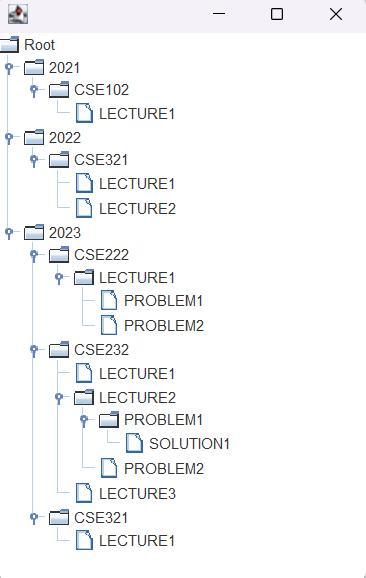
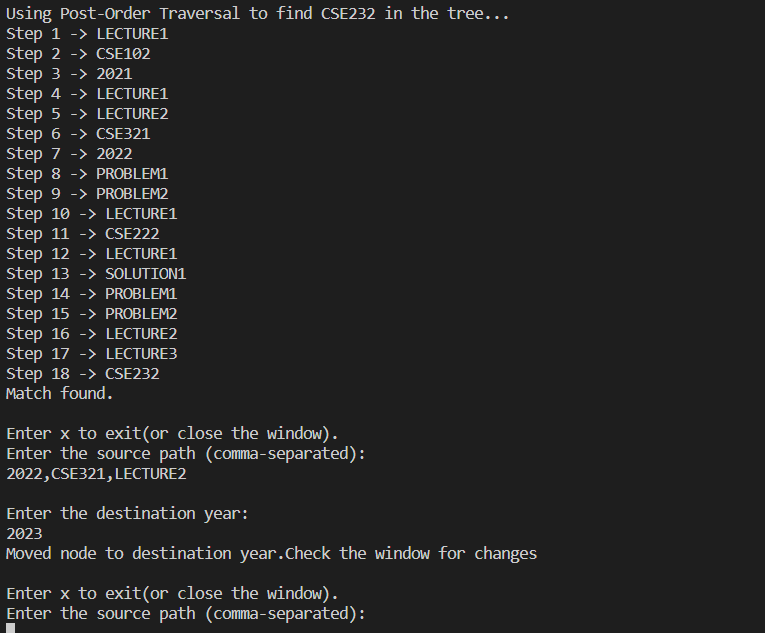
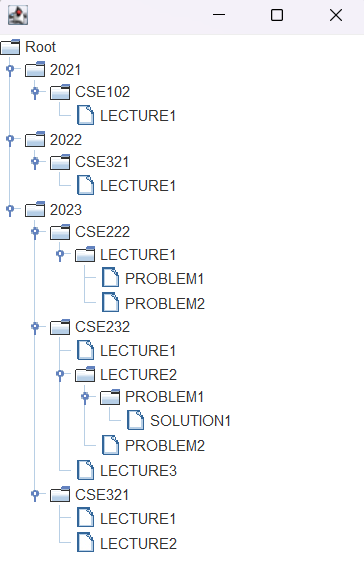
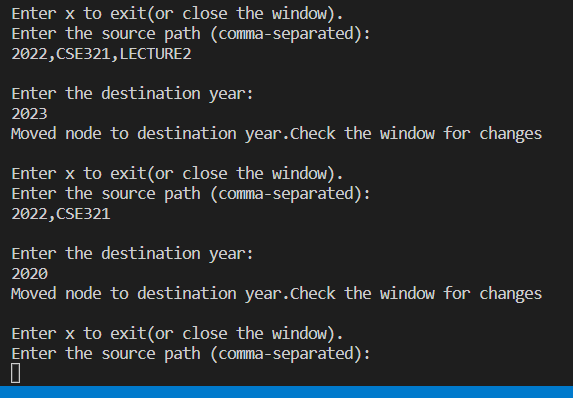
**HÜSEYİN EMRE SEKANLI 200104004049**

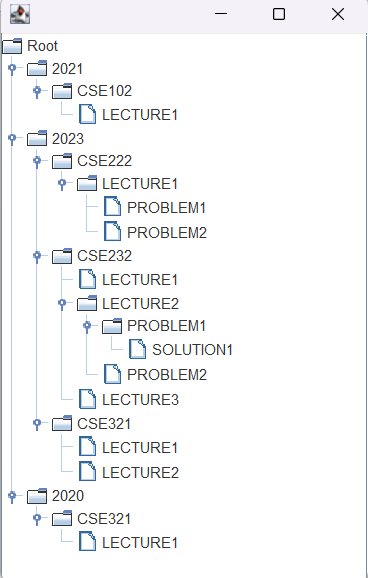
**RUN RESULTS:**

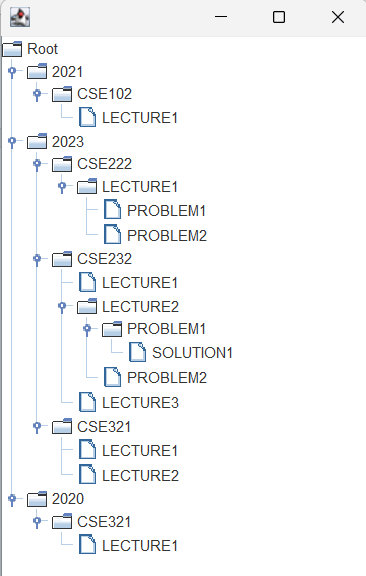
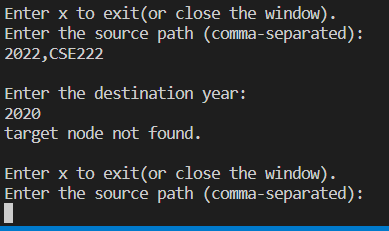
****

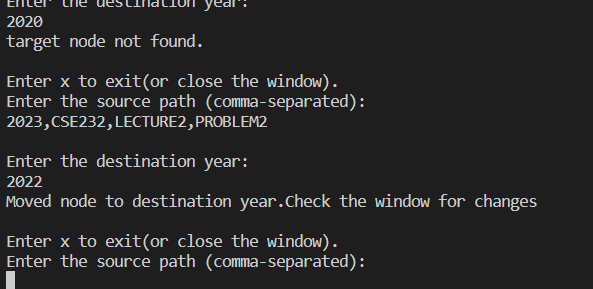
****

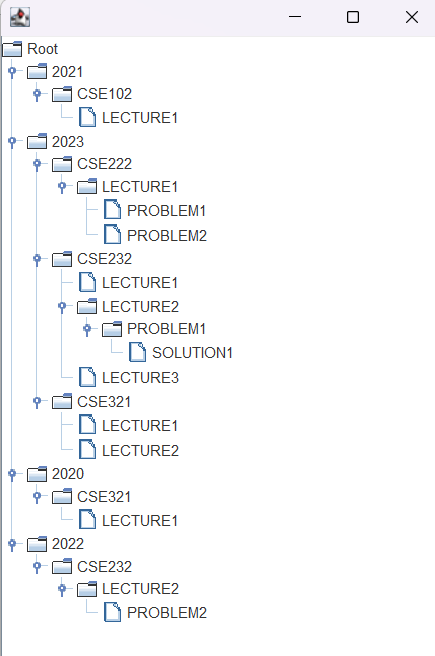
**(WINDOW AFTER THE GIVEN OPERATION:2022,CSE321,LECTURE2 TO 2023)**

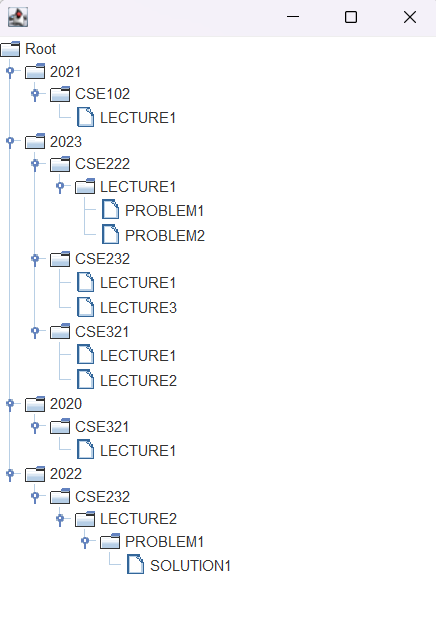
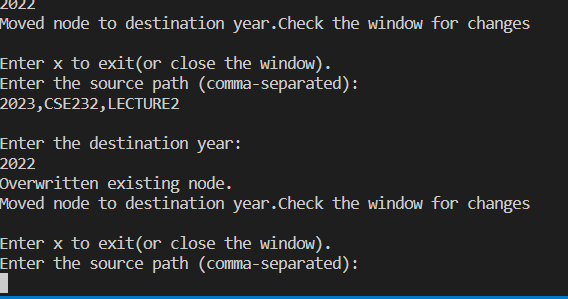
****

**(2022,CSE321 TO 2020)**

**(2022,CSE222 TO 2020 – WINDOW DOESNT CHANGE AND ERROR MESSAGE IS PRINTED)**

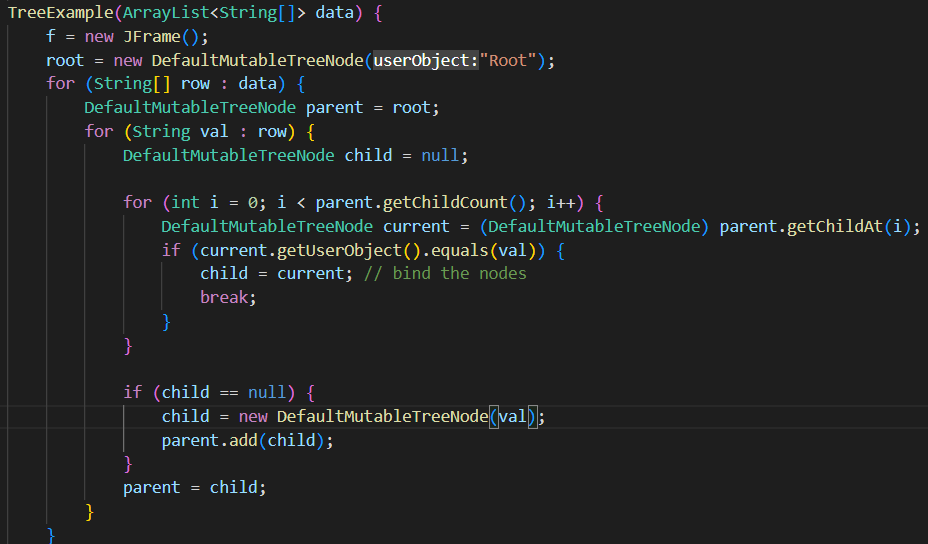
**2023,CSE232,LECTURE2,PROBLEM2 TO 2022**

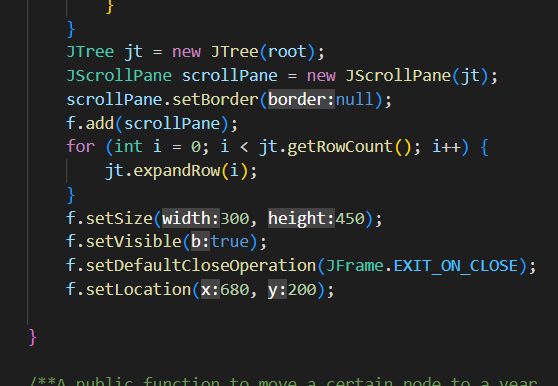
****

**OVERWRITE HAPPENED**

**2023,CSE232,LECTURE2 TO 2022. 2022 ALREADY HAD CSE232,LECTURE2,PROBLEM1 BUT WITH THE OVERWRITE IT NOW HAS SOLUTION1 TOO**

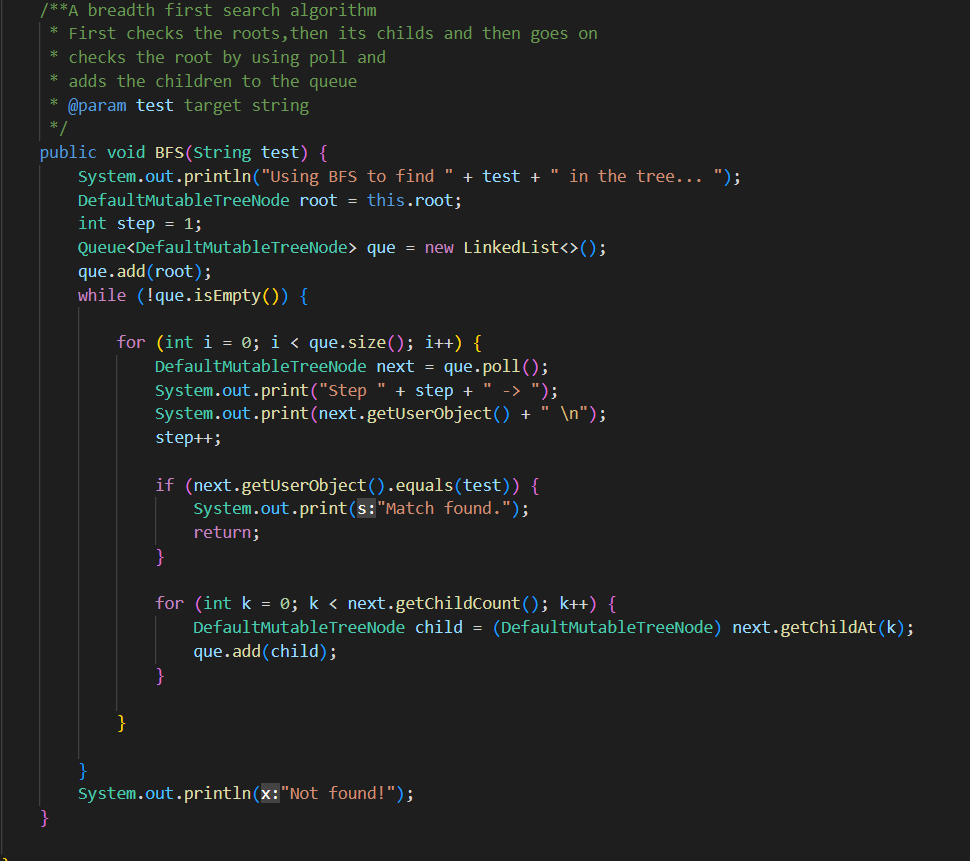
**END OF OUTPUT RESULTS.**

**THE CODE:** 

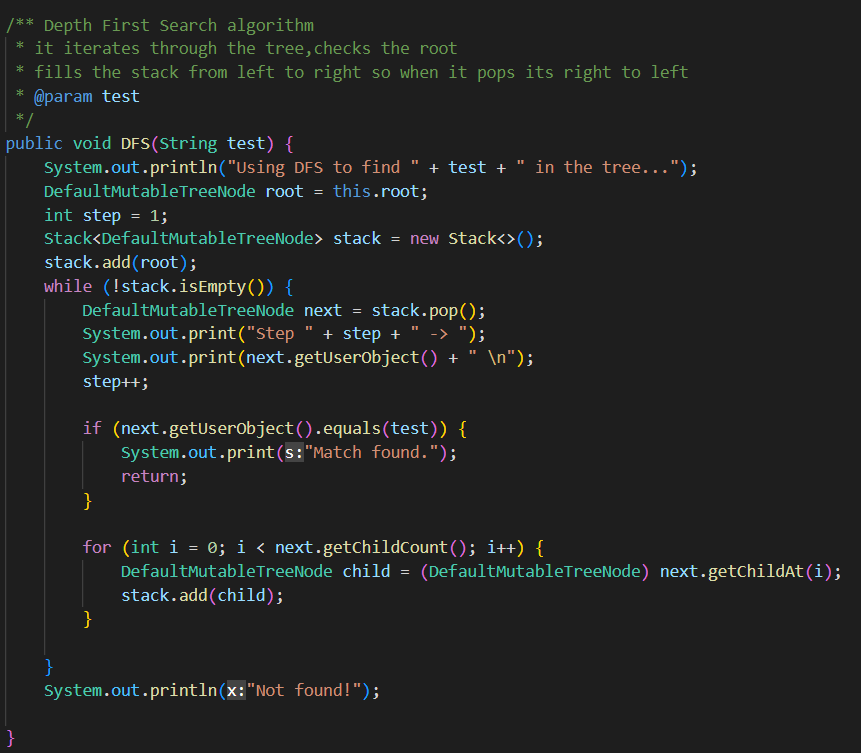


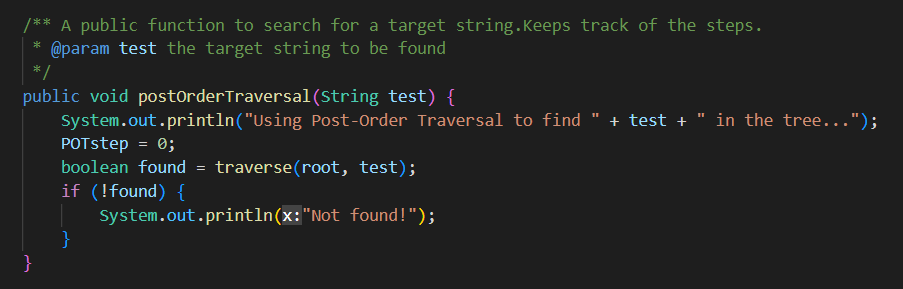
**A TREE EXAMPLE CLASS CONSTRUCTOR.The class holds the root of the tree and a Jframe (also a step count for post-order traversal).It takes an ArrayList of a string array as parameter.It then sets a parent-child relation for each element in the row.It then checks if it already has the same child to avoid duplication,if it has ,binds it.If not, creates a new child node with the data it read from the string.Then makes its parent a child (since the parent is the current node,to access its child nodes)(parent initially set as root).**

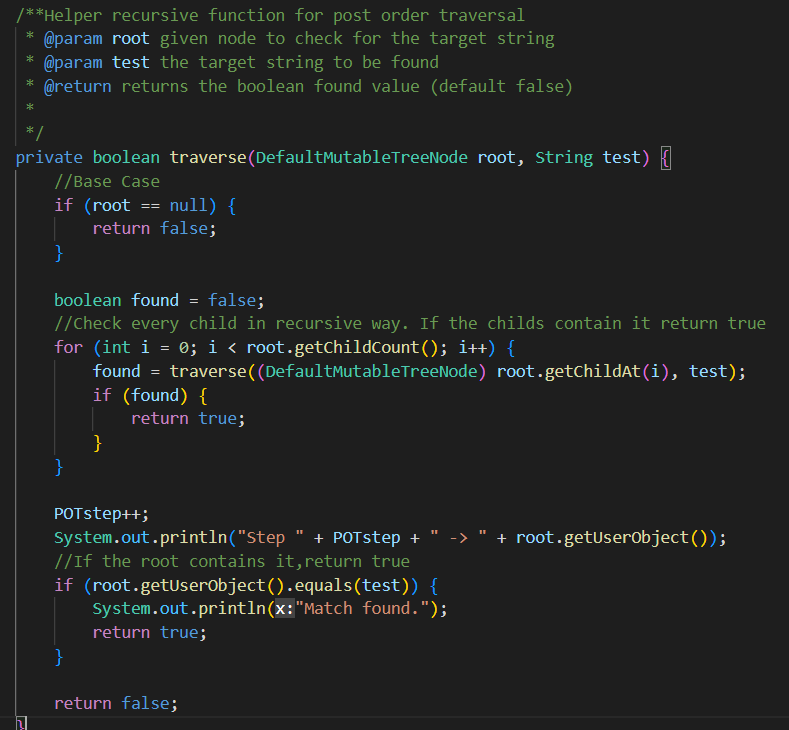
**It then creates a Jtree with the root of our tree as parameter.For adding scrollbar i used Jscrollpane and added jt there.after it add the tree to jframe and set some settings to make it look better.Also close it on window exit.**

****

**THE BFS ALGORITHM:Breadthfirst search. The basic logic of BFS is visit a node,after visiting it,visit all adjacent nodes.So i start the queue with root and entered a loop.I check the root and access its childs,adding them to the queue after i dequeued root.Did that multiple times to reach until the last element.If a match is found,printed found.If no print is done until the end of the code,printed not found.Also kept track of steps in the loop.**

**THE DFS ALGORITHM:.I started the stack with root,checked it while popping it,then added the childs of it to the stack.This kept on until it reached the last element.Since the stack is filled from left to right and 1 pop at a time occurs,it reads the right-most element If a match is found,printed found.If no print is done until the end of the code,printed not found.Also kept track of steps in the loop.**

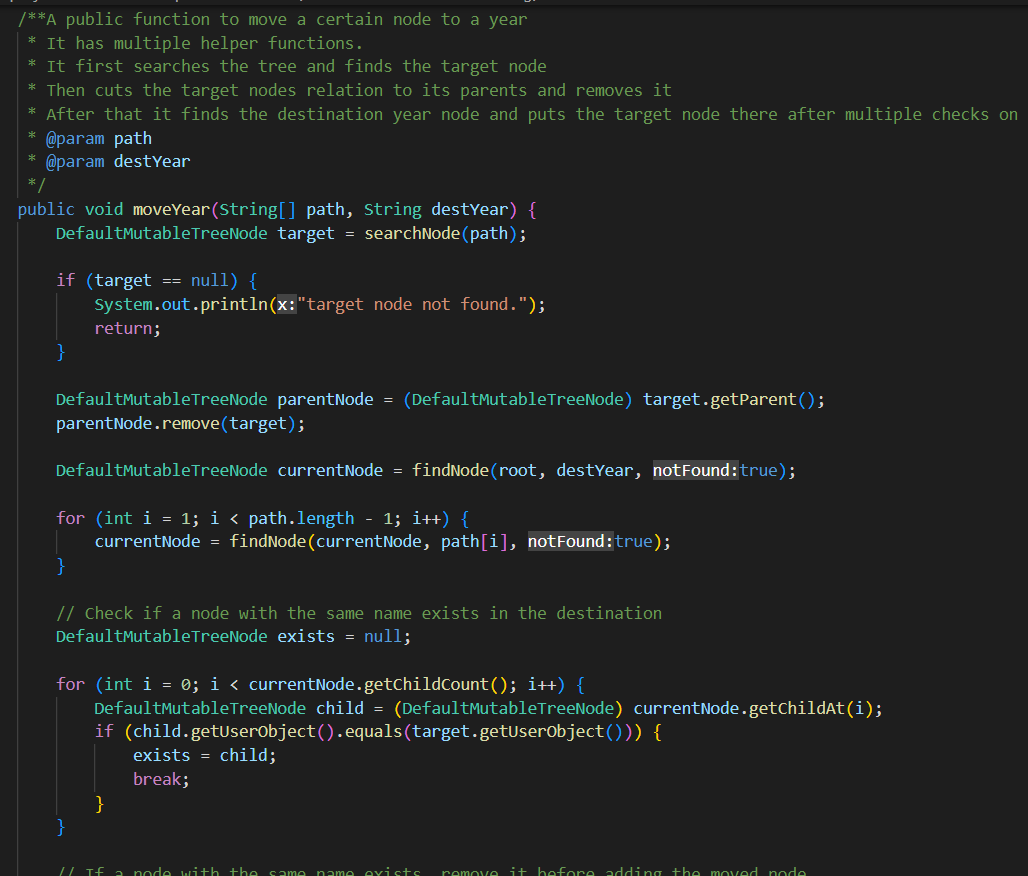
****

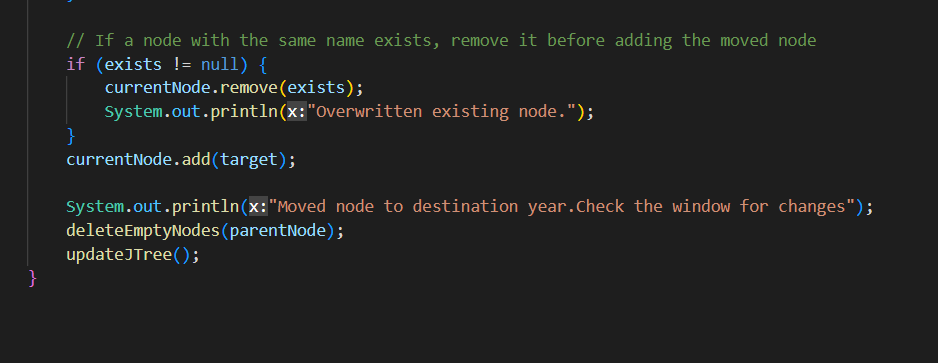
**Post-Order Traversal Algorithm:I added a helper function to use as a recursive.The postordertraversal function calls the recursive function with the original root as its root and the target element to be found.In the recursive function, if root is null it returns false(base case);**

**Boolean found is a tracker i use to check if the program found something.It is defaulted to false .**

**The function checks the childs of the root in a recursive way until it finds the target or the all the elements are checked.If a found occurs,it returns true back till to the postordertraversal function.**

**After checking the childs(if it didnt found the target),it increments the step by 1 and checks the root(not the original root all the time,whatever is the root that recursive function took parameter as) itself.If a match is found,it prints found and returns true; If no return has occurred until it reaches to last line of function,return false.**

****

****

**The Move Algorithm:It first searches if the given path exists.If it doesnt it returns null.**

**If it exists,it creates a parentNode and matches it with the targets parent.Then it cuts ties with its parent.**

**The program later checks for the destination nodes first element(root in this case) , if it finds it returns it to currentnode,if it doesnt find it, it creates it and returns to the currentnode.**

**Does the same thing n times (n being the elements in the path) and gets a currentNode( a target node).It then checks if a node with the same name exists,if it exists,assigns –exists- variable to that child.**

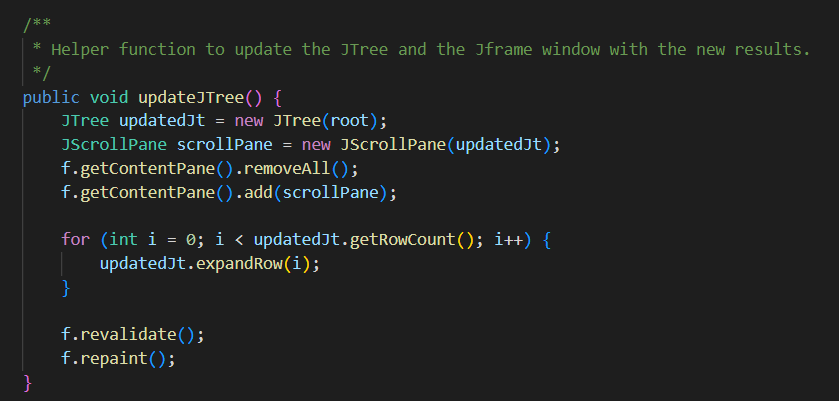
**In the if statement,it removes the existing node . Then adds the new node so an overwrite occurs.**

**If exists is null,it then simply adds the node.**

**It prints a message and then deletes the emptynodes(if there is some) after move operation.**

**It finally updates the Jtree and Jframe with the latest version.**

**HELPERS:**

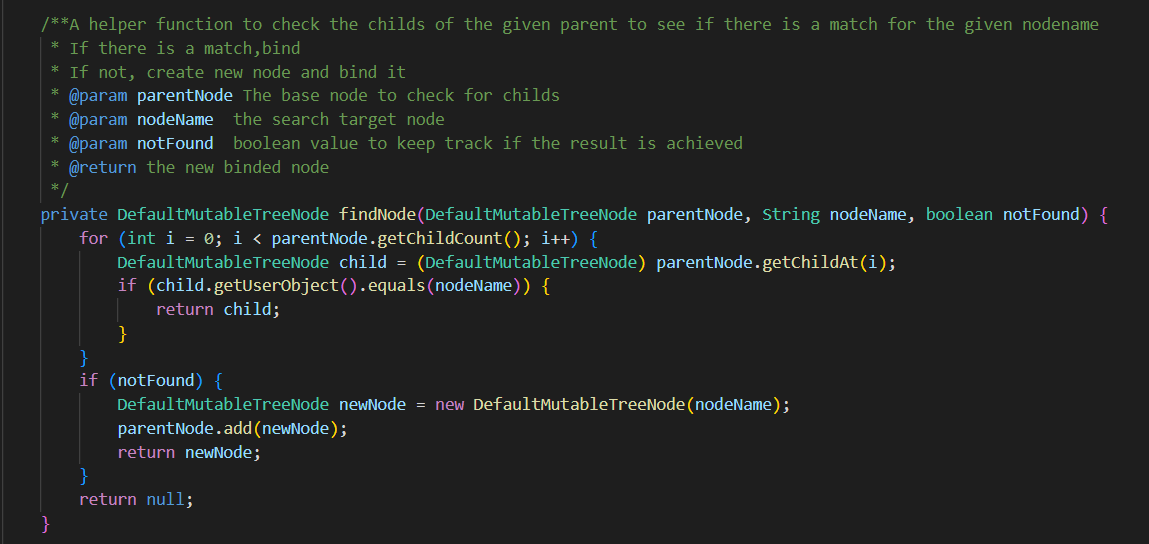
****

**Creates a new Jtree with the latest designed tree,adds a scrollbar to it.**

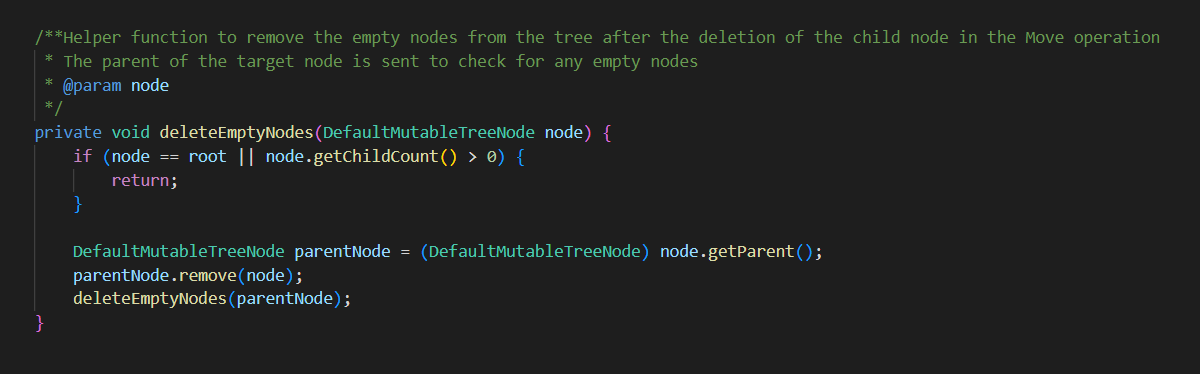
**Removes the existing tree from jframe, then adds the new tree.**

**For loop expands the trees in the window so the user doesnt have to click every one of it one by one.**

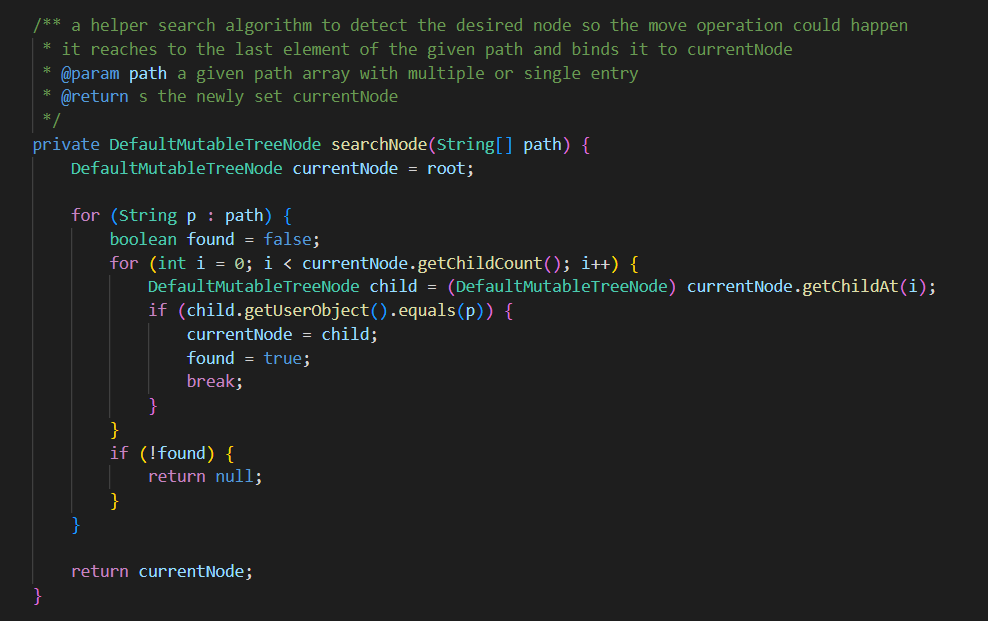
**f.revalidate and f.repaint seems necessary after the remove and add for jframe.(It doesnt update the window otherwise).**

****

**The function checks the target year if it shares nodes with the desired node to move.If it lacks something,the function creates and adds it.The function also binds the parents(if there is) they share.**

****

**The function first checks if the node is the root function or if it has any child.It returns without doing anything if the condition is met.If not,it calls itself recursively and removes itself from its parent until no empty node is left.(this is called after moving a node,to clear whats left behind)**

****

**The code checks the path string to find the target node. If it cant find it it returns null,otherwise returns the target node.**