GIT Department of Computer Engineering

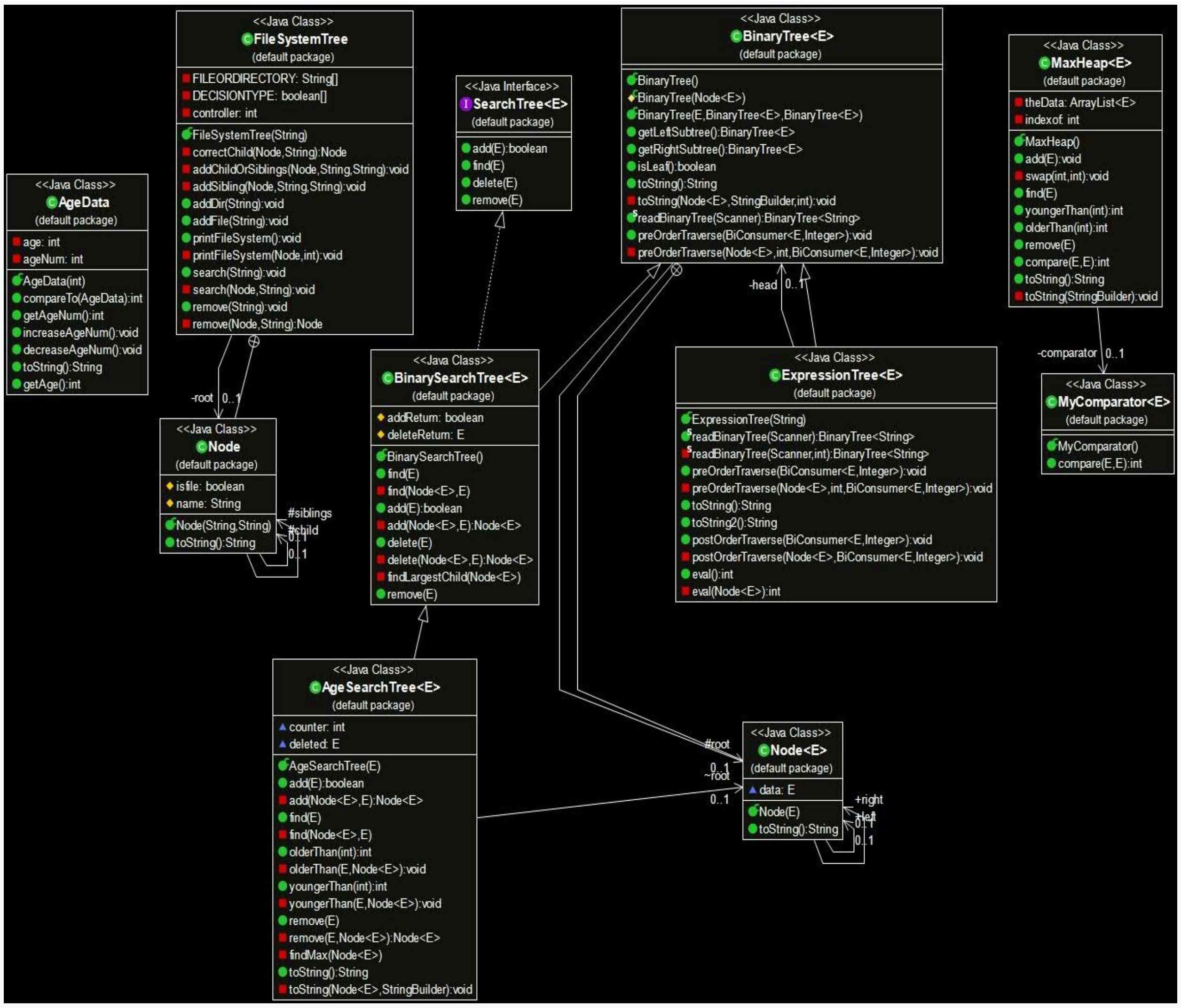
CSE 222/505 - Spring 2020

Homework 5 Report

161044063

ZAFER ALTAY

1-)CLASS DIAGRAM



2-)TEST CASES

For FileSystem:

- No paramater constructor
- addDir method
- addFile method
- remove method
- search method
- printFileSystem method

The expected result and output of these methods are shown in the next heading.

For ExpressionTree:

- readBinaryTree method
- Eval method
- PreOrderTraverse method
- PostOrderTraverse method
- toString method
- toString2 method

The expected result and output of these methods are shown in the next heading.

For AgeSearchTree:

- add method

- olderThan method
- youngerThan method
- Find method
- Remove method
- toString method

The expected result and output of these methods are shown in the next heading.

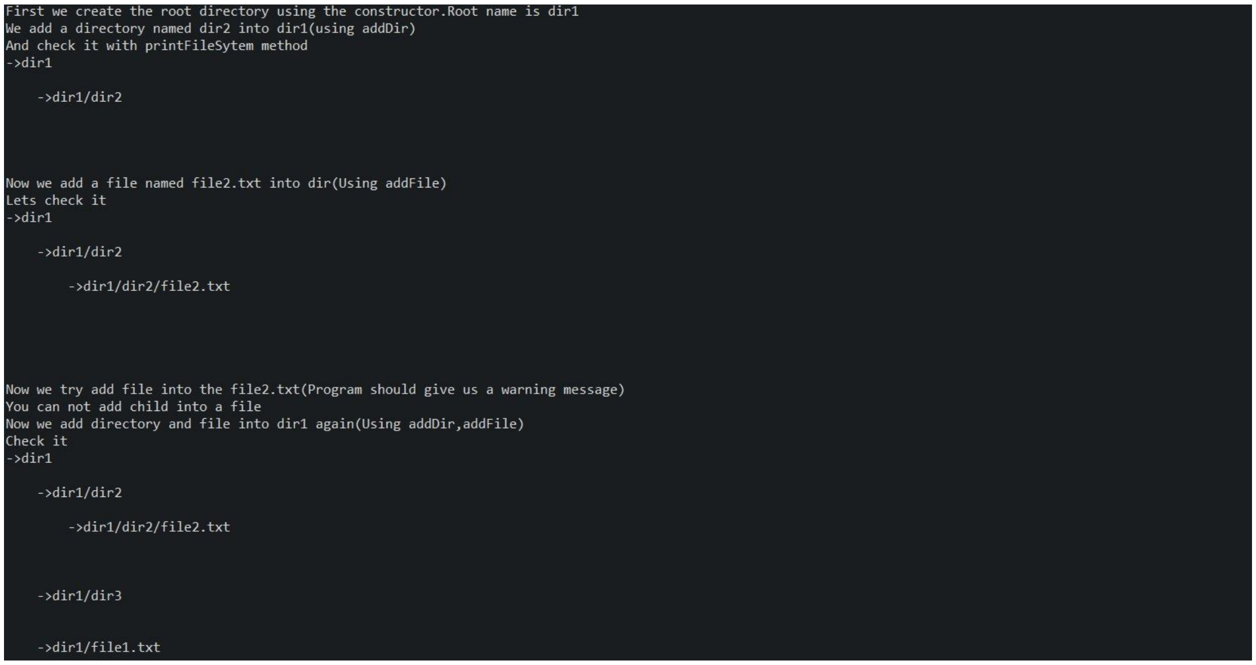
For MaxHeap:

- add method
- remove method
- comparator method
- Find method
- toString method
- Special deletions
- Special additions
- youngerThan method
- olderThan method
- swap method for switch indexes

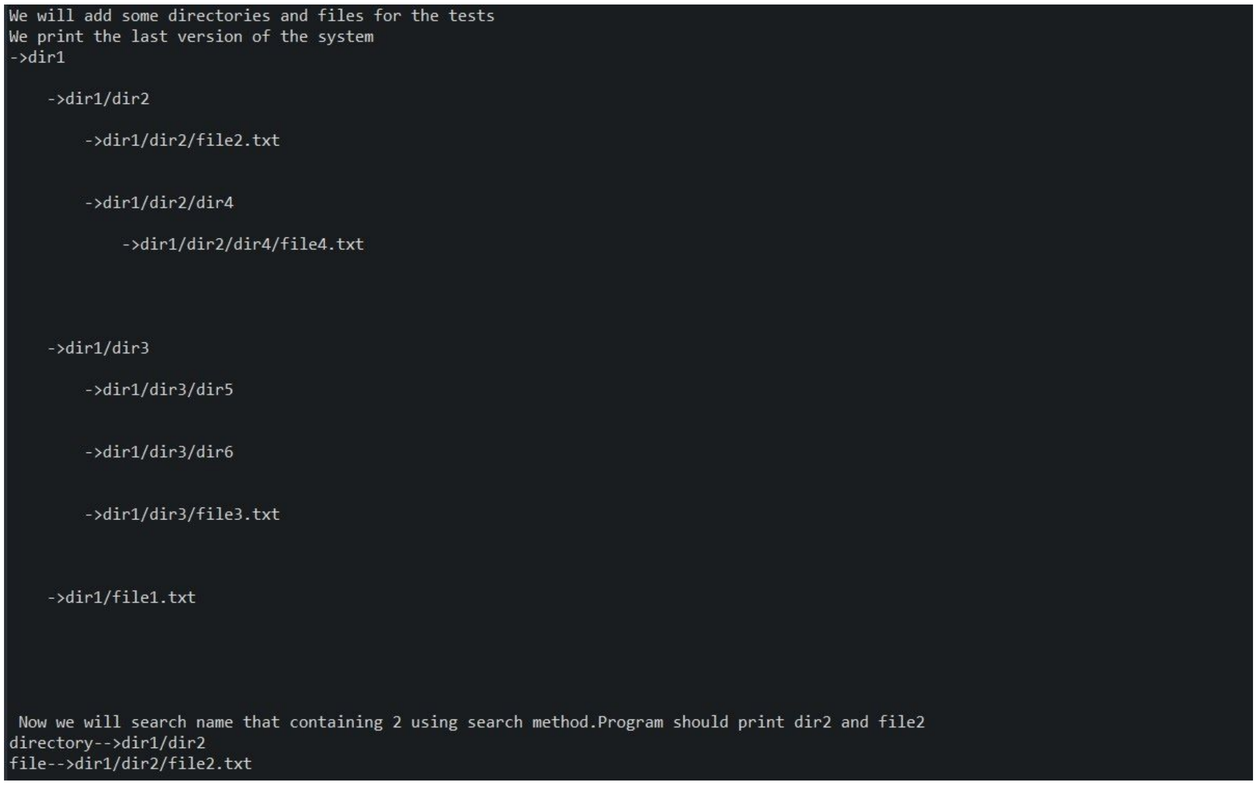
The expected result and output of these methods are shown in the next heading.

3-) Running command and results

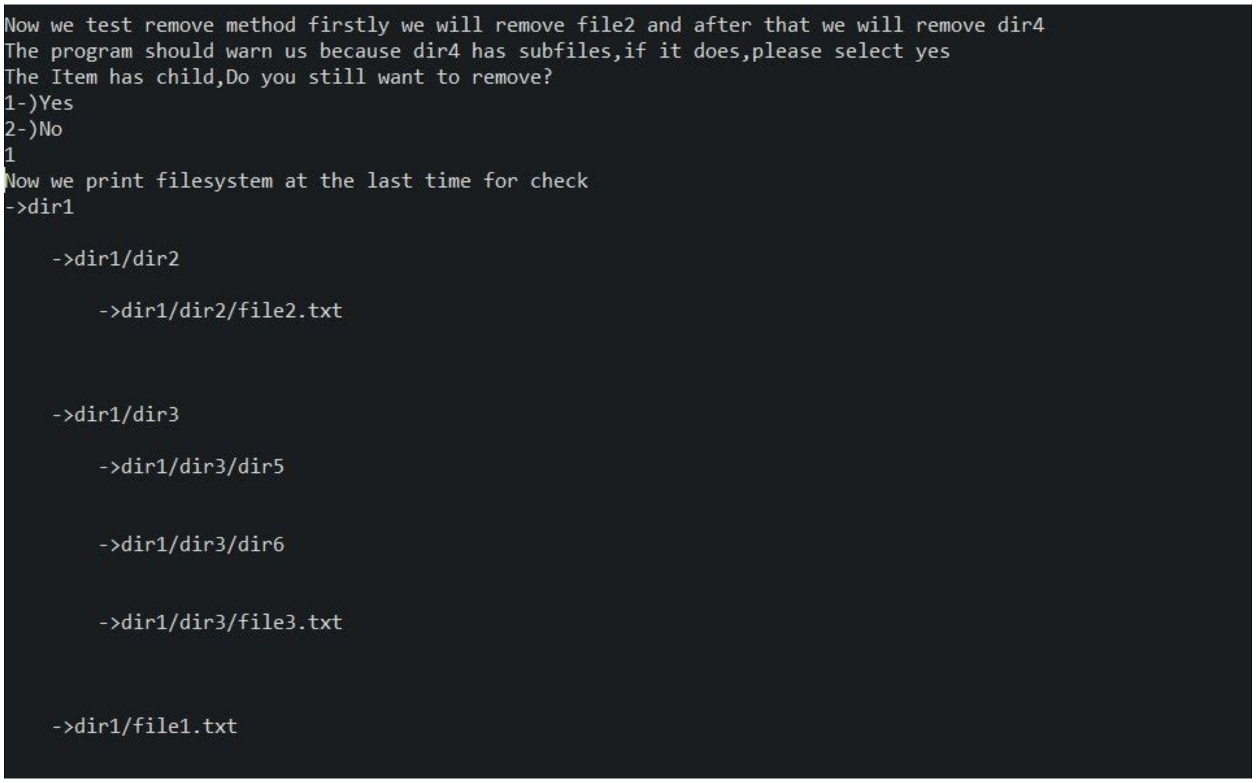
For File System:



```
First we create the root directory using the constructor.Root name is dir1
We add a directory named dir2 into dir1(using addDir)
And check it with printFileSytem method
->dir1

    ->dir1/dir2




Now we add a file named file2.txt into dir(Using addFile)
Lets check it
->dir1

    ->dir1/dir2

        ->dir1/dir2/file2.txt




Now we try add file into the file2.txt(Program should give us a warning message)
You can not add child into a file
Now we add directory and file into dir1 again(Using addDir,addFile)
Check it
->dir1

    ->dir1/dir2

        ->dir1/dir2/file2.txt



    ->dir1/dir3


    ->dir1/file1.txt
```

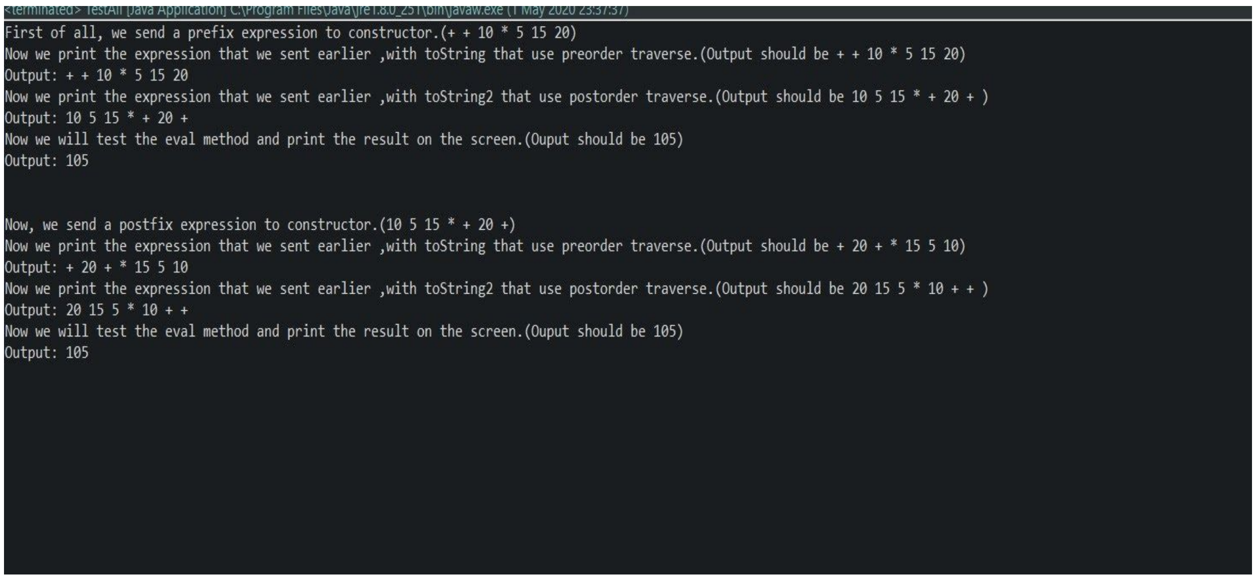


```
We will add some directories and files for the tests
We print the last version of the system
->dir1

    ->dir1/dir2

        ->dir1/dir2/file2.txt


        ->dir1/dir2/dir4

            ->dir1/dir2/dir4/file4.txt



    ->dir1/dir3

        ->dir1/dir3/dir5


        ->dir1/dir3/dir6


        ->dir1/dir3/file3.txt



    ->dir1/file1.txt




 Now we will search name that containing 2 using search method.Program should print dir2 and file2
directory-->dir1/dir2
file-->dir1/dir2/file2.txt
```



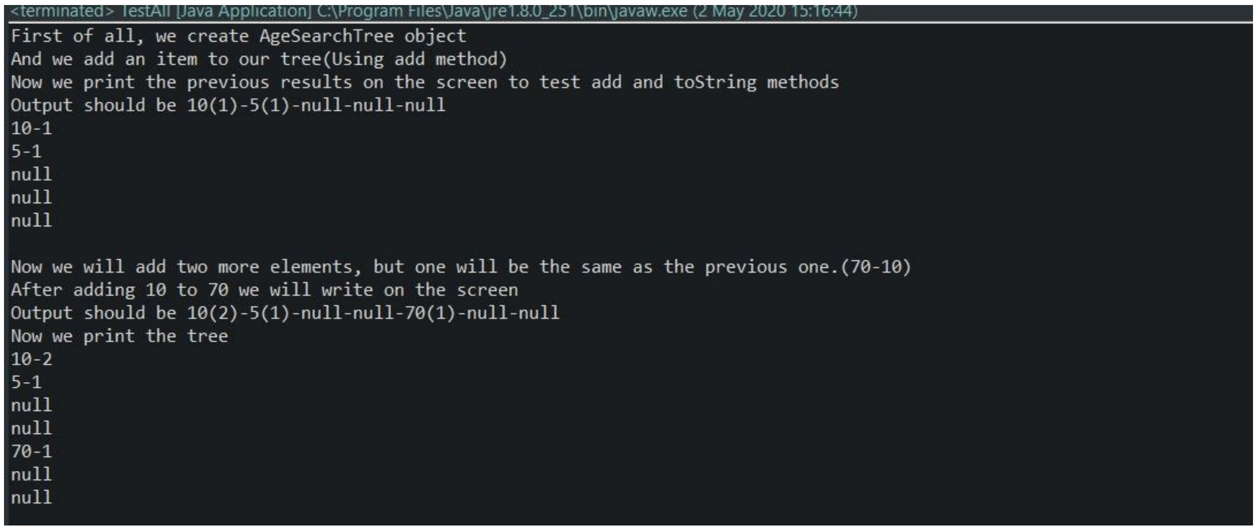
```
Now we test remove method firstly we will remove file2 and after that we will remove dir4
The program should warn us because dir4 has subfiles,if it does,please select yes
The Item has child,Do you still want to remove?
1-)Yes
2-)No
1
Now we print filesystem at the last time for check
->dir1

    ->dir1/dir2

        ->dir1/dir2/file2.txt



    ->dir1/dir3

        ->dir1/dir3/dir5


        ->dir1/dir3/dir6


        ->dir1/dir3/file3.txt



    ->dir1/file1.txt
```

For ExpressionTree:

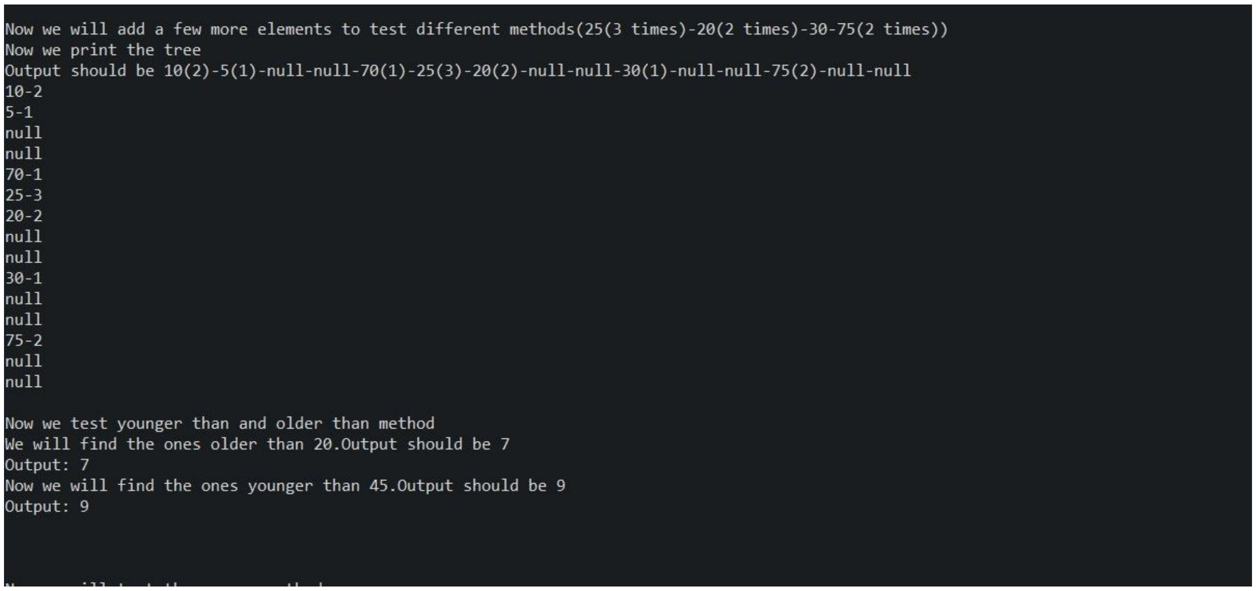


```
<terminated> TestAll [Java Application] C:\Program Files\Java\jre1.8.0_251\bin\javaw.exe (1 May 2020 23:37:37)
First of all, we send a prefix expression to constructor.(+ + 10 * 5 15 20)
Now we print the expression that we sent earlier ,with toString that use preorder traverse.(Output should be + + 10 * 5 15 20)
Output: + + 10 * 5 15 20
Now we print the expression that we sent earlier ,with toString2 that use postorder traverse.(Output should be 10 5 15 * + 20 + )
Output: 10 5 15 * + 20 +
Now we will test the eval method and print the result on the screen.(Ouput should be 105)
Output: 105


Now, we send a postfix expression to constructor.(10 5 15 * + 20 +)
Now we print the expression that we sent earlier ,with toString that use preorder traverse.(Output should be + 20 + * 15 5 10)
Output: + 20 + * 15 5 10
Now we print the expression that we sent earlier ,with toString2 that use postorder traverse.(Output should be 20 15 5 * 10 + + )
Output: 20 15 5 * 10 + +
Now we will test the eval method and print the result on the screen.(Ouput should be 105)
Output: 105
```

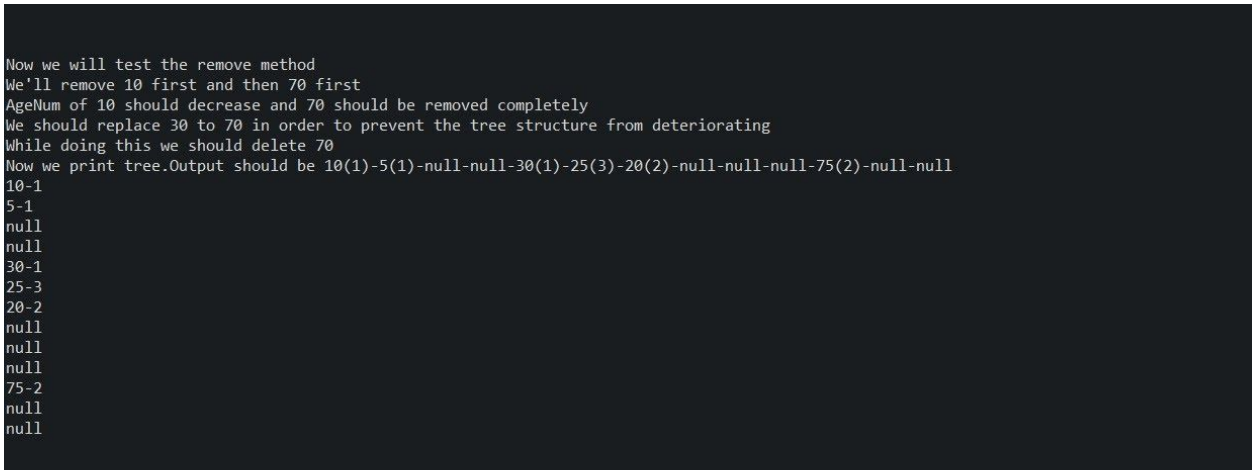
For AgeSearchTree:



```
<terminated> TestAll [Java Application] C:\Program Files\Java\jre1.8.0_251\bin\javaw.exe (2 May 2020 15:16:44)
First of all, we create AgeSearchTree object
And we add an item to our tree(Using add method)
Now we print the previous results on the screen to test add and toString methods
Output should be 10(1)-5(1)-null-null-null
10-1
5-1
null
null
null

Now we will add two more elements, but one will be the same as the previous one.(70-10)
After adding 10 to 70 we will write on the screen
Output should be 10(2)-5(1)-null-null-70(1)-null-null
Now we print the tree
10-2
5-1
null
null
70-1
null
null
```

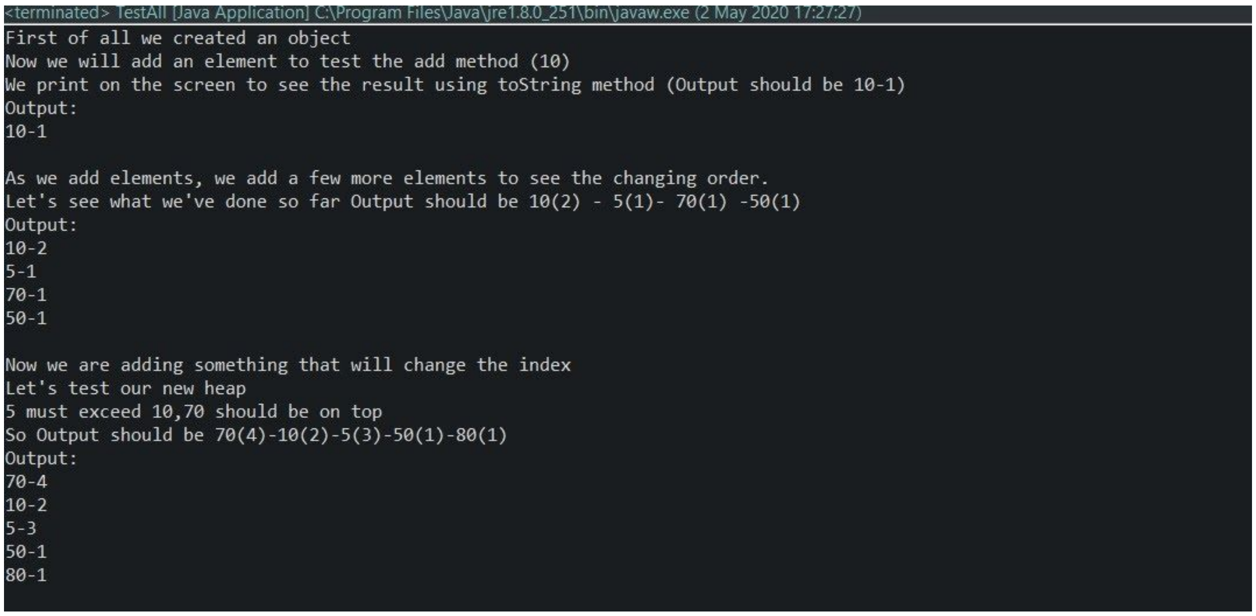


```
Now we will add a few more elements to test different methods(25(3 times)-20(2 times)-30-75(2 times))
Now we print the tree
Output should be 10(2)-5(1)-null-null-70(1)-25(3)-20(2)-null-null-30(1)-null-null-75(2)-null-null
10-2
5-1
null
null
70-1
25-3
20-2
null
null
30-1
null
null
75-2
null
null

Now we test younger than and older than method
We will find the ones older than 20.Output should be 7
Output: 7
Now we will find the ones younger than 45.Output should be 9
Output: 9
```

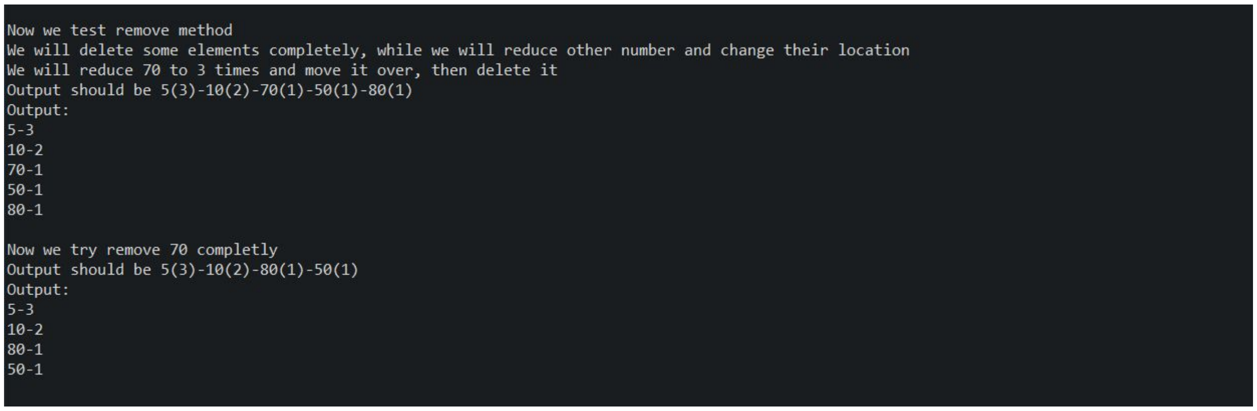


```
Now we will test the remove method
We'll remove 10 first and then 70 first
AgeNum of 10 should decrease and 70 should be removed completely
We should replace 30 to 70 in order to prevent the tree structure from deteriorating
While doing this we should delete 70
Now we print tree.Output should be 10(1)-5(1)-null-null-30(1)-25(3)-20(2)-null-null-null-75(2)-null-null
10-1
5-1
null
null
30-1
25-3
20-2
null
null
null
75-2
null
null
```

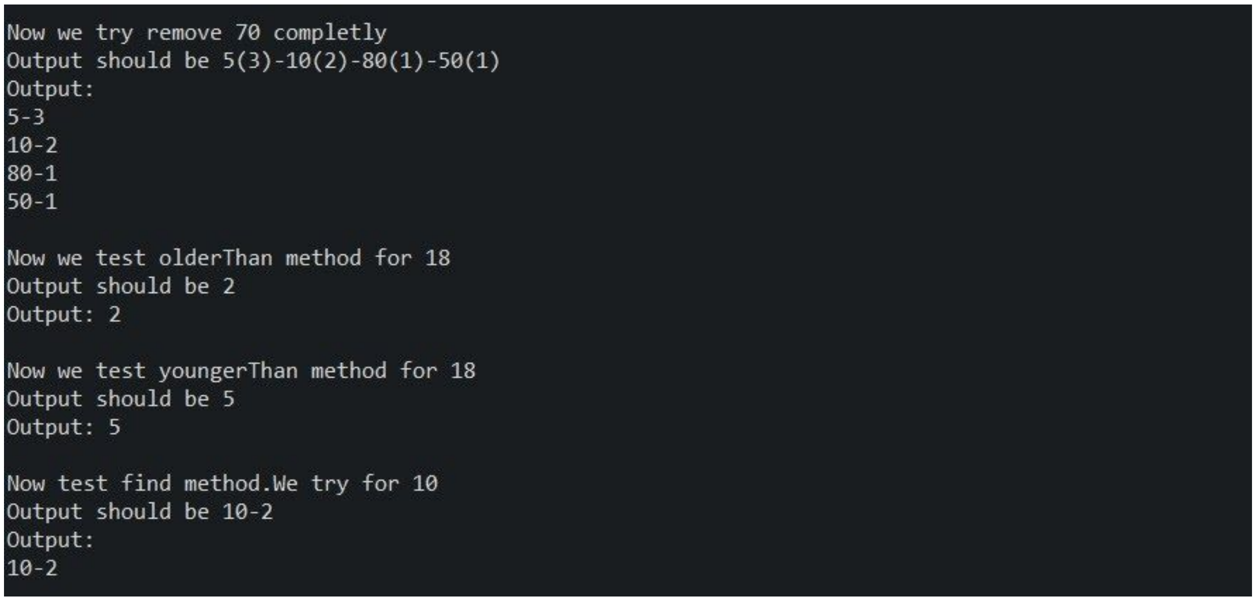
For MaxHeap:



```
<terminated> TestAll [Java Application] C:\Program Files\Java\jre1.8.0_251\bin\javaw.exe (2 May 2020 17:27:27)
First of all we created an object
Now we will add an element to test the add method (10)
We print on the screen to see the result using toString method (Output should be 10-1)
Output:
10-1

As we add elements, we add a few more elements to see the changing order.
Let's see what we've done so far Output should be 10(2) - 5(1)- 70(1) -50(1)
Output:
10-2
5-1
70-1
50-1

Now we are adding something that will change the index
Let's test our new heap
5 must exceed 10,70 should be on top
So Output should be 70(4)-10(2)-5(3)-50(1)-80(1)
Output:
70-4
10-2
5-3
50-1
80-1
```



```
Now we test remove method
We will delete some elements completely, while we will reduce other number and change their location
We will reduce 70 to 3 times and move it over, then delete it
Output should be 5(3)-10(2)-70(1)-50(1)-80(1)
Output:
5-3
10-2
70-1
50-1
80-1

Now we try remove 70 completly
Output should be 5(3)-10(2)-80(1)-50(1)
Output:
5-3
10-2
80-1
50-1
```



```
Now we try remove 70 completly
Output should be 5(3)-10(2)-80(1)-50(1)
Output:
5-3
10-2
80-1
50-1

Now we test olderThan method for 18
Output should be 2
Output: 2

Now we test youngerThan method for 18
Output should be 5
Output: 5

Now test find method.We try for 10
Output should be 10-2
Output:
10-2
```

4-)Problem Solution Approach

For FileSystemTree:

First, I created a node that holds name ,type, children and siblings.I held the type to determine if the node is file or directory.I kept children for the files to be kept under the directory.If we add any items to the subfolder and if child is not null we will add it to siblings of child.If its not null, we add it to the siblings of siblings.(Likely linked list).

addDir and addFile methods using some private methods.They are addSiblings,addChildOrSiblings and correctChild.

correctChild method finds the correct node to be added by split and return it.

addChildOrSiblings method adds if the child of the node it references is empty otherwise sends child of it to addSiblings method.

In addSiblingsMethod, If the siblings of the node is empty,it adds to siblings of node otherwise it will proceed until the siblings is empty.

In the remove method, I proceeded by looking at the children for each correct node.I looked between two / for each correct name.When we got to the right node, we asked for confirmation if it has child.Shows to the node to delete, siblings showed the siblings of node.

In the search method,program searches all the nodes and if the name of node contains the input,print the path of node.

In the printFileSystem method,I wrote recursively the root, then the child, then the siblings.

Expression Tree:

We use the node that we inherited from the binary tree as node.

First of all, I sent the string we received with the constructor to the scanner.

I sent the object of scanner to readBinaryTree.

In readBinaryTree, When I saw a white space, I split the string that i recieved from scanner.I made String array.If the first letter of the first element of the array is an operator, i sent the string directly to private readBinaryTree otherwise i reverse and send the array.

In recursive private readBinaryTree,If the element of array is a number, we created a new node containing that number and returned it because a number has no child.If the element is an operator ,i created a tree named left and right, and finally, using the special constructor, i combined these trees and returned it.Note that I read the elements with the scanner.

PreOrderTraverseMethod is method of binaryTree.

In PostOrderTraverseMethod i sent root to recursive private postOrderTraverse method ,we run the node first to the left, then to the right again and throw it into accept of consumer.

In eval method,i sent root to recursively eval method,In this method,we first move to the left and then to the right, if the value of the node is a number, we return it.If the value of our node is not a number and left right operations are finished, we process and return according to the value of our node.When the recursive function is over, the return value is evaluate of tree.

I used preOrderTraverse method in toString method and I used postOrderTraverse method in toString2 method.

AgeSearchTree:

First, we created the ageData class for our generic type.

This class, which implements the comparable interface, has variables that hold the age and number of age.It also has the compareTo method because it implements the comparator class.

The AgeSearchTree class makes the item it gets generic as the data of root node.

In add method ,If the root is null, I made the item root data(Item is generic input).If not, I sent it to private recursive add class.

In private add class,I compared the item to the data of local root(I used comparr to method of Item).If the result is less than zero, I did the same for root.left.

If the result is greater than zero, I did the same for root.right.

Or, I increased the ageNum of root data if it is equal to zero.Unlike all if root is null,ı created new node and return it.

I called the recursive private find method in the find method.Private find method gets node reference and generic type item.

In private find method,We compare the data of the reference with the item.

If item is less than reference node,we call the function again for the left of the reference

If item is greater than the data of the reference, we run the function again for the right of the reference.

If the item is equal to the data of the reference, we will find the correct node.

If reference is null,i print error message because item is not in tree.

I called the private recursive olderThan method in the OlderThan method.Private olderThan method gets node reference and generic type item.If item is smaller than data of reference root, I increased the number by 1.If it is greather, I avoided over-searching by providing custom controls.My custom control is if left ofroot is not null and right node of left node of root is null and item is greather than data of left root ,we don't need to control the left node because item is bigger than anyway.Function finishs,it returns counter.

I called the private recursive youngerThan method in the youngerThan method.Private youngerThan method gets node reference and generic type item.If item is greather than data of reference root, I increased the number by 1.If it is small, I avoided over-searching by providing custom controls.My custom control is if right root is not null and left node of right node of root is null and item is smaller than data of right root ,we don't need to control the left node because item is smaller than anyway.Function finishs,it returns counter.

In the remove method, we call the recursive private remove method.

In the recursive private remove method, I proceeded to the correct node by making a comparison and calling again for the right root if item is big or calling again the left if item is small.

If age num of correct node is bigger than 1 ,i decreased ageNum.

If the right node has only one child, I set deleted to root data and I return child.If node has 2 child.

If right of left of root is null,I set the deleted to root data,set the data of root to root left data and root left shows left of root left.So the structure of the BinarySearchTree is intact.

But If not,i set the deleted to root data and i called recursive findMax method.

In find max method,I proceeded until the right of right of root is null.When its null ,i created generic type temp and set it to data of right of root and i set the right of root to left of right of root.So the structure of binary search tree is intact.

In the toString method,i called private recursive toString Method.This method gets node reference and object of StringBuilder.If root is null,ı append null into object of SB,if not i append data of root into SB after that I call function again for right of root after ı called for left.I returned the string when the function was finished.

MaxHeap:

First, we created the ageData class for our generic type.

This class, which implements the comparable interface, has variables that hold the age and number of age.It also has the compareTo method because it implements the comparator class.

I created a special comparator class where I can make different comparisons.This class compares the ageNum of the objects it receives.Returns positive, negative, or 0 based on the AgeNum of the first object and the second object.

I created an object of arrayList because I use an arraylist to store the data in the heap.

I created an object of the comparator class. I use this object when making comparisons.

In the add method,I first call the find method and find out if the item to be added already exists.If the item already exists, I increase the ageNum of Item and then I compared the children using the comparator method.If the parent's ageNum is smaller than the parent's child, I moved using the swap method.And I continued this until I did not exceed the list.When the method was finished, all the items were in the right place without any degradation.If the item does not exist before, I add it to the end of the list.

In the find method, I checked the age of all items with the age of the item.I increased indexof once for each element I checked.If there is an element with the same age as the item, I increased the indexof once and returned the correct element.If the indexof and the size of our list are equal when the loop end, the item searched is not in the list.In this case, I returned null and set indexof to -1.Let's not forget that indexof determines the correct element in which index.

In the youngerThan method, I compared the ages of the item and list elements.I collected the ageNum of the younger ones from the elements and returned the sum after the loop was over.

In the olderThan method, I compared the ages of the item and list elements.I collected the ageNum of the older ones from the elements and returned the sum after the loop was over.

In the remove method, I first checked if there is any item using the find method.I checked the ageNumof item if item is available.If ageNum is equal to 1, I moved the last element to the item that should be removed to remove item.I also removed the last item.I used to remove and set method of arraylist.After the set operation, I compared my item with the ageNum of their children and replaced it if it was small.I continued until I crossed the list or found child that has big ageNum.If ageNum of item is greater than 1,I reduced AgeNum by one.Then I reapplied the above comparison conditions for the item.Then I returned the element.

In the toString method, I went around all the items and saved their age and ageNum in a string and I returned it.I used StringBuilder for this.