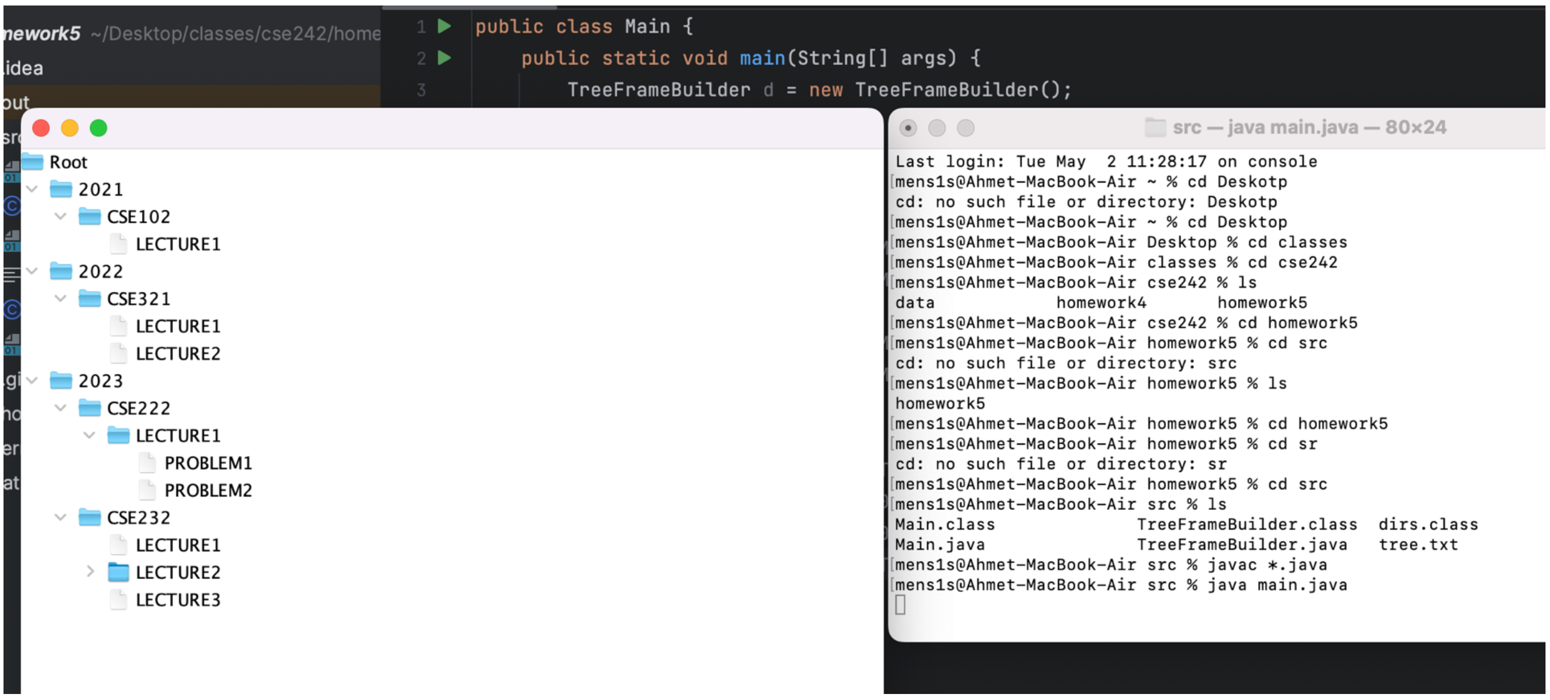
# GTU DEPARTMENT OF ENGINEERING
# COMPUTER ENGINEERING
# SPRING 2022/2023
# HOMEWORK #5

**AHMET YİĞİT**
**200104004066**

## 1. Running Command and Results

### a. Constructor

*i. Constructor is calling 2 functions which are setSize() and getDirs(). SetSize functions are arenging dynamically array size. GetDirs functions read user input from tree.txt and save is dynamiclally reallocated array. GetDirs functions also calls toJTree function to create JTree component automatically by using filled array in getDirs function.*



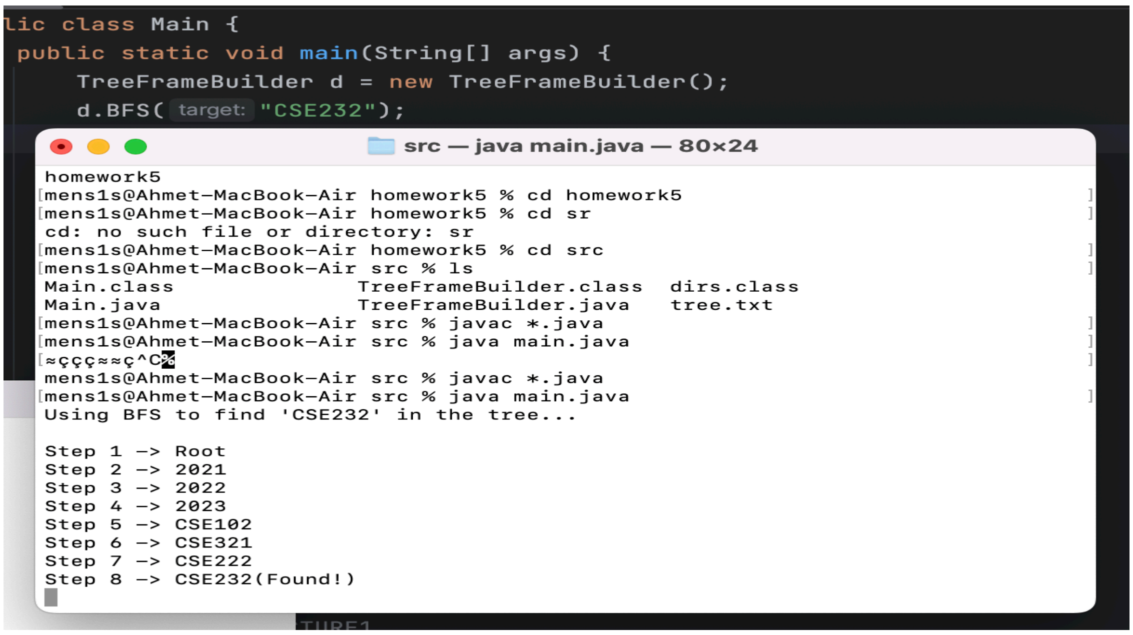
```
public class Main {
    public static void main(String[] args) {
        TreeFrameBuilder d = new TreeFrameBuilder();
```

```
Root
  2021
    CSE102
      LECTURE1
  2022
    CSE321
      LECTURE1
      LECTURE2
  2023
    CSE222
      LECTURE1
        PROBLEM1
        PROBLEM2
    CSE232
      LECTURE1
      LECTURE2
      LECTURE3
```

```
Last login: Tue May  2 11:28:17 on console
mens1s@Ahmet-MacBook-Air ~ % cd Deskotp
cd: no such file or directory: Deskotp
mens1s@Ahmet-MacBook-Air ~ % cd Desktop
mens1s@Ahmet-MacBook-Air Desktop % cd classes
mens1s@Ahmet-MacBook-Air classes % cd cse242
mens1s@Ahmet-MacBook-Air cse242 % ls
data            homework4       homework5
mens1s@Ahmet-MacBook-Air cse242 % cd homework5
mens1s@Ahmet-MacBook-Air homework5 % cd src
cd: no such file or directory: src
mens1s@Ahmet-MacBook-Air homework5 % ls
homework5
mens1s@Ahmet-MacBook-Air homework5 % cd homework5
mens1s@Ahmet-MacBook-Air homework5 % cd sr
cd: no such file or directory: sr
mens1s@Ahmet-MacBook-Air homework5 % cd src
mens1s@Ahmet-MacBook-Air src % ls
Main.class              TreeFrameBuilder.class  dirs.class
Main.java               TreeFrameBuilder.java   tree.txt
mens1s@Ahmet-MacBook-Air src % javac *.java
mens1s@Ahmet-MacBook-Air src % java main.java
```

### b. BFS Search algorithm

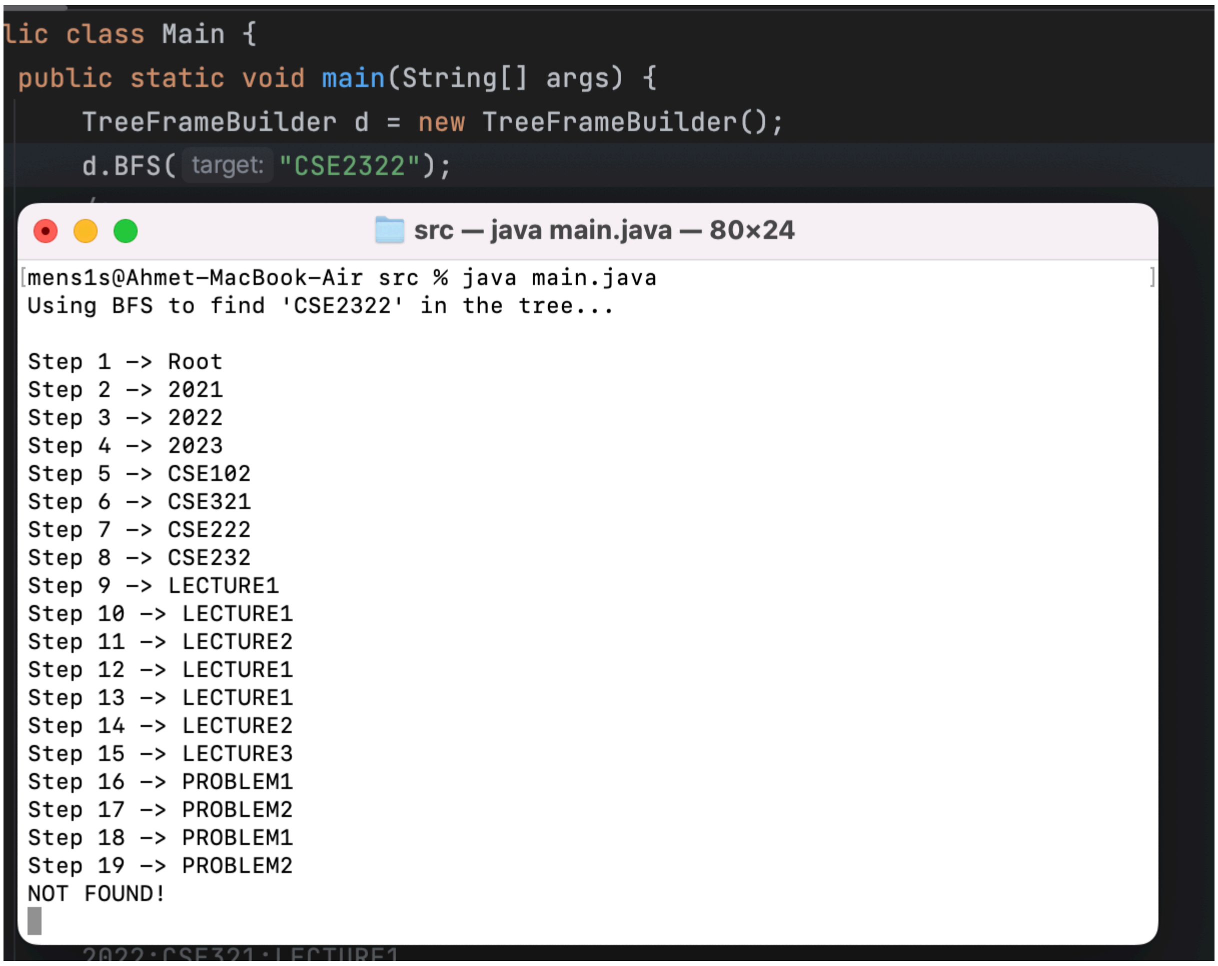
#### i. Success



```
lic class Main {
 public static void main(String[] args) {
     TreeFrameBuilder d = new TreeFrameBuilder();
     d.BFS( target: "CSE232");
```

```
homework5
mens1s@Ahmet-MacBook-Air homework5 % cd homework5
mens1s@Ahmet-MacBook-Air homework5 % cd sr
cd: no such file or directory: sr
mens1s@Ahmet-MacBook-Air homework5 % cd src
mens1s@Ahmet-MacBook-Air src % ls
Main.class              TreeFrameBuilder.class  dirs.class
Main.java               TreeFrameBuilder.java   tree.txt
mens1s@Ahmet-MacBook-Air src % javac *.java
mens1s@Ahmet-MacBook-Air src % java main.java
≈çççç≈≈ç^C%
mens1s@Ahmet-MacBook-Air src % javac *.java
mens1s@Ahmet-MacBook-Air src % java main.java
Using BFS to find 'CSE232' 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(Found!)
```

ii. Unsuccess



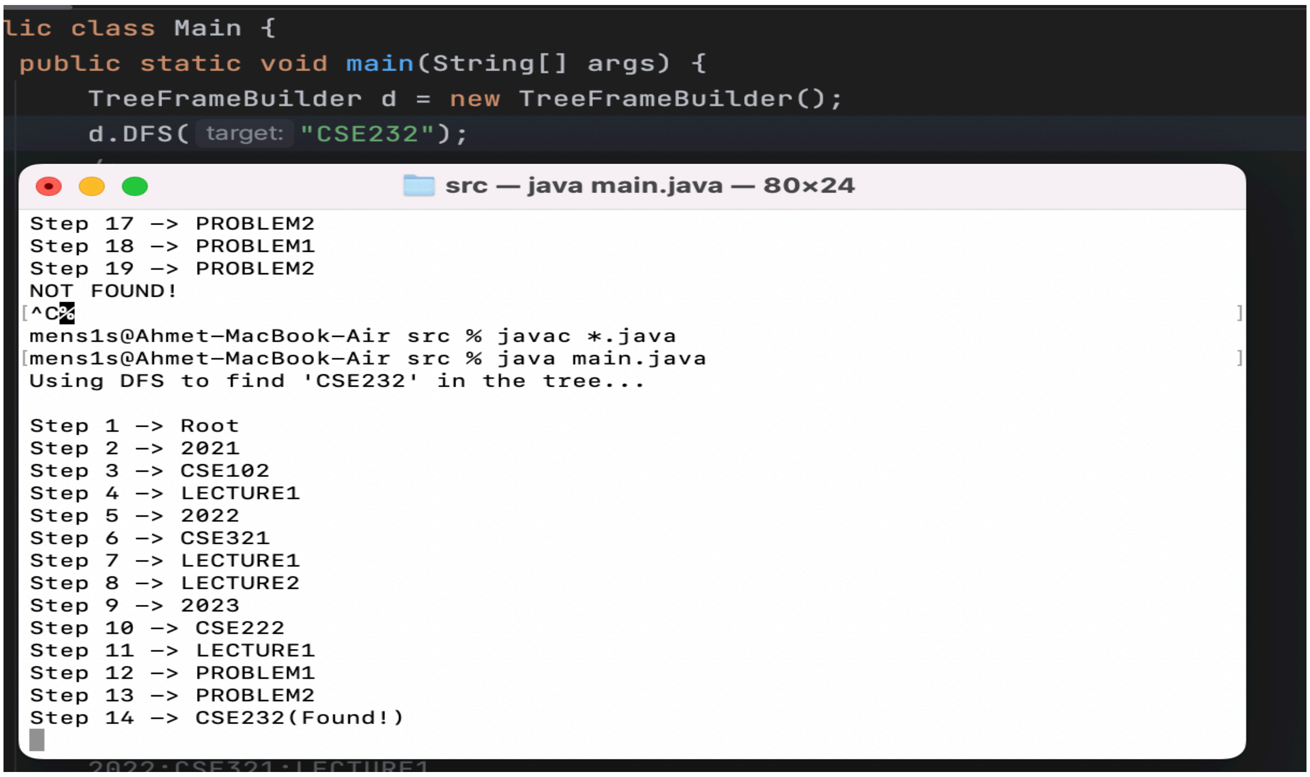
```
lic class Main {
 public static void main(String[] args) {
     TreeFrameBuilder d = new TreeFrameBuilder();
     d.BFS( target: "CSE2322");
```

src — java main.java — 80×24

```
mens1s@Ahmet-MacBook-Air src % java main.java
Using BFS to find 'CSE2322' 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!
```

c. DFS Search

i. Success

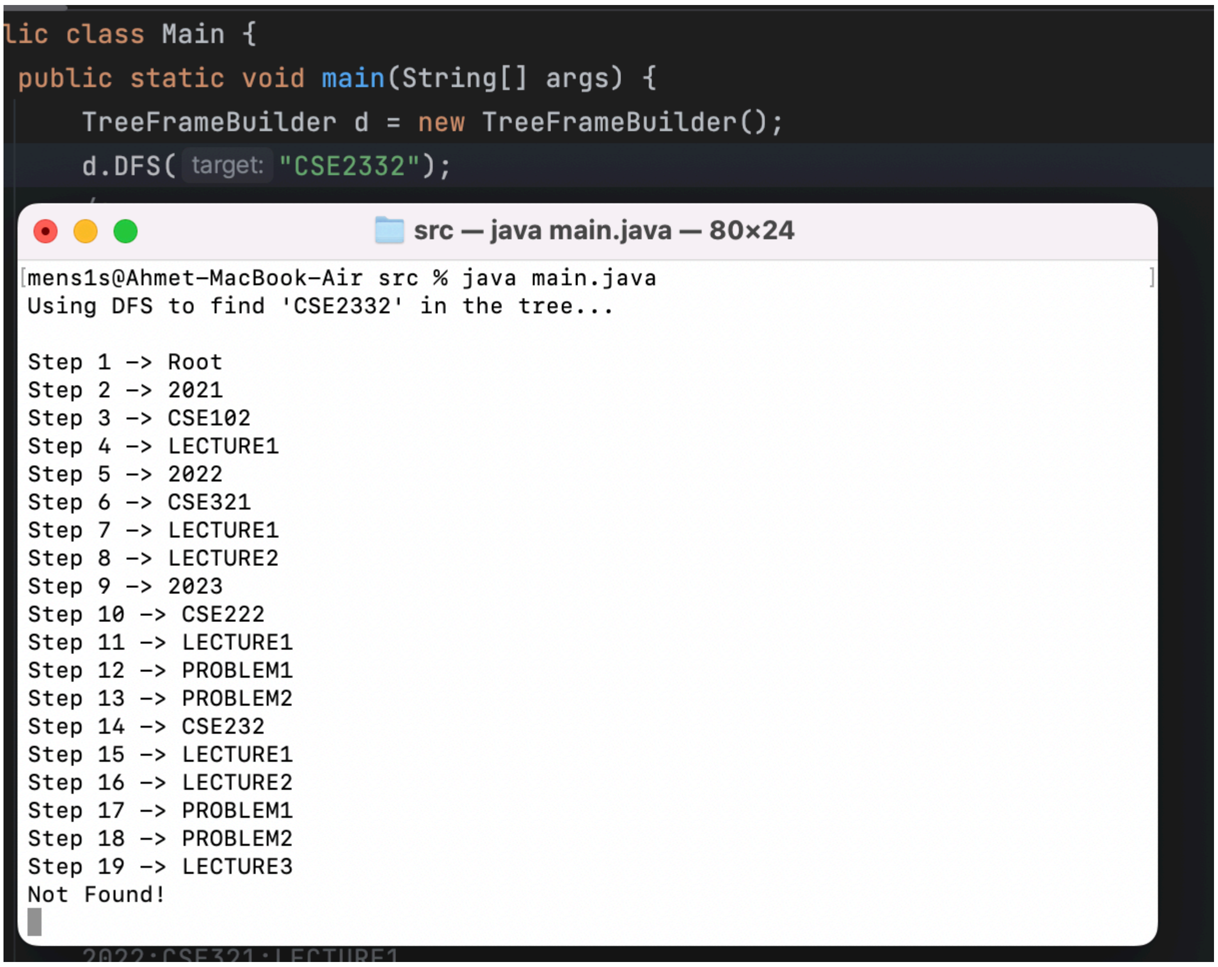


```
lic class Main {
 public static void main(String[] args) {
     TreeFrameBuilder d = new TreeFrameBuilder();
     d.DFS( target: "CSE232");
```

src — java main.java — 80×24

```
Step 17 -> PROBLEM2
Step 18 -> PROBLEM1
Step 19 -> PROBLEM2
NOT FOUND!
^C%
mens1s@Ahmet-MacBook-Air src % javac *.java
mens1s@Ahmet-MacBook-Air src % java main.java
Using DFS to find 'CSE232' in the tree...

Step 1 -> Root
Step 2 -> 2021
Step 3 -> CSE102
Step 4 -> LECTURE1
Step 5 -> 2022
Step 6 -> CSE321
Step 7 -> LECTURE1
Step 8 -> LECTURE2
Step 9 -> 2023
Step 10 -> CSE222
Step 11 -> LECTURE1
Step 12 -> PROBLEM1
Step 13 -> PROBLEM2
Step 14 -> CSE232(Found!)
```

ii. Unsuccess



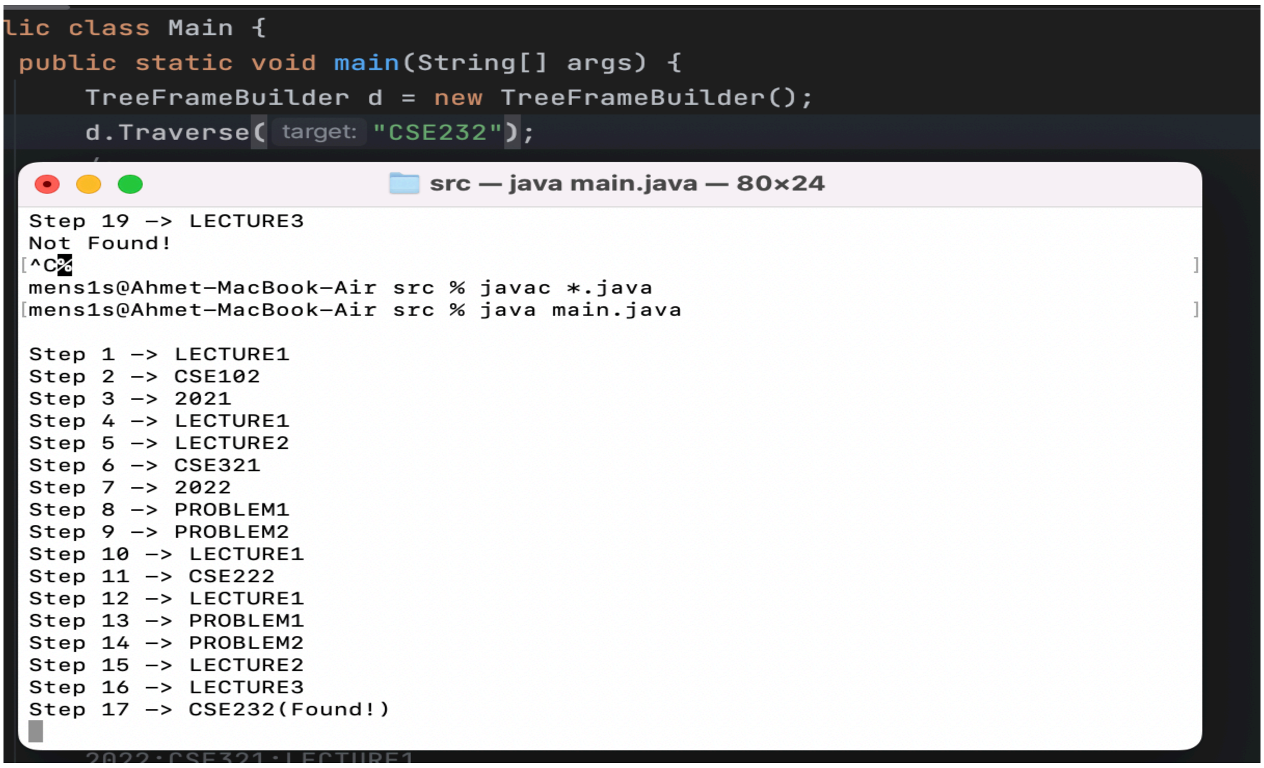
```
lic class Main {
 public static void main(String[] args) {
     TreeFrameBuilder d = new TreeFrameBuilder();
     d.DFS( target: "CSE2332");
```

src — java main.java — 80×24

```
mens1s@Ahmet-MacBook-Air src % java main.java
Using DFS to find 'CSE2332' in the tree...

Step 1 -> Root
Step 2 -> 2021
Step 3 -> CSE102
Step 4 -> LECTURE1
Step 5 -> 2022
Step 6 -> CSE321
Step 7 -> LECTURE1
Step 8 -> LECTURE2
Step 9 -> 2023
Step 10 -> CSE222
Step 11 -> LECTURE1
Step 12 -> PROBLEM1
Step 13 -> PROBLEM2
Step 14 -> CSE232
Step 15 -> LECTURE1
Step 16 -> LECTURE2
Step 17 -> PROBLEM1
Step 18 -> PROBLEM2
Step 19 -> LECTURE3
Not Found!
```

d. Traverse Search

i. Success

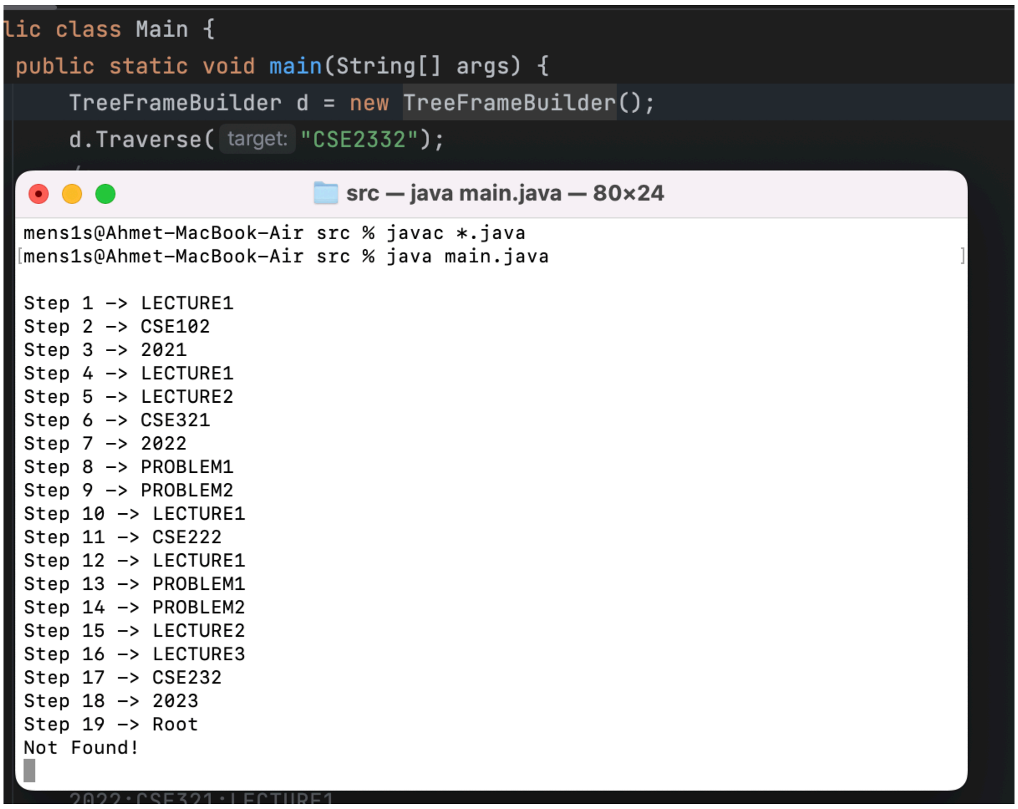


```
lic class Main {
 public static void main(String[] args) {
     TreeFrameBuilder d = new TreeFrameBuilder();
     d.Traverse( target: "CSE232");
```

src — java main.java — 80×24

```
Step 19 -> LECTURE3
Not Found!
^C%
mens1s@Ahmet-MacBook-Air src % javac *.java
mens1s@Ahmet-MacBook-Air src % java main.java

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(Found!)
```

### ii. Unsuccess

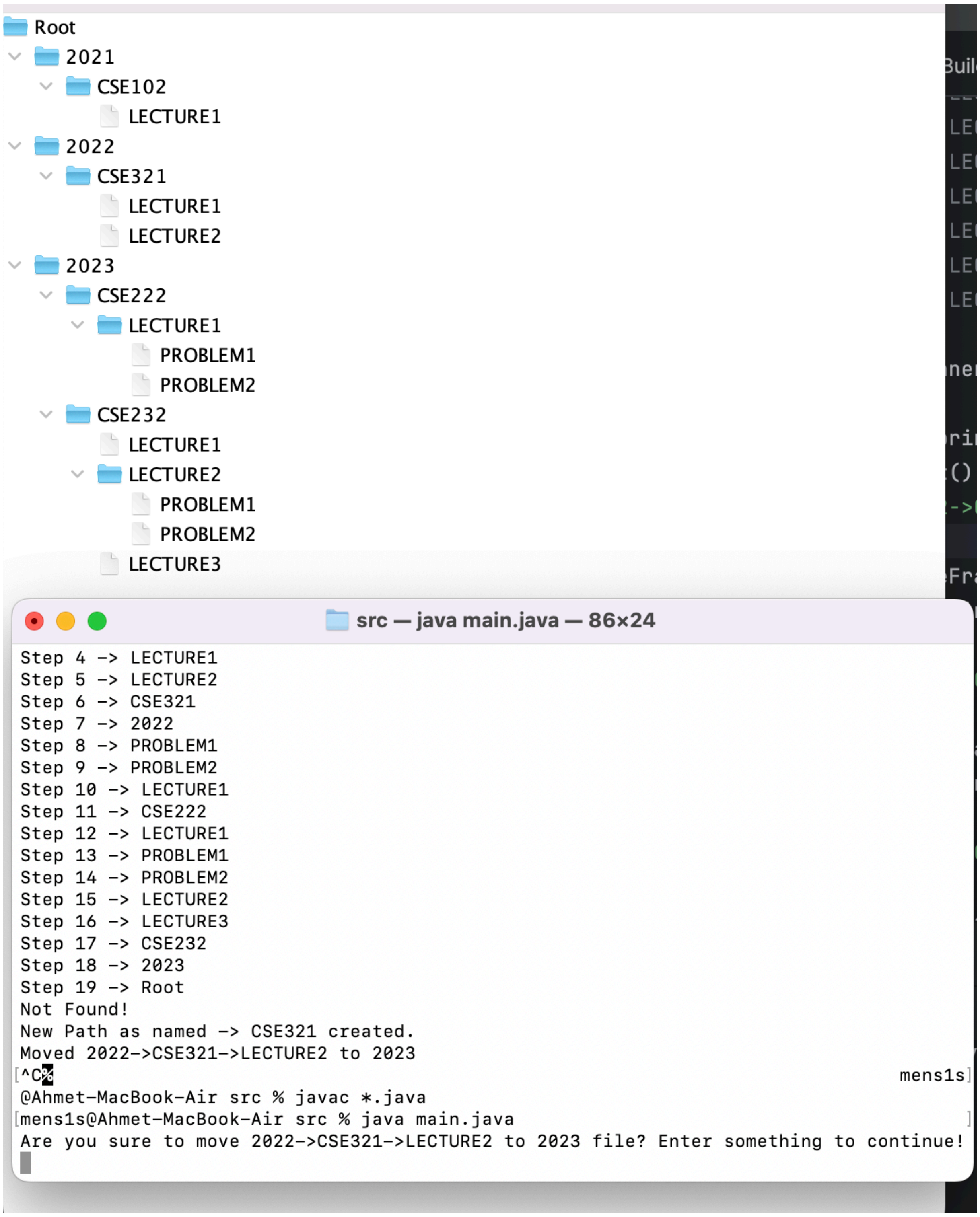


```
lic class Main {
 public static void main(String[] args) {
     TreeFrameBuilder d = new TreeFrameBuilder();
     d.Traverse( target: "CSE2332");
```

```
mens1s@Ahmet-MacBook-Air src % javac *.java
mens1s@Ahmet-MacBook-Air src % java main.java

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
Step 19 -> Root
Not Found!
```

### e. Move Steps

#### i. First Stage

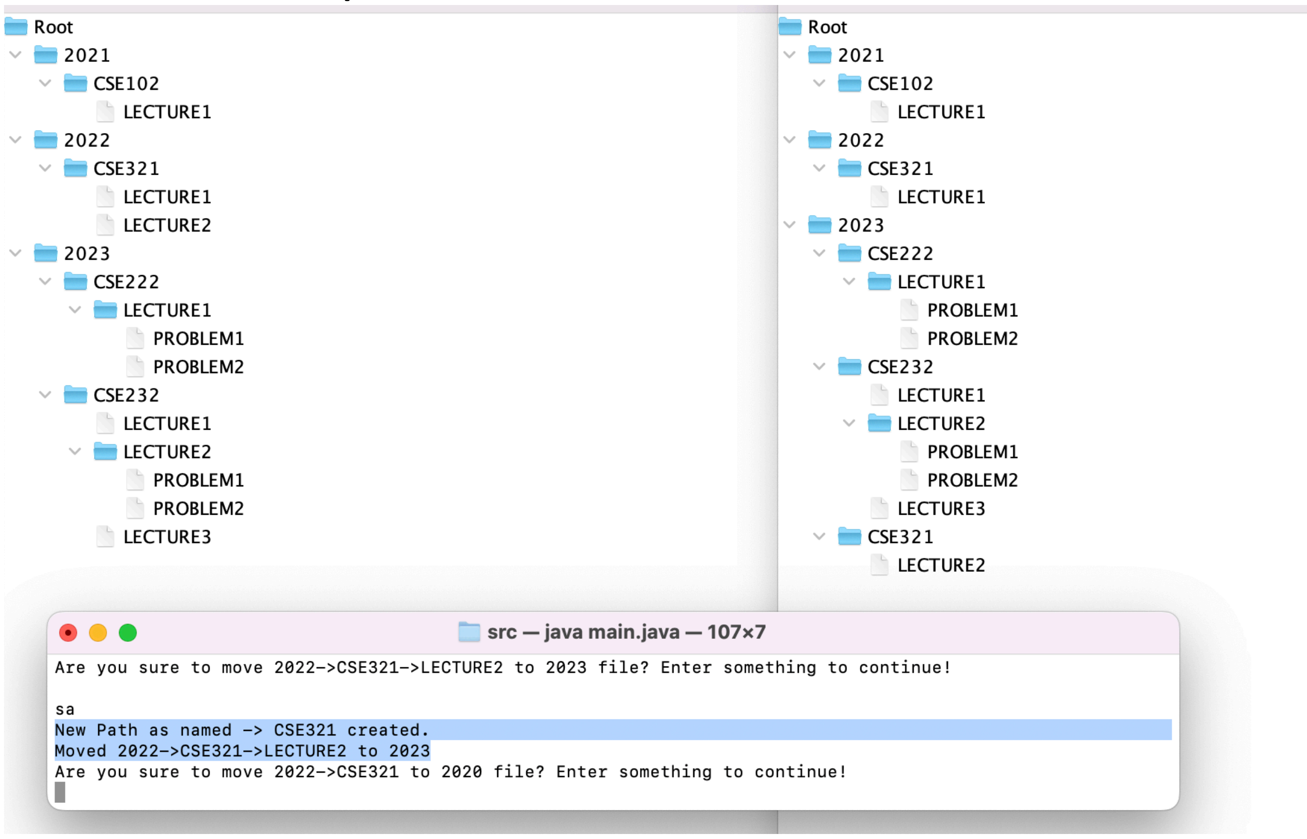


```
Root
  2021
    CSE102
      LECTURE1
  2022
    CSE321
      LECTURE1
      LECTURE2
  2023
    CSE222
      LECTURE1
        PROBLEM1
        PROBLEM2
    CSE232
      LECTURE1
      LECTURE2
        PROBLEM1
        PROBLEM2
      LECTURE3
```

src — java main.java — 86×24

```
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
Step 19 -> Root
Not Found!
New Path as named -> CSE321 created.
Moved 2022->CSE321->LECTURE2 to 2023
[^C%                                                                             mens1s]
@Ahmet-MacBook-Air src % javac *.java
[mens1s@Ahmet-MacBook-Air src % java main.java                                         ]
Are you sure to move 2022->CSE321->LECTURE2 to 2023 file? Enter something to continue!
```

## ii. Move Step-1



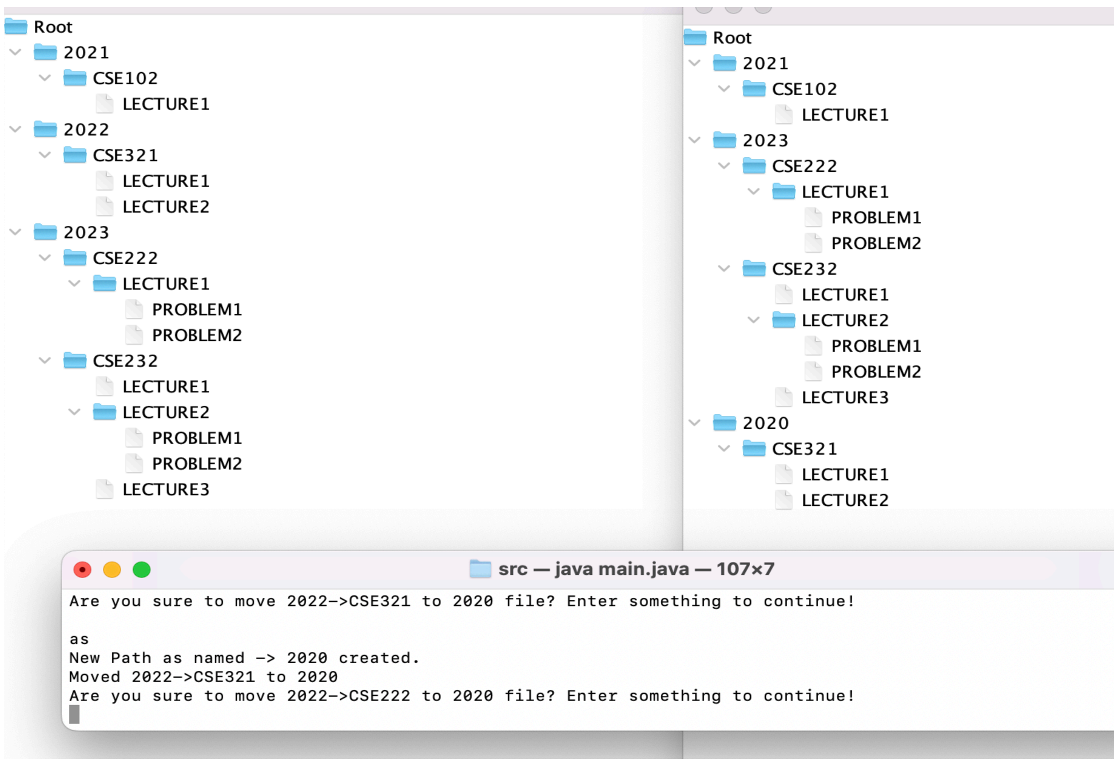
```
Root
  2021
    CSE102
      LECTURE1
  2022
    CSE321
      LECTURE1
      LECTURE2
  2023
    CSE222
      LECTURE1
        PROBLEM1
        PROBLEM2
    CSE232
      LECTURE1
      LECTURE2
        PROBLEM1
        PROBLEM2
      LECTURE3
```

```
Root
  2021
    CSE102
      LECTURE1
  2022
    CSE321
      LECTURE1
  2023
    CSE222
      LECTURE1
        PROBLEM1
        PROBLEM2
    CSE232
      LECTURE1
      LECTURE2
        PROBLEM1
        PROBLEM2
      LECTURE3
    CSE321
      LECTURE2
```

src — java main.java — 107×7

```
Are you sure to move 2022->CSE321->LECTURE2 to 2023 file? Enter something to continue!

sa
New Path as named -> CSE321 created.
Moved 2022->CSE321->LECTURE2 to 2023
Are you sure to move 2022->CSE321 to 2020 file? Enter something to continue!
```

## iii. Move Step -2



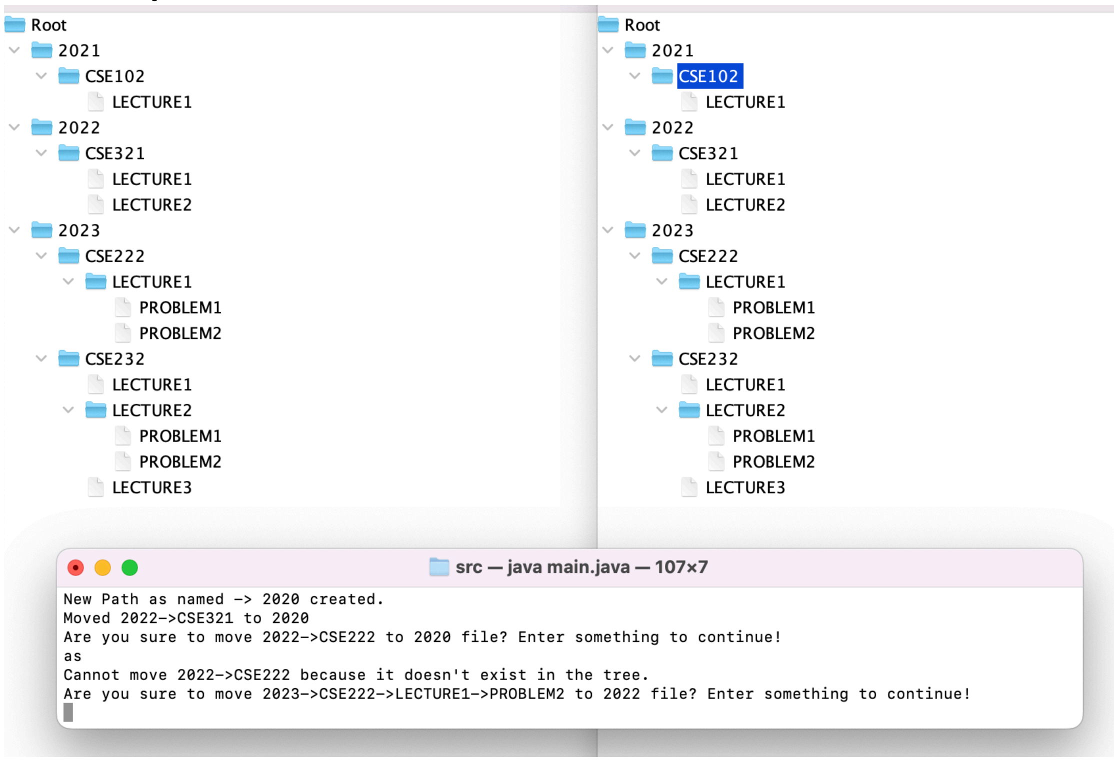
```
Root
  2021
    CSE102
      LECTURE1
  2022
    CSE321
      LECTURE1
      LECTURE2
  2023
    CSE222
      LECTURE1
        PROBLEM1
        PROBLEM2
    CSE232
      LECTURE1
      LECTURE2
        PROBLEM1
        PROBLEM2
      LECTURE3
```

```
Root
  2021
    CSE102
      LECTURE1
  2023
    CSE222
      LECTURE1
        PROBLEM1
        PROBLEM2
    CSE232
      LECTURE1
      LECTURE2
        PROBLEM1
        PROBLEM2
      LECTURE3
  2020
    CSE321
      LECTURE1
      LECTURE2
```

src — java main.java — 107×7

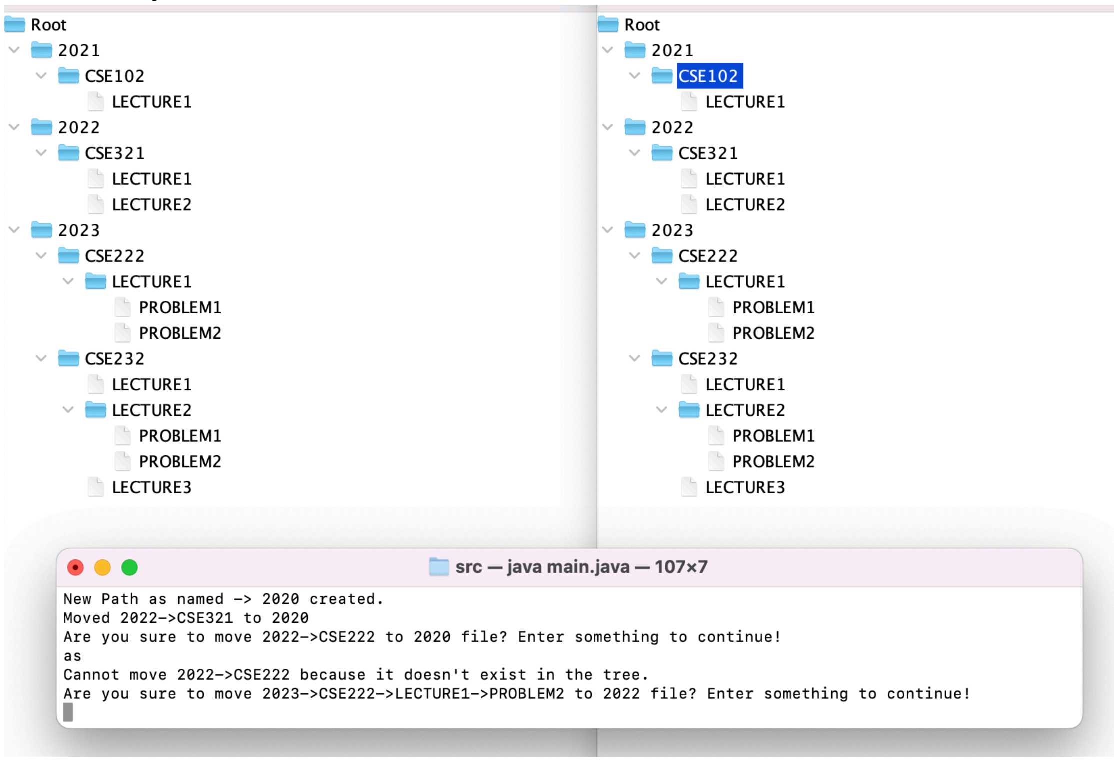
```
Are you sure to move 2022->CSE321 to 2020 file? Enter something to continue!

as
New Path as named -> 2020 created.
Moved 2022->CSE321 to 2020
Are you sure to move 2022->CSE222 to 2020 file? Enter something to continue!
```

iv. **Move Step -3**



```
New Path as named -> 2020 created.
Moved 2022->CSE321 to 2020
Are you sure to move 2022->CSE222 to 2020 file? Enter something to continue!
as
Cannot move 2022->CSE222 because it doesn't exist in the tree.
Are you sure to move 2023->CSE222->LECTURE1->PROBLEM2 to 2022 file? Enter something to continue!
```

v. **Move Step -4**



```
New Path as named -> 2020 created.
Moved 2022->CSE321 to 2020
Are you sure to move 2022->CSE222 to 2020 file? Enter something to continue!
as
Cannot move 2022->CSE222 because it doesn't exist in the tree.
Are you sure to move 2023->CSE222->LECTURE1->PROBLEM2 to 2022 file? Enter something to continue!
```

## 2. Time Complexity Analysis

### a. BFS Search

i. **In BFS Search algorithms time complexity is always O(V+E) because every Vertexies in tree and all edges which** connected to these nodes has to be searched our target value is that or not.

b. DFS Search
    i. It is same as BFS search because their algorithm needs same thing which is all nodes in the tree. Their difference is BFS searches childs DFS searches child's childs.

c. Traverse Search
    i. Most of the case time complexity is O(V^2) because first of all we have to get all nodes from tree after that we check it is our target or not.

3. Solution Approach (Function)
    a. BFS Search
        i. BFS search algorithm includes to search child of parent. When we find the children we add all of them to linked list and we get first element of linked list and search their children and add all to linked list so on. The value of we get first element of linked list is equal to our target we returns true other wise we continue to search children if there is no children we return false.
    b. DFS Search
        i. Their algorithm so likely. But in DFS search when we find child we go to it is child until there is no child. If we find our target value in the children we return True otherwise return false.
    c. Traverse Search
        i. It is Reversed BFS search.
    d. Move Algorithm
        i. My aim is so simple. First of all, I want to sure that USER_GIVEN_FROM_PATH is true or not, if true I continue my solution, if not I gave a error message in terminal and all flows continue.
        ii. If USER_GIVEN_FROM_PATH is true, I update to TO path because if there is parent directory and that parent directory is not given in the TO path I update TO path. After updating to PATH, I go dir by dir because if there is no dir named before I will create it. In same time I delete FROM_PATH to if there is empty.