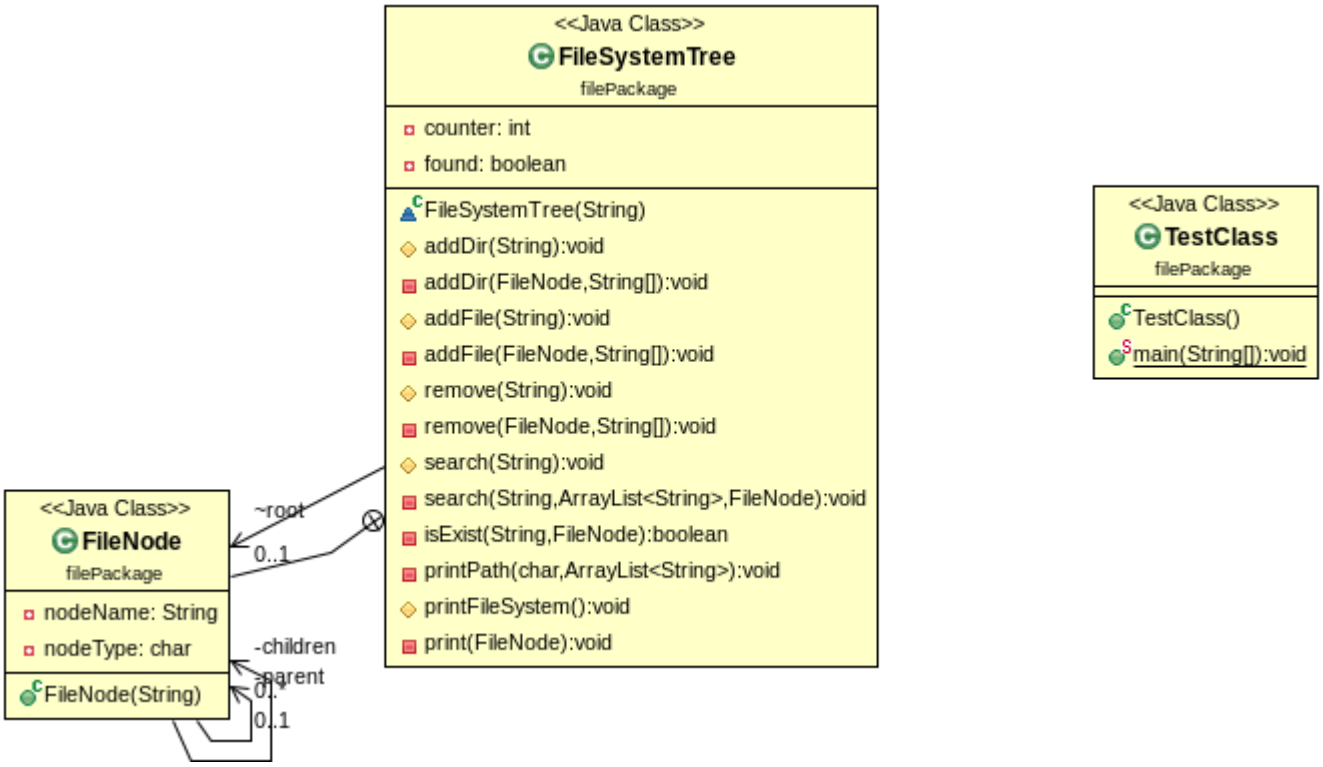
DATA STRUCTURE AND ALGORITHMS
HW05

MOHAMMAD ASHRAF YAWAR
161044123

PART01

CLASS DIAGRAM:



PROBLEM SOLUTION APPROACH:

- we are asked to implement a general tree structure in java:

I have a main class called FileSystemTree and this class is entire tree actually
each file or directory in filesystem tree is represendted as FileNode class which is a nested class of FileSystemTree class with private signature.

I represented the relationships between a parent node (which in this case is directory type input) will have several children which simply represented as ArrayList data structure.

Each node in arraylist data strudture indicates a child node of a particular parent node where the child nodes can eighther be a file or directory.a FileNode class represents a file or a directory accordigly and each node or directoty keeps it's related sub directories as a list and the name of the node and type of it (if file then nodetype = 'f' if directory then nodetype='d').

I used recursion in each function :
>> addDir()
>>addFile()
>>remove()
>>search()
>>some private function

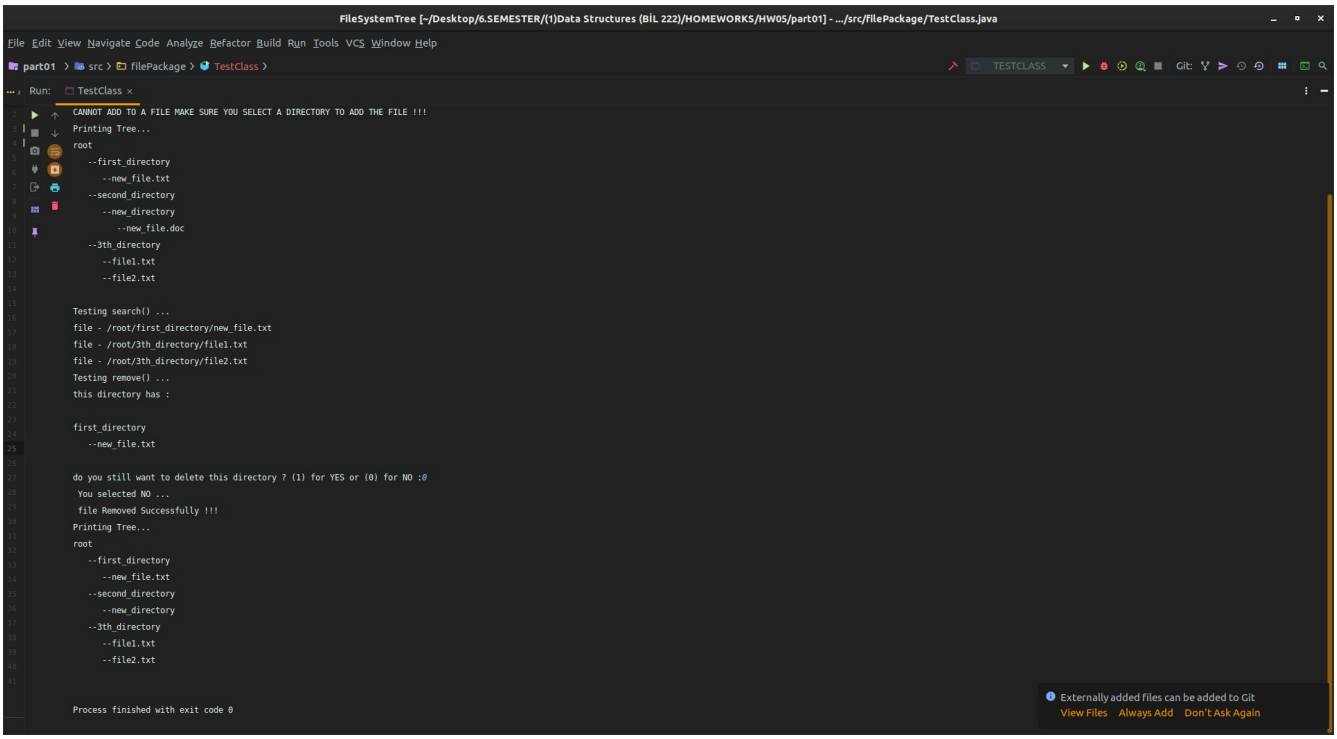
all the FileSystemTree class methos are protected so that can only be access through same packages(excluding some private function which area only used inside the FileSystemTree class).
I used ArrayList data structure to maintain easiness an accessibility.

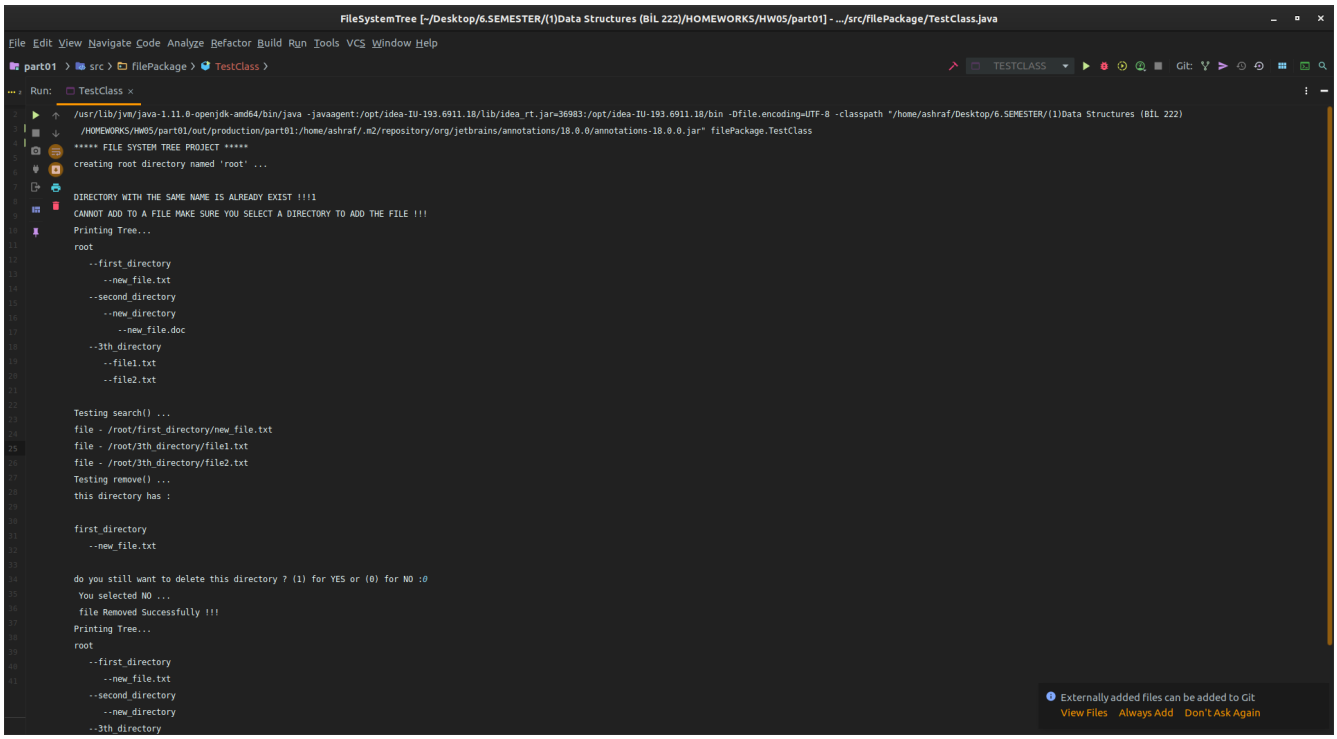
TEST CASES:

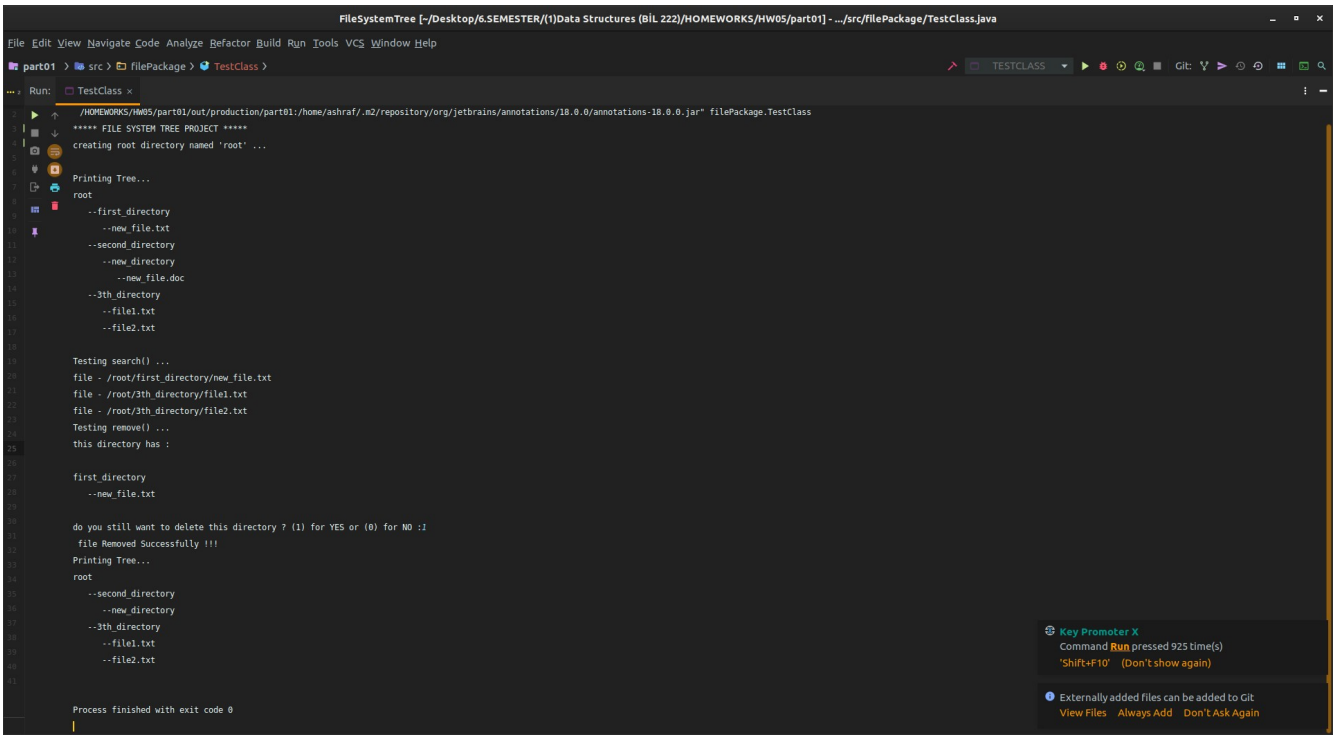
Test Case ID	Test Scenarios	Test Data	Expected Results	Actual Result	Pass/Fail
T01	Add directory	addDir(root/ first_directory)	true	true	Pass
T02	Add file	addFile(root/ first_directory/ new_file.txt)	true	true	Pass
T03	Adding directory into a txt file	Add File(root/first_directory/new_file.txt/ dir)	false	false	Pass
T04	Printing general tree	printFileSystem()	Should print general tree	Prints as expected	Pass
T05	Searching for a key String in generla tree	search("txt")	Should list all the nodes which contains txt string in it	Lists as expected	Pass
T06	Searching for a key String in generla tree	search("---")	Should print blank	Prints empty list	Pass

T07	Removing from tree	remove(“root/ first_directory”)	It should ask for user if to delete the first_directory or not because first_directory has some sub directories in it	As expected	Pass
T08	Removing from tree	remove("root/ second_directory/ new_directory/ new_file.doc")	Is should delete new_file.doc from general tree.	As expected	Pass

COMMAND LINE SCREEN SHOTS:

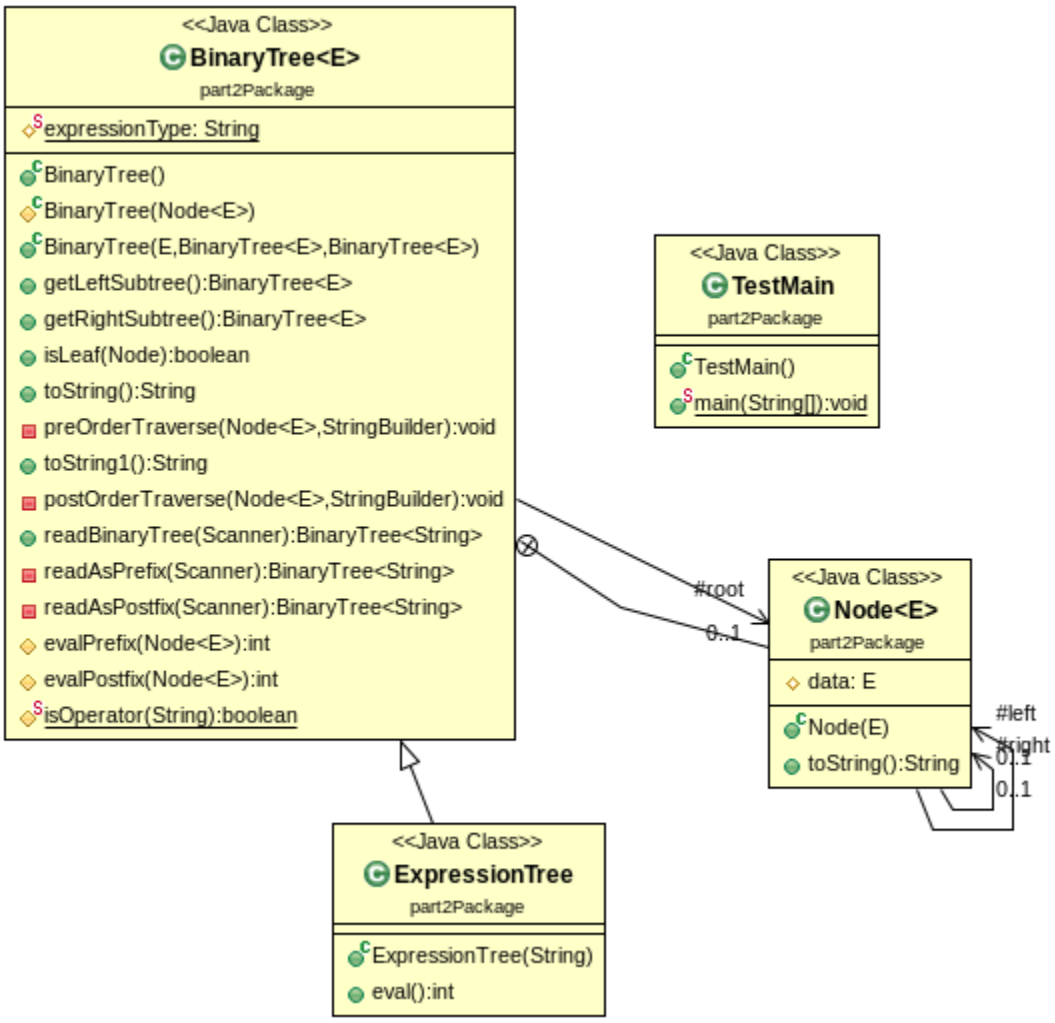



FileSystemTree [~/Desktop/6.SEMESTER/(1)Data Structures (BİL 222)/HOMEWORKS/HW05/part01] - .../src/filePackage/TestClass.java
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
part01 › src › filePackage › TestClass ›
TESTCLASS
Git:
Run: TestClass ×
/usr/lib/jvm/java-1.11.0-openjdk-amd64/bin/java -javaagent:/opt/idea-IU-193.6911.18/lib/idea_rt.jar=36983:/opt/idea-IU-193.6911.18/bin -Dfile.encoding=UTF-8 -classpath "/home/ashraf/Desktop/6.SEMESTER/(1)Data Structures (BİL 222)
/HOMEWORKS/HW05/part01/out/production/part01:/home/ashraf/.m2/repository/org/jetbrains/annotations/18.0.0/annotations-18.0.0.jar" filePackage.TestClass
***** FILE SYSTEM TREE PROJECT *****
creating root directory named 'root' ...
DIRECTORY WITH THE SAME NAME IS ALREADY EXIST !!!1
CANNOT ADD TO A FILE MAKE SURE YOU SELECT A DIRECTORY TO ADD THE FILE !!!
Printing Tree...
root
--first_directory
--new_file.txt
--second_directory
--new_directory
--new_file.doc
--3th_directory
--file1.txt
--file2.txt
Testing search() ...
file - /root/first_directory/new_file.txt
file - /root/3th_directory/file1.txt
file - /root/3th_directory/file2.txt
Testing remove() ...
this directory has :
first_directory
--new_file.txt
do you still want to delete this directory ? (1) for YES or (0) for NO :0
You selected NO ...
file Removed Successfully !!!
Printing Tree...
root
--first_directory
--new_file.txt
--second_directory
--new_directory
--3th_directory
Externally added files can be added to Git
View Files Always Add Don't Ask Again


FileSystemTree [~/Desktop/6.SEMESTER/(1)Data Structures (BİL 222)/HOMEWORKS/HW05/part01] - .../src/filePackage/TestClass.java
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
part01 › src › filePackage › TestClass ›
TESTCLASS
Git:
Run: TestClass ×
/HOMEWORKS/HW05/part01/out/production/part01:/home/ashraf/.m2/repository/org/jetbrains/annotations/18.0.0/annotations-18.0.0.jar" filePackage.TestClass
***** FILE SYSTEM TREE PROJECT *****
creating root directory named 'root' ...
Printing Tree...
root
--first_directory
--new_file.txt
--second_directory
--new_directory
--new_file.doc
--3th_directory
--file1.txt
--file2.txt
Testing search() ...
file - /root/first_directory/new_file.txt
file - /root/3th_directory/file1.txt
file - /root/3th_directory/file2.txt
Testing remove() ...
this directory has :
first_directory
--new_file.txt
do you still want to delete this directory ? (1) for YES or (0) for NO :1
file Removed Successfully !!!
Printing Tree...
root
--second_directory
--new_directory
--3th_directory
--file1.txt
--file2.txt
Process finished with exit code 0
Key Promoter X
Command Run pressed 925 time(s)
'Shift+F10' (Don't show again)
Externally added files can be added to Git
View Files Always Add Don't Ask Again

PART02

CLASS DIAGRAM:

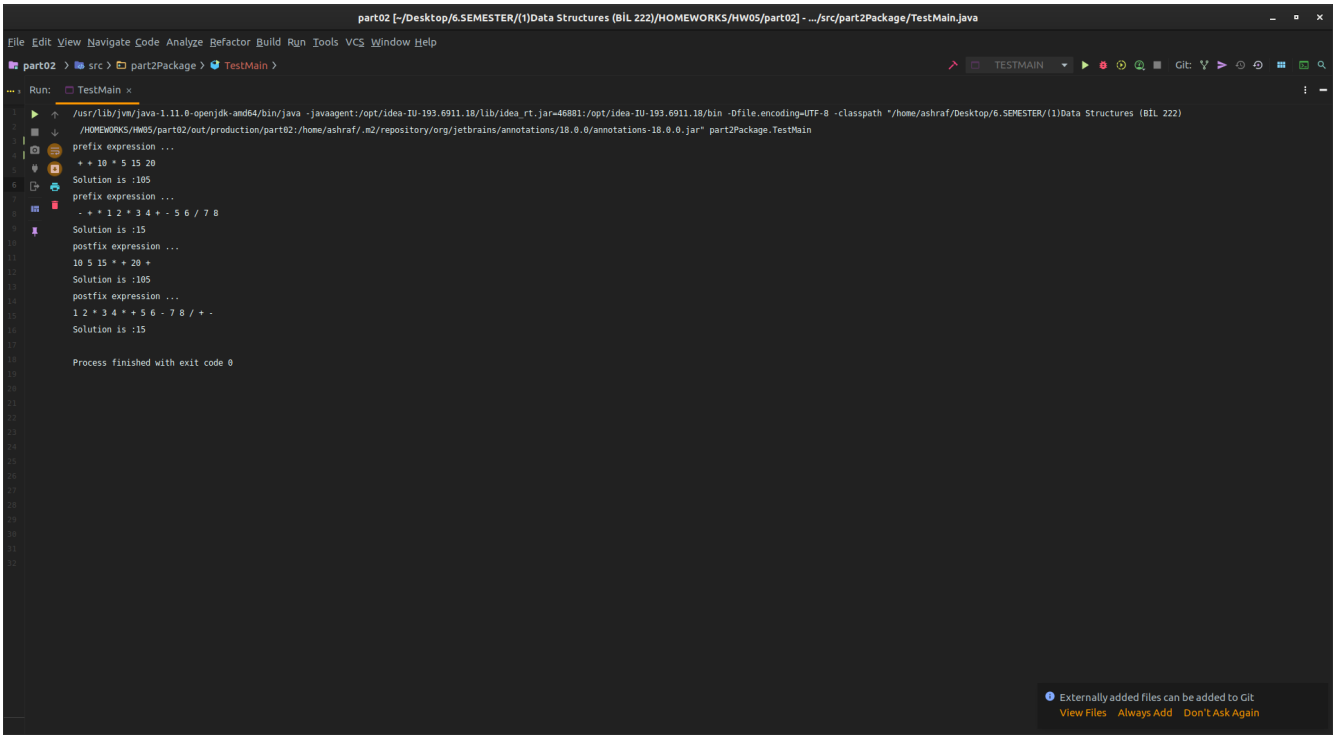


PROBLEM SOLUTION APPROACH:

in this part of project I have used some wrapper functions in order to distinguesh bitween the prefix expression and its evaulations as well as postfix expression and its evaulation.

Since we extended BinatyTree class in ExpressionTree class we can use BinaryTree class and overrides its methods as needed.
I overrided readBinaryTree class by makings it work for both prefix and postfix expressions and construct the certain expression tree according to expression type.
Every necessary methods in each classes has it’s own related exception handling part as needed.
I kept root node in BinaryTree class and access it as needed from ExpressionTree class since ExpressionTree class extends BinaryTree class.
Each part of this project has javaDoc related folders.

COMMAND LINE SCREEN SHOTS:

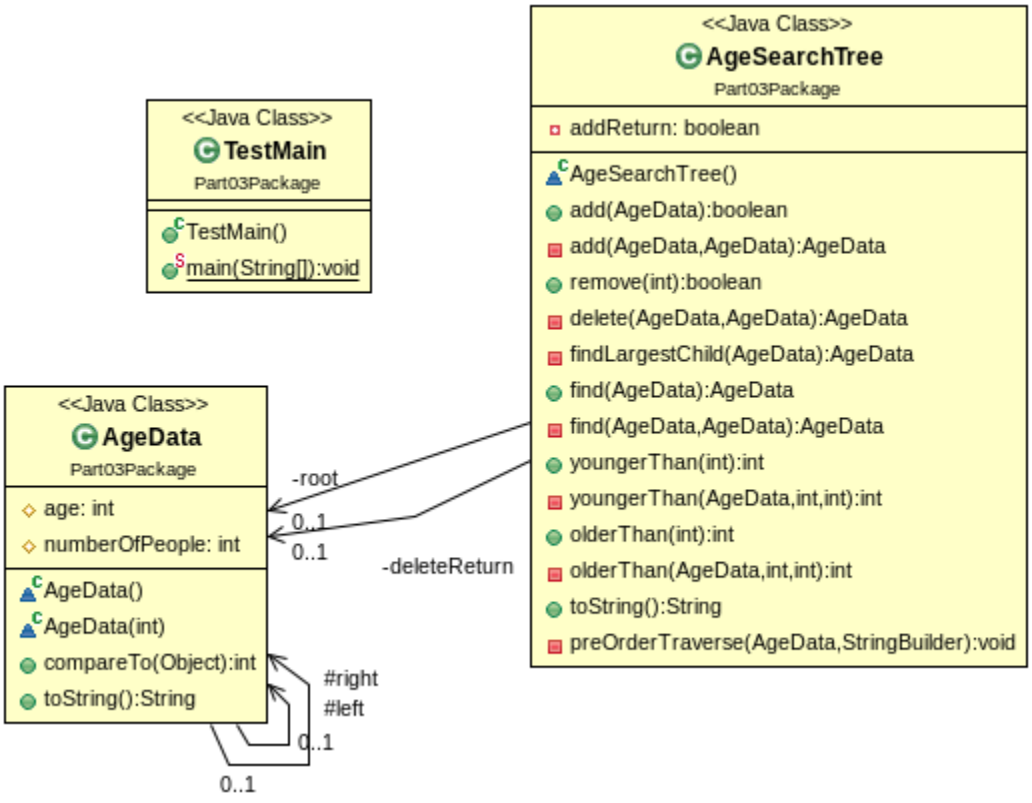


TEST CASES:

Test Case ID	Test Scenarios	Test Data	Expected Results	Actual Result	Pass/Fail
T01	Construct Expression tree	expTree = new ExpressionTree("+ + 10 * 5 15 20");	It should construct a binary expression tree in prefix manner	As expected	Pass
T02	Construct Expression tree	expTree1 = new ExpressionTree("1 0 5 15 * + 20 +")	It should construct a binary expression tree in postfix manner	As expected	Pass
T03	Print tree using toString()	exptTree.soString( )	It should print expression tree in preorder manner	As exprected	Pass
T04	Print tree using toString1()	exptTree1 .soStrin g1()	It should print expression tree in postorder manner	As exprected	Pass
T05	Evaluate preorder expression tree	ExpTree.eval()	75	75	Pass
T06	Evaluate postorder expression tree	ExpTree1.eval()	75		Pass

PART03

CLASS DIAGRAM:



PROBLEM SOLUTION APPROACH:

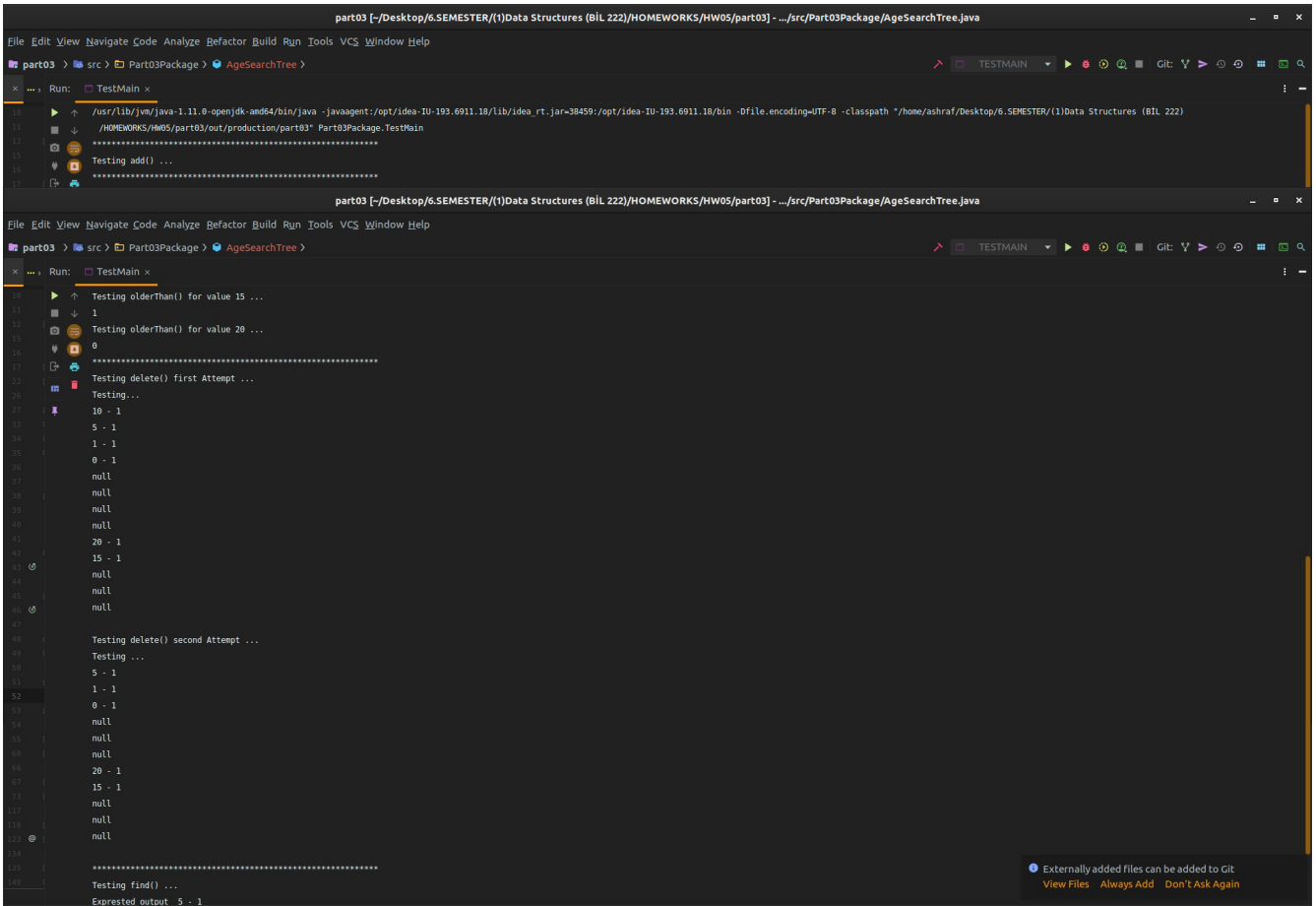
– I have used and override methods from BinarySearchTree class which are:

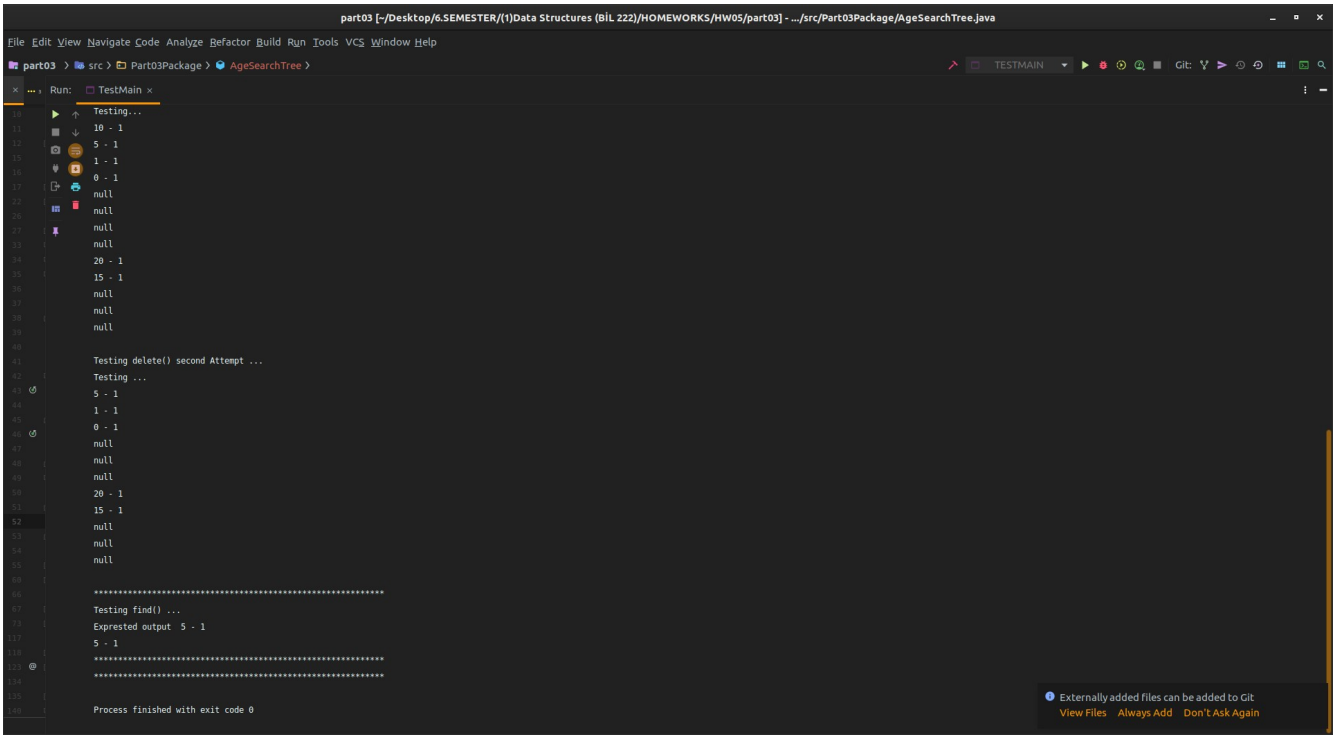
add()
delete()
toString()
find()
findLargestChild()

– and more addintional functions :
youngerThan()
olderThan()
preOrderTraverse()

– each function has it’s own wrapper function to traverse AgeSearchTree recursively .
– AgeSearchTree class uses AgeData class as elements in a BinarySearchTree.
– function who are used internally has private access modifies .

COMMAND LINE SCREEN SHOTS:





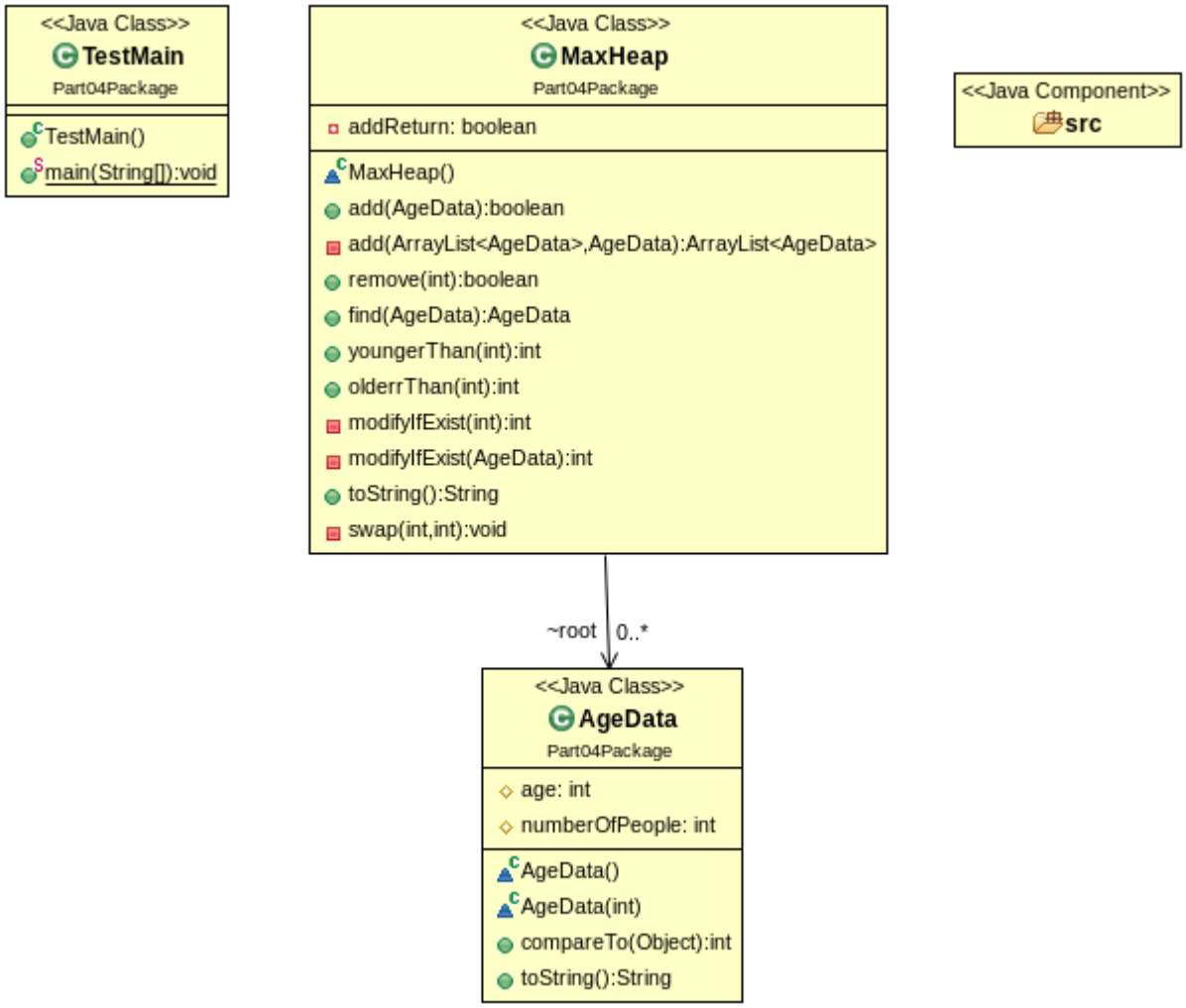
TEST CASES:

Test Case ID	Test Scenarios	Test Data	Expected Results	Actual Result	Pass/Fail
T01	Add an age to BinarySearchTree	ageTree.add(new AgeData(10))	It should as 10 to tree	As expected	Pass
T02	Add an age to BinarySearchTree	ageTree.add(new AgeData(20))	It should as 20 to tree	As expected	Pass
T03	Add an age to BinarySearchTree	ageTree.add(new AgeData(20))	It should as 20 to tree	As expected	Pass
T04	Add an age to BinarySearchTree	ageTree.add(new AgeData(20))	It should as 20 to tree	As expected	Pass
T05	Add an age to BinarySearchTree	ageTree.add(new AgeData(10))	It should as 10 to tree	As expected	Pass
T06	Add an age to BinarySearchTree	ageTree.add(new AgeData(1))	It should as 1 to tree	As expected	Pass
T07	Print tree using toString	System.out.println(ageTree.toString())	It should print tree	As expected	Pass

T08	Find counts of the age in tree which are smaller than the given input age(youngerThan)	ageTree.youngerThan(5)	1	1	Pass
T09	Find counts of the age in tree which are smaller than the given input age(youngerThan)	ageTree.youngerThan(10)	2	2	Pass
T10	Find counts of the age in tree which are smaller than the given input age(youngerThan)	ageTree.youngerThan(15)	3	3	Pass
T11	Find counts of the age in tree which are bigger than the given input age(olderThan)	geTree.olderThan(5))	2	2	Pass
T12	Find counts of the age in tree which are bigger than the given input age(olderThan)	geTree.olderThan(10))	1	1	Pass
T13	Find counts of the age in tree which are bigger than the given input age(olderThan)	geTree.olderThan(15))	1	1	Pass
T14	Removing an element which is in the tree	remove(10)	Should returns the object if removed	As expected	Pass
T15	Removing an element which is NOT in the tree	remove(100)	Should return null	As expected	Pass
T16	Finding an existing element in the tree	find(new AgeData(20))	It should return corresponding age and number of people node	As expected	Pass
T17	Finding NON existing element in the tree	find(new AgeData(100))	It should return false	As expected	Pass

PART04

CLASS DIAGRAM:



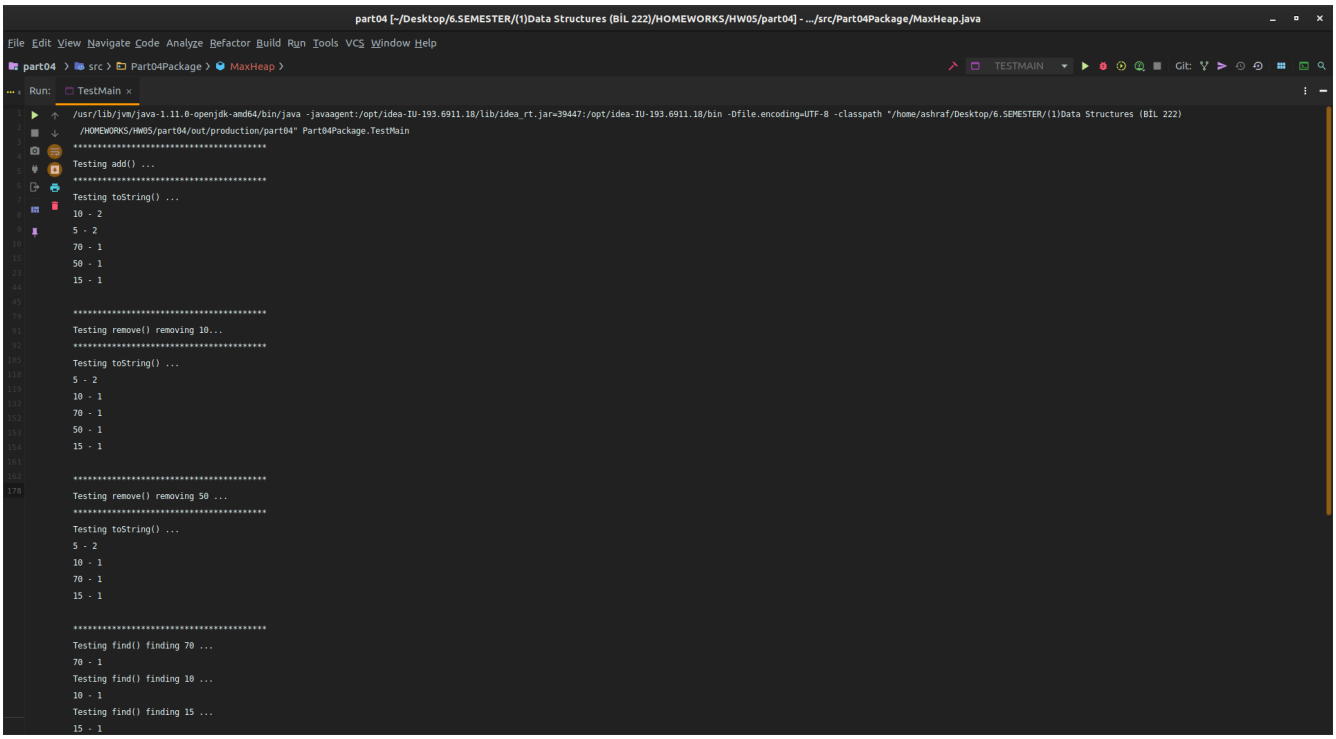
PROBLEM SOLUTION APPROACH:

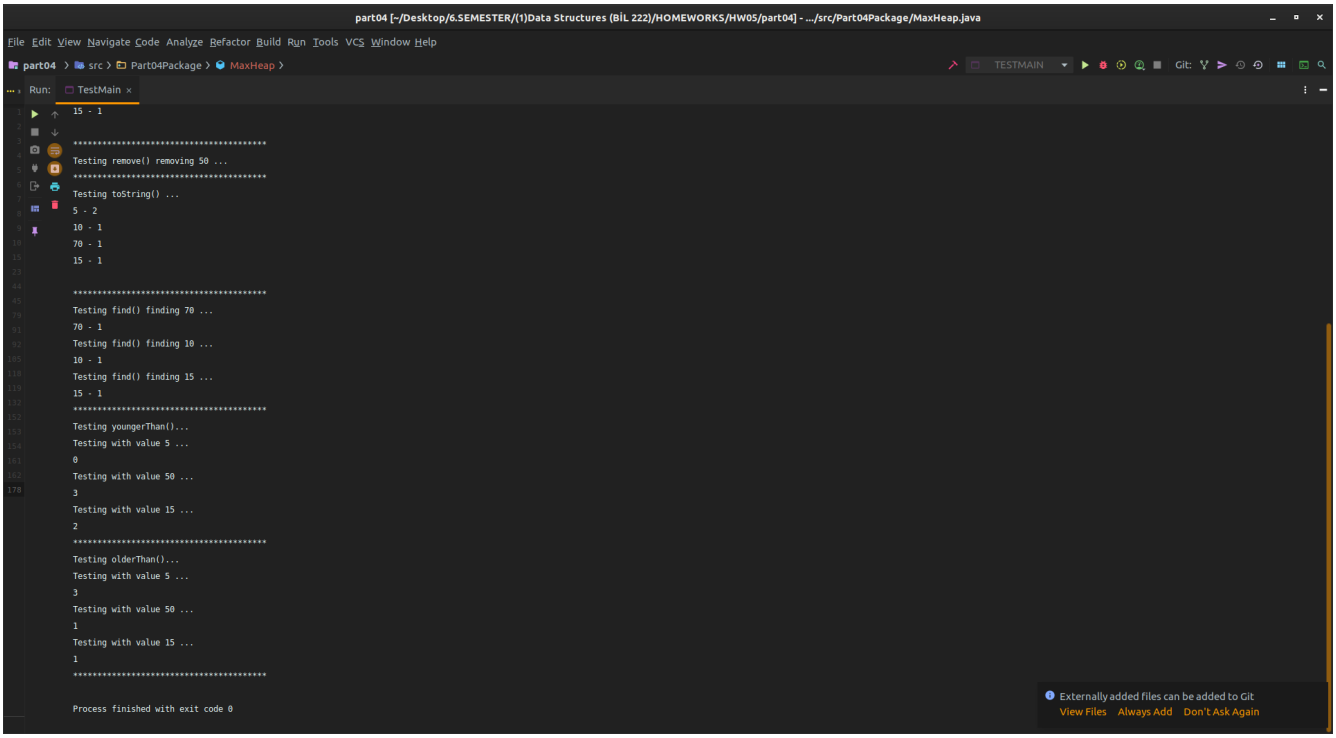
– I have used compareTo methods to compare the numperOfPeople in any particular node of heap tree.

– heap class which holds an Arraylist to represent heap tree (max_heap tree) and comparison is done considering counts of people in a particular node of heap elements.

– AgeData class is used to hold the infomarion about one age and it’s count

COMMAND LINE SCREEN SHOTS:





TEST CASES:

Test Case ID	Test Scenarios	Test Data	Expected Results	Actual Result	Pass/Fail
T01	Adding element into heap tree	add(new AgeData(10))	It should add 10 to heap tree	As expected	Pass
T02	Adding element into heap tree	add(new AgeData(5))	It should add 5 to heap tree	As expected	Pass
T03	Adding element into heap tree	add(new AgeData(70))	It should add 70 to heap tree	As expected	Pass
T04	Adding element into heap tree	add(new AgeData(10))	It should increment number of people of node which holds age 10 and then update the heap tree	As expected	Pass
T05	Adding element into heap tree	add(new AgeData(50))	It should add 50 to heap tree	As expected	Pass

T06	Adding element into heap tree	add(new AgeData(5))	It should increment number of people of node which holds age 5 and then update the heap tree	As expected	Pass
T07	Adding element into heap tree	add(new AgeData(15))	It should add 15 to heap tree	As expected	Pass
T08	Displaying heap tree using toString method	heap.toString()	It should retruns string holding heap tree to be displayed using print methods of java	As expected	Pass
T09	Removing an element existing in heap tree	heap.remove(10);	It should decrement the number of people of node which holds 10 since our 10 counts were 2 and then update the heap tree	As expected	Pass
T10	Removing an element existing in heap tree	heap.remove(50);	It should delete node which holds 50 in it since we have only one 50 so it will delete the corresponding node	As expected	Pass
T11	Finding an existing element in the heap tree	find(new AgeData(5))	It should return corresponding age and number of people node	As expected	Pass
T12	Finding NON existing element in the heap tree	find(new AgeData(50))	It should return false	As expected	Pass
T13	Find counts of the age in tree which are smaller than the given input age(youngerThan)	heap.youngerThan (5)	0	0	Pass
T14	Find counts of the age in tree which are smaller than the given input age(olderThan)	heap.olderThan(10)	2	2	Pass