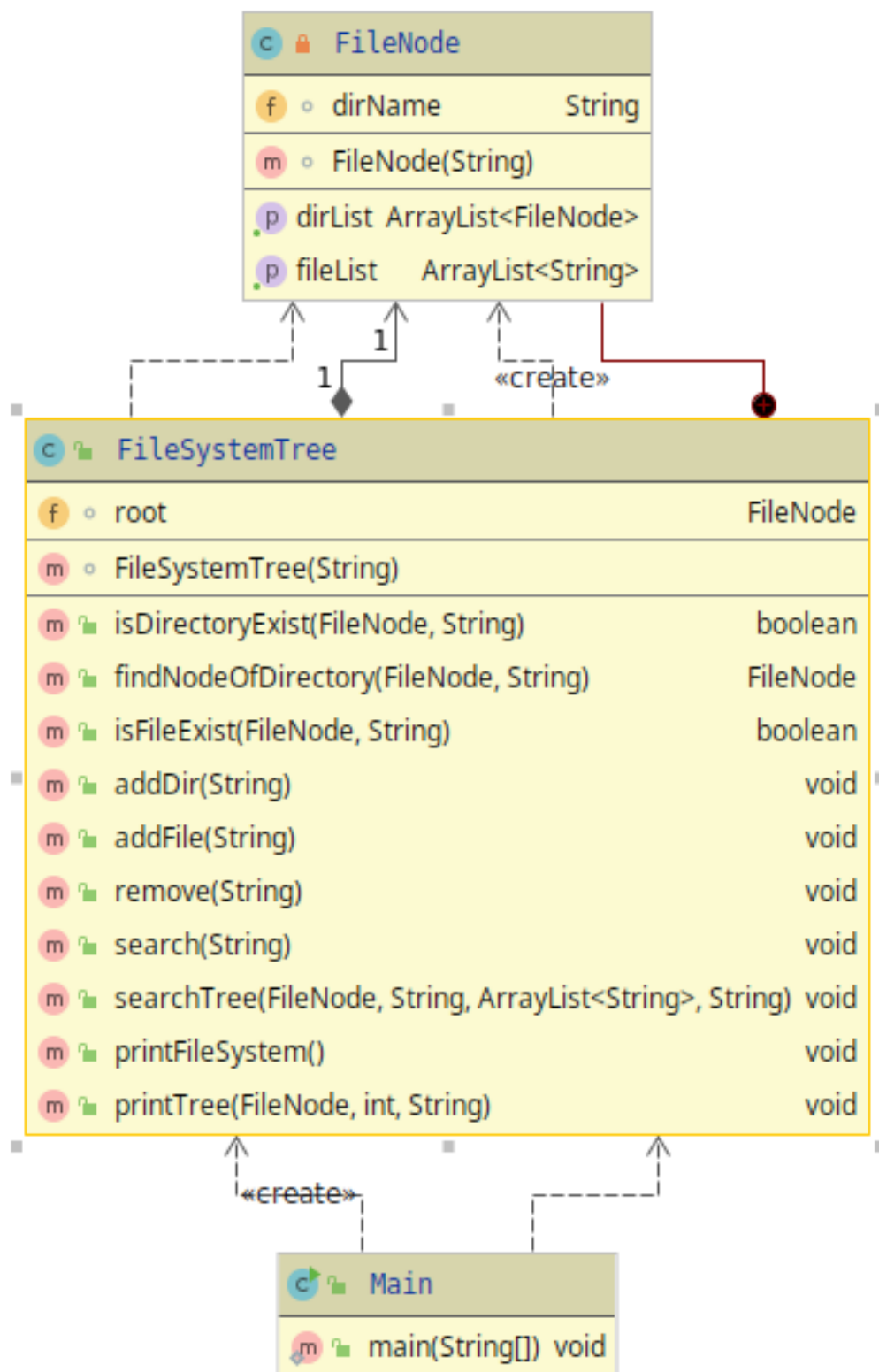


**GIT Department of Computer Engineering
CSE 222 – Spring 2020
Homework 5 Report**

**Esra Emirli
151044069**

Q1

1 Class Diagrams



2 Problem solutions approach

There may be directories or files under the directories in the system, so every directory should be kept as a node, but the files should be kept only as strings.

So into FileNode

- directoryList and fileList are added. For this directory name, dirName variable was added.

Of course, it first created a root with the string entered in the constructor to create the tree structure.

When the **addDir method** is called, it must first be split according to path “/” so that it can be entered in the nodes and registered in the right place.

After splitting, the last index is given the directory to be registered, while the others show which nodes should be entered. For this reason, the presence of directory names in the indices is checked every time and dirList is added when it is seen that the last element is absent.

The **addFile method** is the same as addDir, the only difference is adding to the fileList.

Path is taken again for the **remove method** and this path is split to “/”.

Since it is in add operations, the presence of directory names is checked. If it is the directory that you want to delete, go to that directory and see if there is any directory or file in it. If there is any, the screen is printed with printTree and the user is informed that they will be deleted. Approval is obtained.

If the file to be deleted is directly deleted.

The word given for the **search method** is searched. Since there is no path, the whole tree should be visited.

Therefore, the searchtree method is used to assist the specified search method. this method takes root and lists to keep results.

This method starts from root, navigates to the directorList registered in root one by one and does the same for every directory in it. It adds to path at every opportunity. This path is also sent as a parameter because it is recursive. If it finds the word, this path is saved. Path is reset after added to resultlist.

The **printFileSystem method** requires a renewed node each time to navigate the tree like search method. For this, it gets help from the printTree method.

In the printTree method, a space is used to understand which directory is under which directory.

3 Test cases

Test Case ID	Test Scenario	Test Steps	Test Data	Excepted Results	Actual Results	Pass/Fail
T01	if you want to register with wrong root, addDir	1.splits the incoming string. 2. its first statement must be root. So it compares the first expression with the root name	addDir("xx")	A root named 'xx' not valid!	As expected	Pass
T02	if you did not enter a directory name to add, addDir	1.splits the incoming string. 2. root name is valid 3. if the splited string length is less than 1, give an error	addDir("root")	example path: root / new_direName	As expected	Pass
T03	normal adding, addDir	Go to the last element in the splits string and add it to the dirList.	addDir("root/ first_dir")	-	As expected	Pass
T04	if you want to re-add the already existing directory name, addDir	Go to the last element in the splits string and checks dirList.	addDir("root/ first_dir")	A directory named "first_dir" already exists!	As expected	Pass
T05	if you want to register with wrong root, addFile	1.splits the incoming string. 2. its first statement must be root. So it compares the first expression with the root name	addFile("xx")	A root named 'xx' not valid!	As expected	Pass
T06	if you did not enter a file name to add, addFile	1.splits the incoming string. 2. root name is valid 3. if the splited string length is less than 1, give an error	addFile("root")	example path: root / new_fileName	As expected	Pass
T07	normal adding, addFile	Go to the last element in the splits string and add it to the fileList.	addFile("root/ first_file")	-	As expected	Pass
T08	if you want to	Go to the last	addFile("root/	A directory	As expected	Pass

	re-add the already existing file name, addFile	element in the splits string and checks fileList.	first_file")	named "first_file" already exists!		
T09	if the root is to be deleted, remove	1.splits the incoming string. 2. root name is valid 3. if the splitted string length is less than 2, give an error	remove("root")	example path: root / new_fileName	As expected	Pass
T10	If the invalid dirname is to be deleted, remove	Go to the last element in the splits string and checks dirList.	remove("root/invalid")	Directory or File named invalid not found!	As expected	Pass
T11	If the invalid filename is to be deleted, remove	Go to the last element in the splits string and checks fileList.	remove("root/invalidFile")	Directory or File named invalidFile not found!	As expected	Pass
T12	If the valid dirname is to be deleted, remove	Go to the last element in the splits string and checks dirList.	remove("root/first_dir")	-	As expected	Pass
T13	If the invalid filename is to be deleted, remove	Go to the last element in the splits string and checks fileList.	remove("root/first_file")	-	As expected	Pass
T14	If deleting dirName with files	List files in dirName and ask	remove("root/first_file")	These files / directories will also be deleted. Do you approve? Y or N	As expected	Pass
T15	Does the given word exist in all files and directories, search	With the help of the <i>searchTree</i> method, it saves all the paths that the given word goes through.	search("dir")	root/first_dir	As expected	Pass
T16	printFileSystem method to print the whole tree.	With the help of the <i>printTree</i> method, it prints files under all directories with appropriate tabs	printFileSyste()	root first_directory : ff1	As expected	Pass

4 Running command and results

- T01:



```
4
5 public static void main(String[] args) {
6     FileSystemTree myFileSystem = new FileSystemTree( key: "root");
7     myFileSystem.addDir( path: "xoot/first_directory");
8 }

Main > main()

Run: Main x
/usr/lib/jvm/jdk-13.0.2/bin/java -javaagent:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/lib/idea_rt.jar=45143:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/bin -jar /home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/bin/idea_rt.jar
A root named 'xoot' not valid!
```

- T02:




```
5 public static void main(String[] args) {
6     FileSystemTree myFileSystem = new FileSystemTree( key: "root");
7     myFileSystem.addDir( path: "root");
8     System.out.println("\n\n");
9     myFileSystem.addDir( path: "root/first_directory");
10 }

Main > main()

Run: Main x
/usr/lib/jvm/jdk-13.0.2/bin/java -javaagent:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/lib/idea_rt.jar=37725:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/bin -jar /home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/bin/idea_rt.jar
example path: root/new_dirName
```

- T03 - T04:

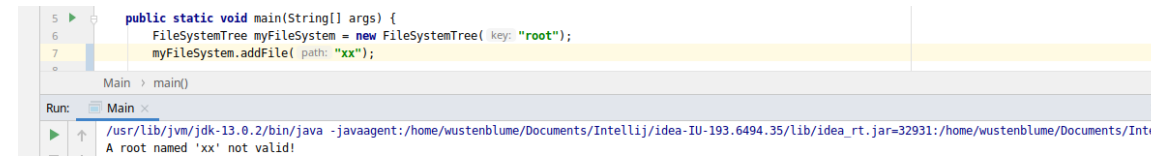


```
5 public static void main(String[] args) {
6     FileSystemTree myFileSystem = new FileSystemTree( key: "root");
7     myFileSystem.addDir( path: "root/first_directory");
8     myFileSystem.addDir( path: "root/first_directory");
9 }

Main > main()

Run: Main x
/usr/lib/jvm/jdk-13.0.2/bin/java -javaagent:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/lib/idea_rt.jar=35311:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/bin -jar /home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/bin/idea_rt.jar
A directory named first_directory already exists!
```

- T05 :




```
5 public static void main(String[] args) {
6     FileSystemTree myFileSystem = new FileSystemTree( key: "root");
7     myFileSystem.addFile( path: "xx");
8 }

Main > main()

Run: Main x
/usr/lib/jvm/jdk-13.0.2/bin/java -javaagent:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/lib/idea_rt.jar=32931:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/bin -jar /home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/bin/idea_rt.jar
A root named 'xx' not valid!
```

- T06 :



```
4
5 public static void main(String[] args) {
6     FileSystemTree myFileSystem = new FileSystemTree( key: "root");
7     myFileSystem.addFile( path: "root");
8 }

Main > main()

Run: Main x
/usr/lib/jvm/jdk-13.0.2/bin/java -javaagent:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/lib/idea_rt.jar=36065:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/bin -jar /home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/bin/idea_rt.jar
example path: root/new_fileName
```

- T07 – T08 :



```
5 public static void main(String[] args) {
6     FileSystemTree myFileSystem = new FileSystemTree( key: "root");
7     myFileSystem.addFile( path: "root/first_file1");
8     myFileSystem.addFile( path: "root/first_file1");
9 }

Main > main()

Run: Main x
/usr/lib/jvm/jdk-13.0.2/bin/java -javaagent:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/lib/idea_rt.jar=34563:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/bin -jar /home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/bin/idea_rt.jar
A file named first_file1 already exists!
```

- T09:



```
5 public static void main(String[] args) {
6     FileSystemTree myFileSystem = new FileSystemTree( key: "root");
7     myFileSystem.remove( path: "root");
8 }

Main > main()

Run: Main x
/usr/lib/jvm/jdk-13.0.2/bin/java -javaagent:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/lib/idea_rt.jar=39967:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/bin -jar /home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/bin/idea_rt.jar
example path: root/name
```

- T10 - T11:



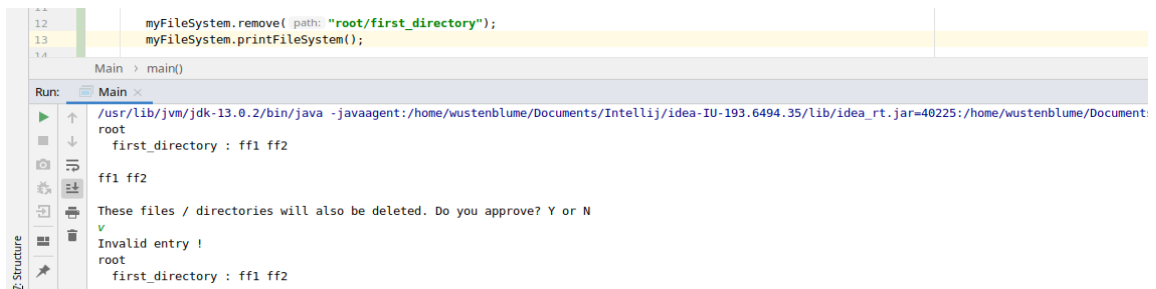
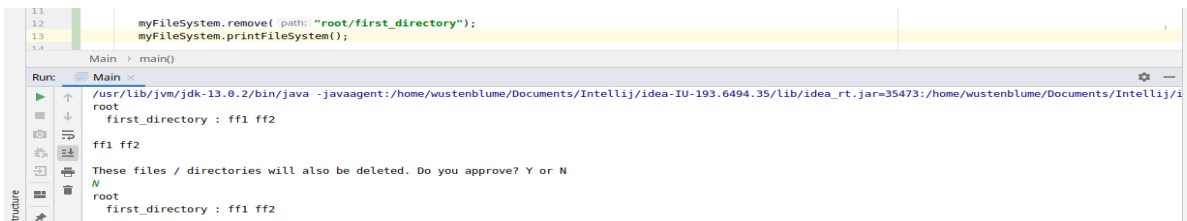
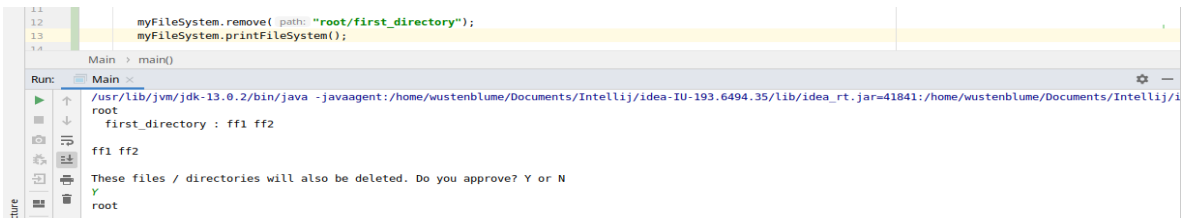
- T12 :



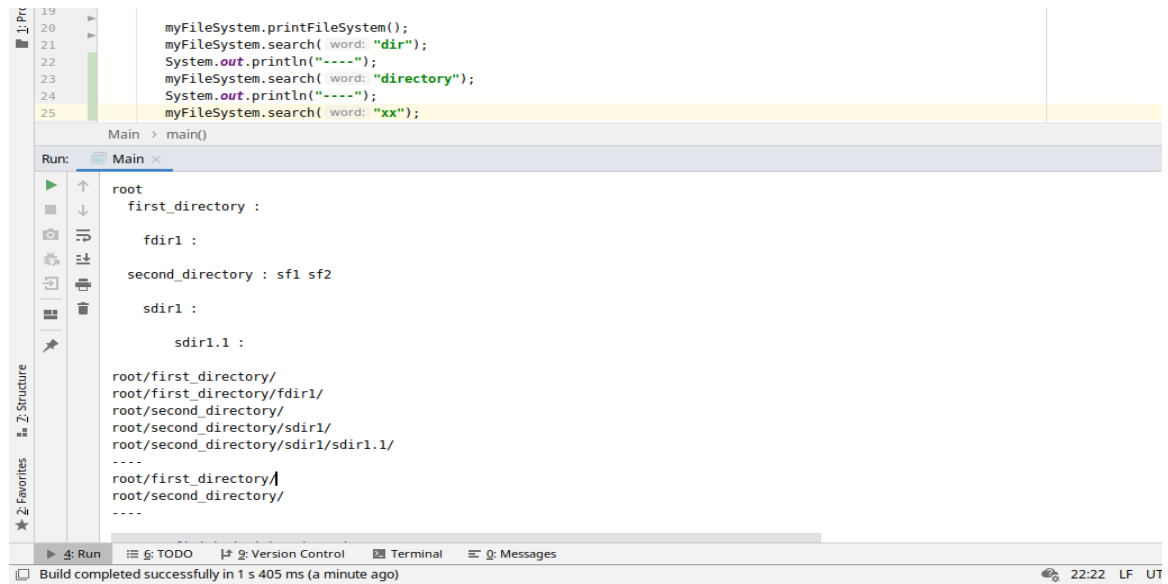
- T13 :



- T14:

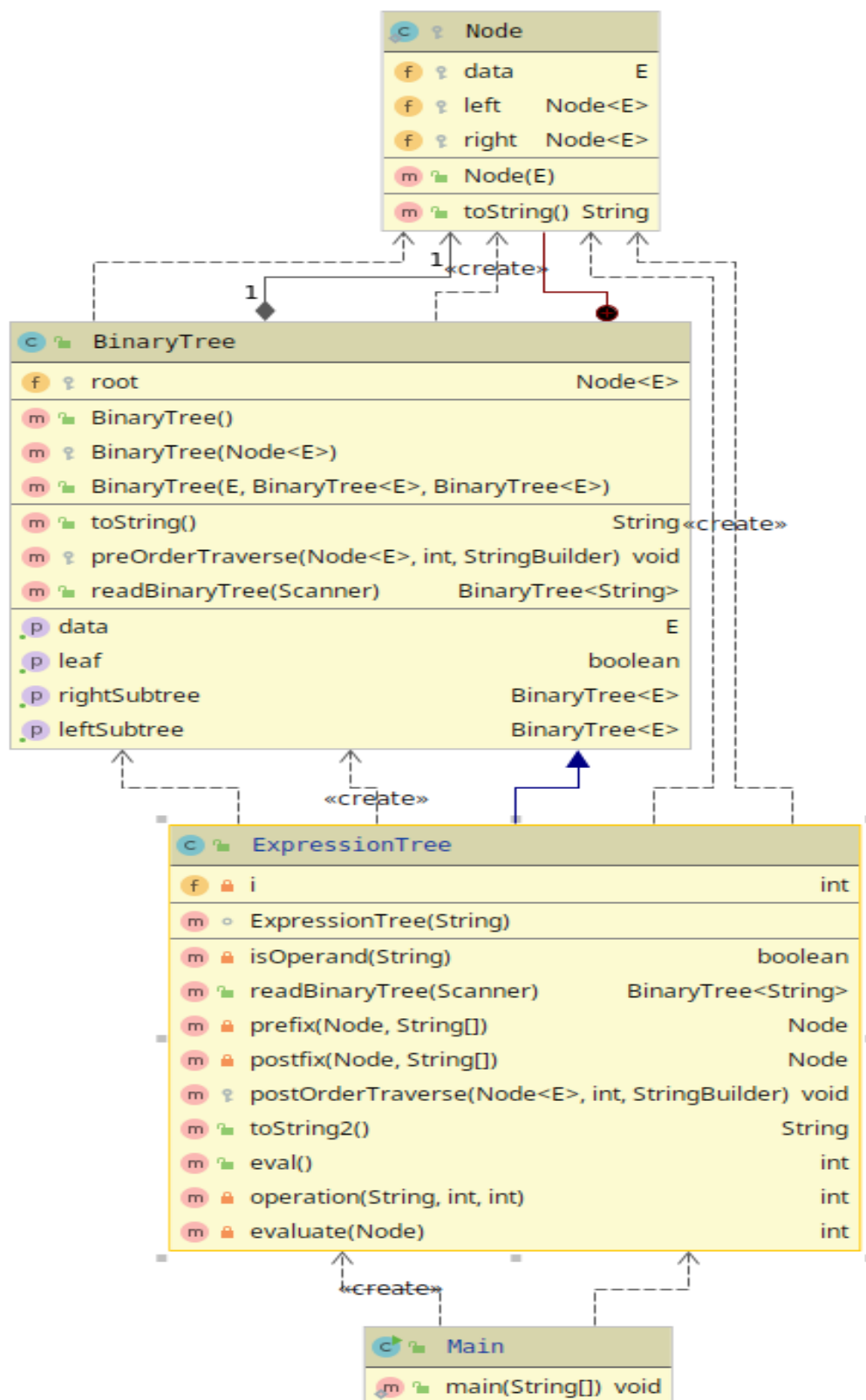


- T15 - T16:



Q2

1 Class Diagrams



2 Problem solutions approach

According to the statement given in the construct, a tree structure will be created through **readBinaryTree method**.

But it is not known whether this expression will be postfix or prefix.

For this, by sending a statement to readBinaryTree and split there according to " ".

If the first index of the string obtained as a result of the split is operand, this means that it is prefix, if the last index is an opera, this means that postfix is an expression. If not both, this statement is invalid.

- ✓ In the general structure of the tree, numerical values will be located to the left and right of the operands. The left and right of the numerical value will be null.

If the expression is prefix, the expression and root are sent to the prefix method. In prefix, it has priority over left and right. Therefore, first check whether the left is empty and if the expression is operand, a new node is created and the function is called again. The expression is saved directly to the left if it is not operand. Nodes with a return value will be saved as a left tree.

After the left tree is completed, the same procedures are performed for the right side.

If the expression is postfix, the operations done in prefix are the same in postfix method, only the order is different. postfix gives priority to the right. Then it goes to the left.

Thus, the tree structure is created.

postorder methods is traversed with the postorder (Left, Right, Root) rule. It moves to the left and then the right nodes to be the last root.

The **toString2 method** prints the string returned by the postorder method.

For the **eval method**, it should be calculated by calculating from the bottom of the tree to the top.

For this, recursive is used. A evaluate method that takes root is defined. In this method, calculations start from the right tree. The evaluation value of the right tree and the evaluation value of the left tree are finally done according to the operand in root and the result is returned.

3 Test cases

Test Case ID	Test Scenario	Test Steps	Test Data	Excepted Results	Actual Results	Pass/Fail
T01	when an expression is entered in the constructor	sends this to readBinaryTree	new ExpressionTree(".")	-	-	Pass

T02	If the last index of the split expression is operand, readBinaryTree	1.create root with the last index. Send the expression with root to the postfix method. 2. return the root from postfix to the root of the new binary tree and return the tree	ReadBinaryTree (Scanner scan)	Binary tree	As expected	Pass
	postfix	1.if the right of the root is null, 1.1 If the expression is operand, create a new node and send it to postfix as root. 1.2 if not, save it to the right as a node and continue 2.if the left of the root is null, 2.1 If the expression is operand, create a new node and send it to postfix as root. 2.2 if not, save it to the left as a node and continue	Postfix(root , expression)	Root of tree	As expected	Pass
T03	If the first index of the split expression is operand, readBinaryTree	1.create root with the first index. Send the expression with root to the postfix method. 2. return the root from prefix to the root of the new binary tree and return the tree	ReadBinaryTree (Scanner scan)	Binary tree	As expected	Pass
	prefix	1.if the right of the root is null, 1.1 If the expression is operand, create a new node and send it to postfix as root. 1.2 if not, save it to the right as a node and continue 2.if the left of the root is null, 2.1 If the expression is operand, create a new node and send it to postfix as root. 2.2 if not, save it to the left as a node and continue	Prefix(root , expression)	Root of tree	As expected	Pass
T04	if split expression is not both operand,	Return null	ReadBinaryTree (Scanner scan)	Exception ("Undefined Expression	As expected	Pass

	readBinaryTree			! ")		
T05	postOrderTraverse	1.postOrderTraverse method to traverse the tree post order. 2. append the result to stringBuilder	PostOrderTraverse (root, deph, strignbuilder)	return value void, the value saved in stringBuilder with reference is taken.	As expected	Pass
T06	toString2	1. which will create a string of the tree structure in post order using postOrderTraverse		String	As expected	Pass
T07	eval	1.Calls a new method whose parameter is root.		int	As expected	Pass
	evaluate	1.It performs all operations by going to the bottom of the nodes.	evaluate(root)	Int	As expected	Pass

4 Running command and results

- T01:

```

11 public static void main(String[] args) throws IOException {
12
13     //Create a tree for preorder expression
14     ExpressionTree expTree = new ExpressionTree("+ + 10 * 5 15 20");
15 }

```

- T02:

```

13 //Create a tree for preorder expression
14 ExpressionTree expTree = new ExpressionTree("10 5 15 * + 20 +");
15 System.out.println(expTree.toString());
16

```

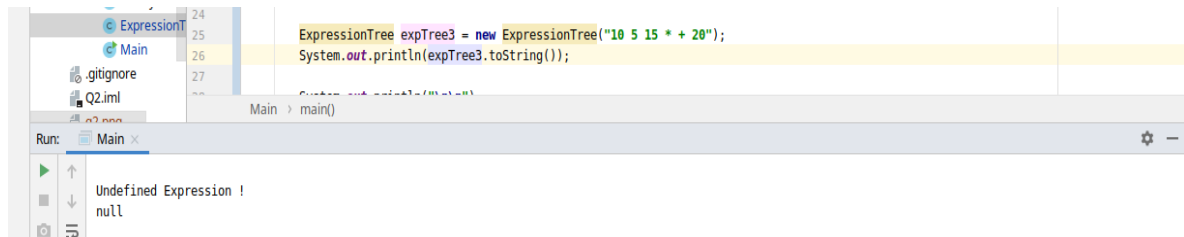
- T03 :

```

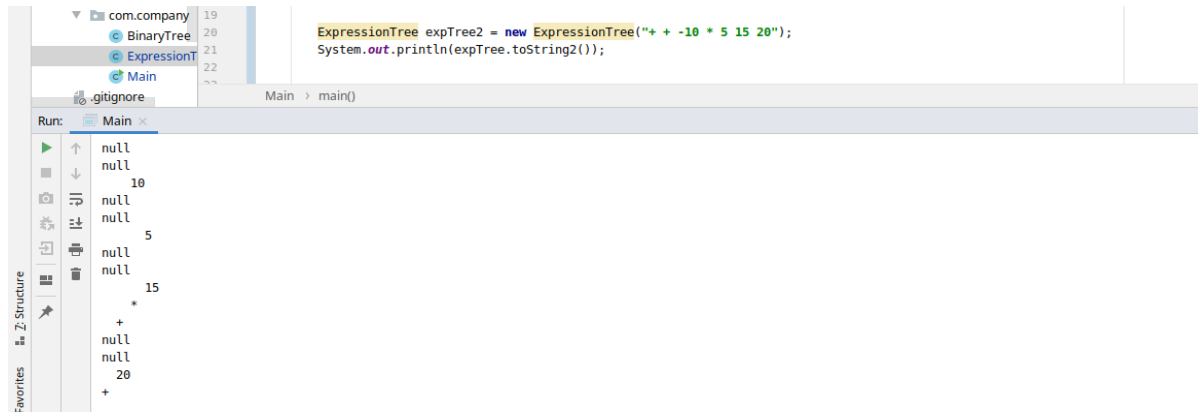
19 ExpressionTree expTree2 = new ExpressionTree("+ + -10 * 5 15 20");
20 System.out.println(expTree2.toString());
21
22

```

- T04:



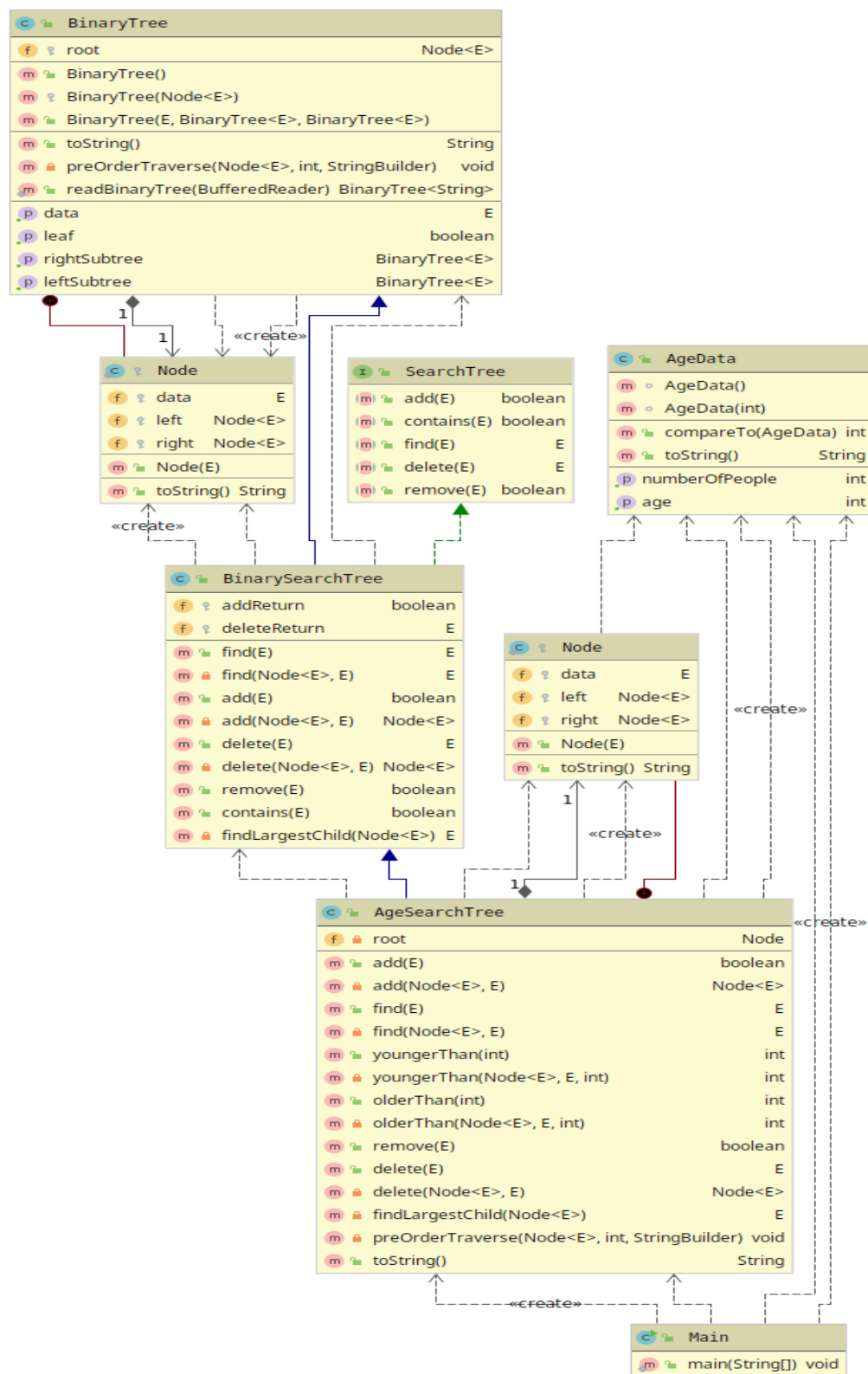
- T05 - T06:



- T07:



1 Class Diagrams



2 Problem solutions approach

Each node in AgeSearchTree must be filled with AgeData. AgeData keeps track of age and how many people have been recorded with this age.

The **add method** is to see if there is a person at that age. If there is a person at that age, a new registration should not be made, and the numberOfPeople value in the node of that age should be increased.

An auxiliary **add (root, item) method** is used for this add method. It is checked whether the given root is empty or not, whether the age to be registered is larger or smaller than the root's data. If root is empty, a new node is created and this node is assigned to the root of the tree. If the tree is full, the comparison process is performed and according to the comparison, a recursive look will be found on the right or left side of the tree. If the data is mapped, this means that the age exists before and numberOfPeople is increased.

The **remove method** is also considered the same as the add method. While looking for age, with root.data, it decides whether to look to the right or left of the tree according to the size and size. If the age sought is found as a result of comparison, numberOfPeople is reduced, if this reduction is 0, it is deleted.

The **find method** searches like any other and returns the node of the age it finds. **toString method** returns string in the form of node's age – numberOfPeople

The same searches are made in the **youngerThan method**. In the first node comparison where the given age is great, all the values of that node are visited and additions are made by looking at the numberOfPeople.

The same calls are made in the **olderThan method**. In the first node comparison where the given age is lower, all the values of that node are visited and additions are made by looking at the numberOfPeople.

3 Test cases

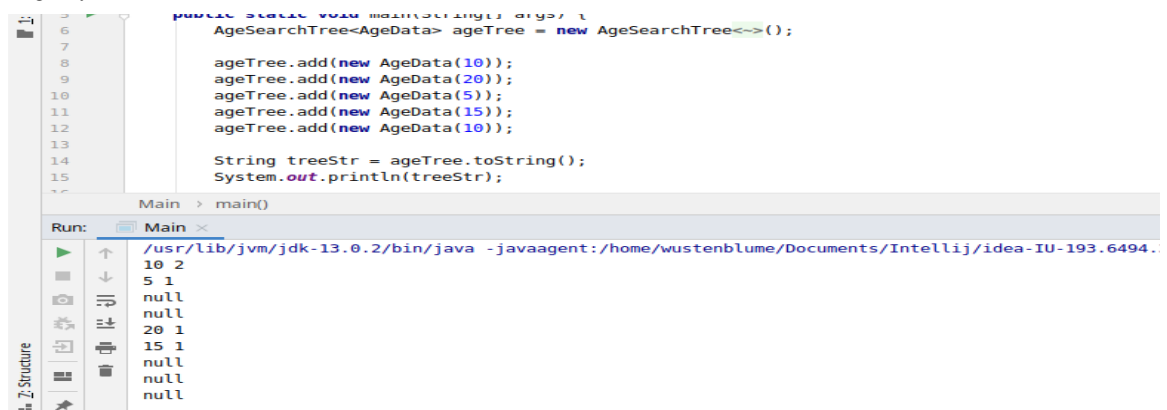
Test Case ID	Test Scenario	Test Steps	Test Data	Excepted Results	Actual Results	Pass/Fail
T01	When add is done by giving ageData the valid age.	1. Take item (ageData object) 2. Send the item to private add function with root 3. check root is null 4. item's age less than root.data 5. item's age bigger than root.data 6. item's age is equal root.data	New AgeData(10)	10 1 null null	As expected	Pass

T02	When add is done by giving ageData the invalid age.	1. Take item (ageData object)	new AgeData(-5)	Exception (“Age can not be less than 0 !”)	As Expected	Pass
T03	remove multiple saved ages	1. Take item (ageData object) 2. Send the item to public delete function 3.delete function send the item to private delete function with 4. check root is null 5. item’s age less than root.data 6. item’s age bigger than root.data 7. item’s age is equal root.data 8. reduce the number of registered at this age.	New AgeData(10)	10 1 5 1 null null	As Expected	Pass
T04	remove the age once saved	1. Take item (ageData object) 2. Send the item to public delete function 3.delete function send the item to private delete function with 4. check root is null 5. item’s age less than root.data 6. item’s age bigger than root.data 7. item’s age is equal root.data 8. remove node	New AgeData(10)	5 1 null	As Expected	Pass
T05	find the recorded data	1. Take item (ageData object) 2. Send the item to private find function with root 3. check root is null 4. item’s age less than root.data 5. item’s age bigger than root.data 6. item’s age is equal root.data return root.data 7. return value is not null	New AgeData(10)	10 - 2	As Expected	Pass
T06	find the unrecorded data	1. Take item (ageData object) 2. Send the item to private find function with root 3. check root is null 4. item’s age less than root.data 5. item’s age bigger	New AgeData(100)	Exception (“This age could not be found !”) -1 - 0	As Expected	Pass

		than root.data 6. item's age is equal root.data return root.data 7. return value is null				
T07	youngerThan	1. Take item (ageData object) 2. Send the item to private youngerThan function with root and count that default 0 3. check root is null 4. item's age less than root.data 5. item's age bigger than root.data, increase the value as much as the number that age. 6. item's age is equal root.data 7. return count	15	3	As Expected	Pass
T06	olderThan	1. Take item (ageData object) 2. Send the item to private youngerThan function with root and count that default 0 3. check root is null 4. item's age less than root.data, increase the value as much as the number that age. 5. item's age bigger than root.data 6. item's age is equal root.data 7. return count	15	1	As Expected	Pass

4 Running command and results

- T01 :



```

public static void main(String[] args) {
    AgeSearchTree<AgeData> ageTree = new AgeSearchTree<>();

    ageTree.add(new AgeData(10));
    ageTree.add(new AgeData(20));
    ageTree.add(new AgeData(5));
    ageTree.add(new AgeData(15));
    ageTree.add(new AgeData(10));

    String treeStr = ageTree.toString();
    System.out.println(treeStr);
}
  
```

Run: Main

```

/usr/lib/jvm/jdk-13.0.2/bin/java -javaagent:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494..
10 2
5 1
null
null
20 1
15 1
null
null
null
  
```

- T02 :

```

6      public static void main(String[] args) {
7          AgeSearchTree<AgeData> ageTree = new AgeSearchTree<>();
8
9          ageTree.add(new AgeData(-5));
10         System.out.println(ageTree.toString());
    
```

Run: Main

```

/usr/lib/jvm/jdk-13.0.2/bin/java -javaagent:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/lib/idea_rt.jar=44551:/home/wust
Age cannot be less than 0 !
0 0
null
null
    
```

- T03:

```

16
17         ageTree.remove(new AgeData(10));
18         treeStr = ageTree.toString();
19         System.out.println(treeStr);
    
```

Run: Main

```

/usr/lib/jvm/jdk-13.0.2/bin/java -javaagent:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/lib/idea_rt.jar=33759:/home/wustenblume/
10 2
5 1
null
null
20 1
15 1
null
null
null
....
10 1
5 1
null
null
20 1
15 1
null
null
null
.
    
```

- T04 :

```

21         ageTree.remove(new AgeData(10));
22         treeStr = ageTree.toString();
23         System.out.println(treeStr);
24
    
```

Run: Main

```

10 1
5 1
null
null
20 1
15 1
null
null
null
....
5 1
null
20 1
15 1
null
null
null
    
```

- T05:

```

17         //Find the number of people at any age
18         System.out.println(ageTree.find(new AgeData(10)).toString());
19
    
```

Run: Main

```

/usr/lib/jvm/jdk-13.0.2/bin/java -javaagent:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/lib/idea_rt.jar=37101:/home/wustenblume/Documents/Int
10 2
5 1
null
null
20 1
15 1
null
null
null
....
10 - 2
    
```

- T06:

```

17 //Find the number of people at any age
18 System.out.println(ageTree.find(new AgeData(100)).toString());
19
Main > main()
Run: Main x
/usr/lib/jvm/jdk-13.0.2/bin/java -javaagent:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/lib/idea_rt.jar=45497:/home/wustenblum
10 2
5 1
null
null
20 1
15 1
null
null
null
----
This age could not be found !
-1 - 0

```

- T07:

```

16
17 //Print the number of people younger than 15
18 System.out.println("youngerThan : " + ageTree.youngerThan( target: 15));
Main > main()
Run: Main x
/usr/lib/jvm/jdk-13.0.2/bin/java -javaagent:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/lib/idea_rt.jar=33603:/home/wustenblume/Doc
10 2
5 1
null
null
20 1
15 1
null
null
null
----
youngerThan : 3

```

- T08:

```

17 System.out.println("olderThan : " + ageTree.olderThan( target: 15));
18
Main > main()
Run: Main x
/usr/lib/jvm/jdk-13.0.2/bin/java -javaagent:/home/wustenblume/Documents/IntelliJ/idea-IU-193.6494.35/lib/idea_rt.jar=42911:/home/wustenblume/Document
10 2
5 1
null
null
20 1
15 1
null
null
null
----
olderThan : 1

```