Gebze Technical University
Department of Computer Engineering CSE
222/505 - Spring 2020
Homework 5 Report

Alper YAŞAR
151044072

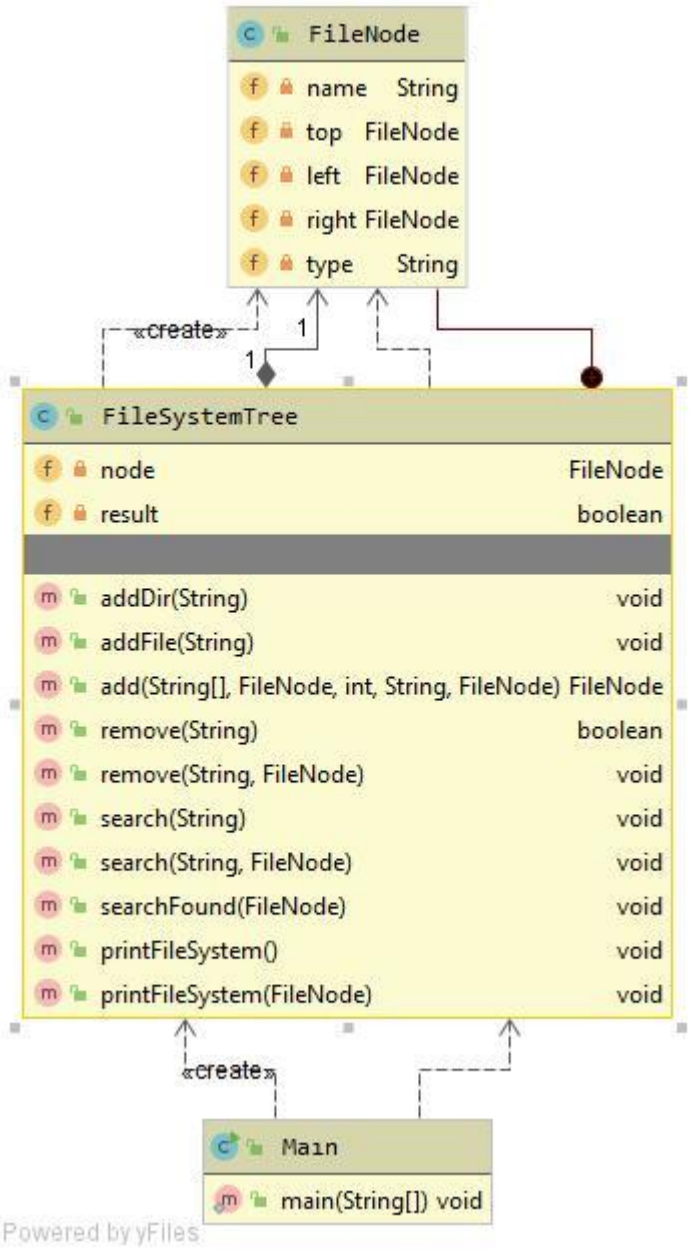
1 - Q1

1. SYSTEM REQUIREMENTS

Implement FileSystemTree class to handle a file system hierarchy in a general tree structure. You need to implement FileNode class to handle the nodes of the tree. A node can be created either for a file or a directory. You will decide how discrimination is done between files and directories. Add(E item)

1. **addir()**
   Add directory to directories path with creating tree. Call add function. Split string into string array. Send array, path node, zero(for array index) and “dir” for what type inserting to path.
2. **addFile()**
   Add directory to directories path with creating tree. Call add function. Split string into string array. Send array, path node, zero(for array index) and “file” for what type inserting to path.
3. **add()**
   If index equal to array size -1 which means this is inserting element, and check tree’s node null. If node is null than return new node to last and insert to tree. Assign previous directory to top which means holding file or directory in which directory.
   Else if check directory path. If directory path name matching with directory send its left element to recursive and increase index. Because subdirectory holding in directory left.
   Else if not matching names check subdirectories. So send right node that’s holding other elements in same directory.
4. **remove()**
   Firstly assign false to result. Because if file or directory wishing to delete is not found in path it must be false and printing warning in main. Split string to array and send last element and node to another remove function. This is searching recursively and deleting element.
   Call recursive to end of tree. If the file is wishing to delete is found delete it but is the wishing deleting is directory check it has element at once ask to user and listing what have been in folder.
5. **search()**
   Call function to end of tree and if given string has been found then send node to searchFound because this function recursively writing directory path.
6. **printFileSystem()**
   Print all files and directories in tree.

2. Class Diagram



3. PROBLEM SOLUTION APPROACH

Generally sending wanting add, search or deleting file or directory to method. Then split string to array and send with node trees and array to new method. Because this new methods searching recursively and insert or deleting wishing. Recursively search is better than loop.

4. TEST CASES

Test Case	Test Scenerio	Test Data	Expected Result
1. addDir	add directories to the file system. The path of the new directory will be given as a parameter to the method.	root/first_directory	Add to root left node.
2. addDir		root/second_directory	Add to “first directory” right node. Like a sibling.
3. addDir		root/second_directory/new_directory	Add to “second directory” left node. Like a child it.
4. addFile	add files to the file system. The path of the new file will be given as a parameter to the method.	root/second_directory/new_directory/new_file .doc	Add to “new directory” left node. Like a child it.
5. search	search the entire file system for a directory or a file including the given search characters in its name.	“new”	dir - root/second_directory/new_directory file - root/second_directory/new_directory/new_file.doc
6. remove	remove a directory (or a file) from the file system. The path of the directory (or the file) will be given as a parameter to the method. The method will warn the user if the path cannot be found. If the directory includes some other directories (or files), method will list the contents and ask the user whether to remove or not.	root/first_directory/new_file.txt	The path not found.
7. remove		root/second_directory/new_directory	Listing all files and directories in directory and ask the user for deleting.
8. printFileSystem	returns the number of people younger than an age.		print the whole tree.

Tested with given examples in pdf.

5. RUNNING AND RESULTS

```
All Element in Tree
root
first_directory
new_file.txt
second_directory
new_directory
new_file.doc

Search file or directory names including "new"
file - root/first_directory/new_file.txt
dir - root/second_directory/new_directory
file - root/second_directory/new_directory/new_file.doc

In new_directory
new_file.doc
Do you want to delete this directory?
1 - Yes
2 - No
1

 All Element in Tree
root
first_directory
second_directory
```

All result is expecting.

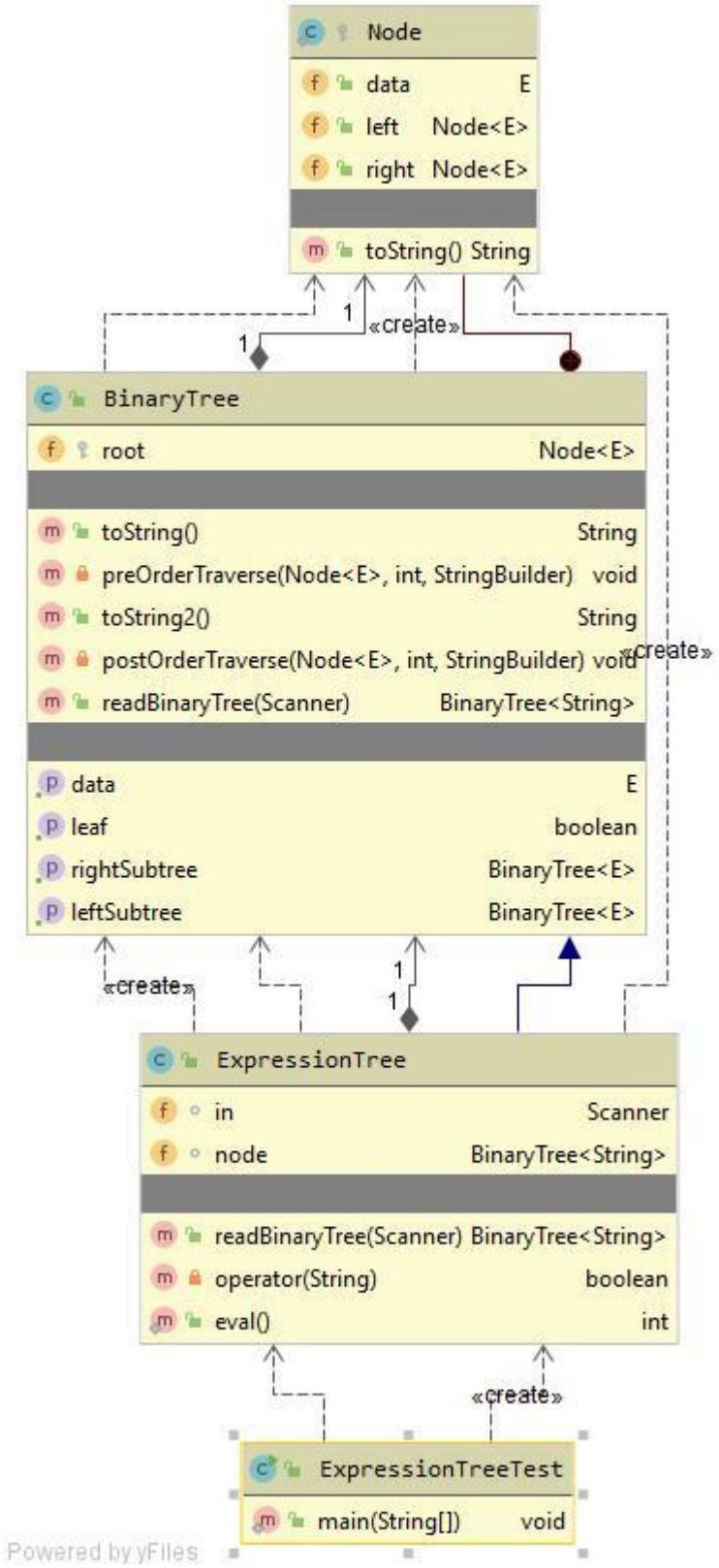
```
Do you want to delete this directory?
1 - Yes
2 - No
2

 All Element in Tree
root
first_directory
second_directory
new_directory
new_file.doc
```

When user choose do not delete. Existing file.

3 – Q3

1. Class Diagram



2. Problem Solution Approach

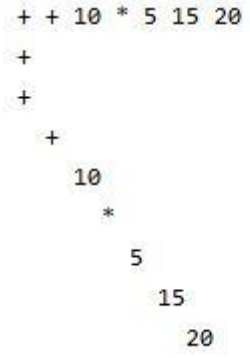
Implemented ExpressionTree class of arithmetic operations which extends the BinaryTree class implementation. ExpressionTree implementation have a constructor to initialize the tree structure with the given expression string. The expression string will be given as a parameter to the constructor. The expressions will include integer operands and arithmetic operators. Operands and operators will be separated by spaces. The constructor will use the overridden readBinaryTree method.

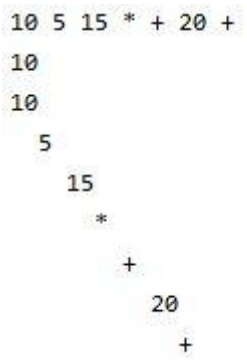
Reading string as a wanted from me but I can not split them left and right child. I try read and just read all expression to left child and can not evaluate expression.

3. Test Case

Test Case	Test Scenerio	Test Data	Expected Result
1. constructor	Read to tree binary	`+ + 10 * 5 15 20`	
2. constructor		`10 5 15 * + 20 +`	.

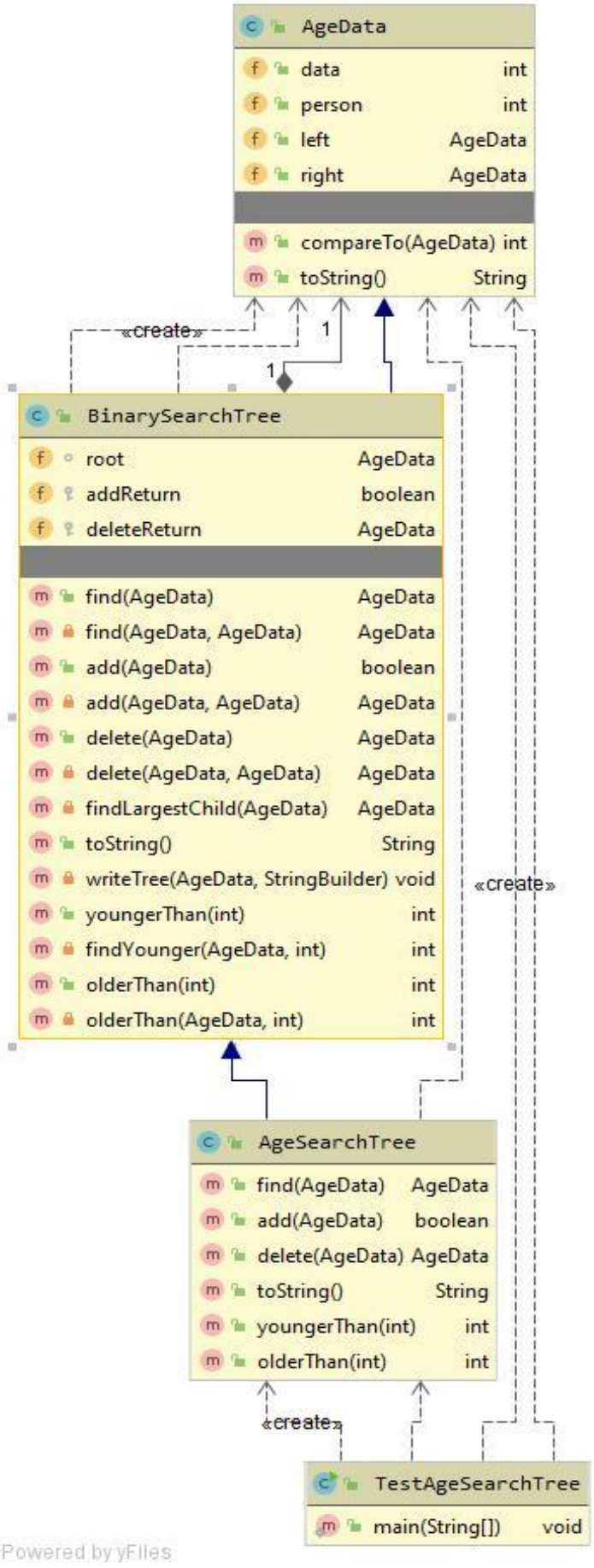
4. Running and Result





3 – Q3

5. Class Diagram



6. Problem Solution Approach

Record the number of people in each age for a population. Extended the BinarySearchTree class as AgeSearchTree class. Implemented AgeData class to handle both age and number of people at that age values. AgeData is Comparable

• While adding a node, the add function will first check if a node with that age exists. If it exists, the number of people field of the AgeData object in that node will be increased 1. Otherwise a new node with the AgeData object will be inserted.

• While removing a node, the remove function will first check if a node with that age exists. If it exists and the number of people field of this node's AgeData object is greater than 1, it will decrease the number of people field 1. If the number of people field is 1, it will remove the node.

• The find method will get an AgeData object of any age and find the AgeData object with the same age and return it.

• youngerThan method which returns the number of people younger than an age.

• olderThan method which returns the number of people older than an age.

7. Test Case

Test Case	Test Scenerio	Test Data	Expected Result
1. add	first check if a node exists. If it exists, the number of people will be increased 1. Otherwise a new object will be inserted.	ageTree.add(new AgeData(10));	Add to root.
2. add		ageTree.add(new AgeData(20));	Add to root right node.
3. add		ageTree.add(new AgeData(5));	Add to root left node.
4. add		ageTree.add(new AgeData(10));	Increment root person number 1.
5. remove	first check if a node exists. If it exists and the number of people is greater than 1, it will decrease the number of people 1. If the number of people is 1, it will remove the node.	ageTree.remove(new AgeData(10));	Decrement root person number 1.
6. toString	Return all node a string	ageTree.toString();	Write all node in tree
7. find	get an AgeData object of any age and find the AgeData object with the same age and return it.	ageTree.find(new AgeData(10))	10 – 1 Because of deleting and decrement 1.
8. youngerthan	returns the number of people younger than an age.	ageTree.youngerThan(15)	2
9. olderthan	returns the number of people older than an age.	ageTree.olderThan(15)	1

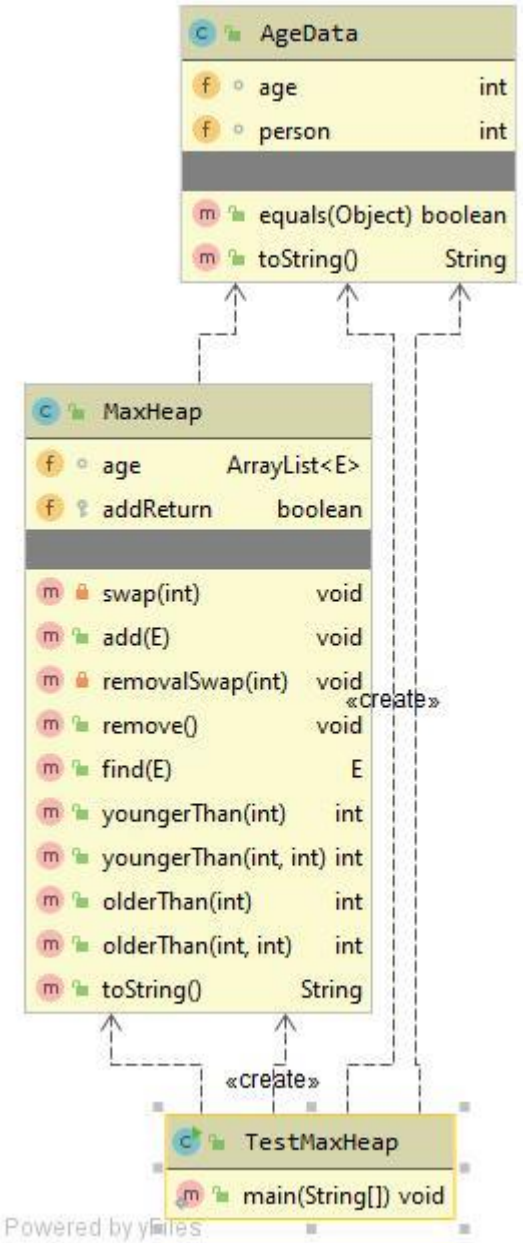
8. Running and Result

```
10 - 2
5 - 1
null
null
20 - 1
15 - 1
null
null
null

3
10 - 2
1
```

4 – Q4

1. Class Diagram



2. Problem Solution Approach

Solved the problem in Q3 by using a max_heap (where the maximum element is in the root node). Implemented ArrayList. Implemented MaxHeap class to handle the ArrayList heap operations. Used the AgeData class you implemented in Q3 to hold age and the number of people at that age data. The key of heap will be “number of people” this time. So, the age which the highest number of people is at, will be at the root.

• add function to add a new record. It will first check if an AgeData object with that age exists in any index of the ArrayList. If it exists, the number of people field of the AgeData object in that node will be increased 1. (Check if any changes in heap is needed since the key is “number of people”.) Otherwise a new heap record with the AgeData object will be inserted.

• While removing a node, the remove function will first check if a node with that age exists. If it exists and the number of people field of this node’s AgeData object is greater than 1, it will decrease the number of people field 1. (Check if any changes needed since the key is “number of people”.) If the number of people field is 1, it will remove the node.

• The find method will get an AgeData object of any age and find the AgeData object with the same age and return it.

• Add a youngerThan method which returns the number of people younger than an age.

• Add an olderThan method which returns the number of people older than an age.

3. Test Case

Test Case	Test Scenerio	Test Data	Expected Result
1. add	first check if an object exists in ArrayList. If it exists, the number of people will be increased 1. Otherwise a new AgeData object will be inserted.	heap.add(new AgeData(10));	Add to root.
2. add		heap.add(new AgeData(5));	Add the end of list
3. add		heap.add(new AgeData(10));	Increment root person number
4. add		heap.add(new AgeData(70));	Add the end of list
5. add		heap.add(new AgeData(70));	Increment person number 1 and swap with parent(5)
5. remove	first check if a node exists. If it exists and the number of people is greater than 1, it will decrease the number of people field 1. If the number of people field is 1, it will remove the node.	heap.remove(new AgeData(10));	Decrement root person number 1. And swap with child (70)
6. toString	Return all node a string	heap.toString();	Write all node in tree
7. find	get an AgeData object of any age and find the AgeData object with the same age and return it.	ageTree.find(new AgeData(10))	10 – 1 Because of deleting and decrement 1.
8. youngerthan	returns the number of people younger than an age.	heap.youngerThan(15)	2
9. olderthan	returns the number of people older than an age.	heap.olderThan(15)	2

4. Running and Result

```
[10 - 2, 5 - 2, 70 - 1, 50 - 1, 15 - 1]
2
10 - 2
3
```