Homework #5 Report

Part-1:

In my Node design I declare a boolean. This boolean data field for determine if node is a directory or file.Also there is a list field that hold children nodes of my node, String field for hold path name and a previous node field that hold parent nodes of child nodes.I use this prev field in my find method.

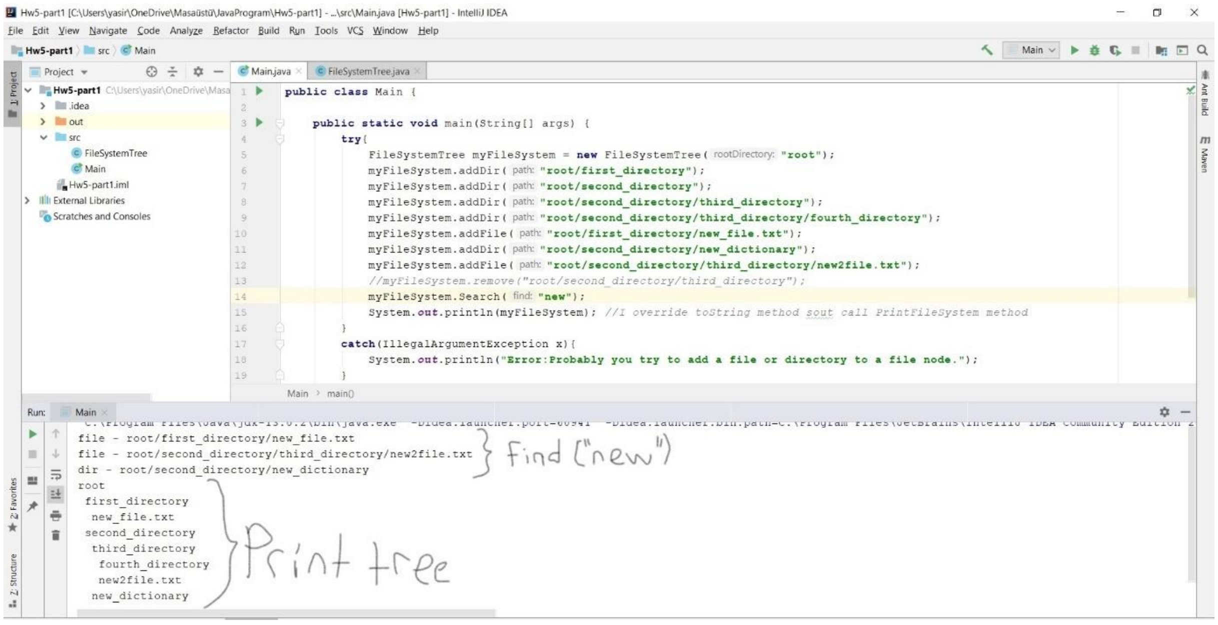
AddDir method take a string argument.This argument is path and seperate with '/' characters. I seperate path with using '/' character as token. Then traverse binary tree according to path until last / char and add last token this path as a Directory. If you try to add directory to a file, It will not add it because you cant add Directory to a file node.

AddFile method work same as AddDir method. Only difference is that adding last token as a file not directory, and again if you try to add a file to a file it will not add it because you cant add file to a file node.

Remove method take a string argument as a path. Again it tokenize string with using / char. And traverse tree to find last token node according to given path. If this node has not any children it directly remove this node but if this node has some children than this node remove also remove its children too. So in this case, first program prints Nodes that will be deleted and ask user are you sure for delete or not.If user enter 0 it will not remove but if user enter 1 then these nodes removed.

Search method find given string in nodes name and print all path for this nodes and it is a file or directory info like in pdf. In this method I use my prev data field.

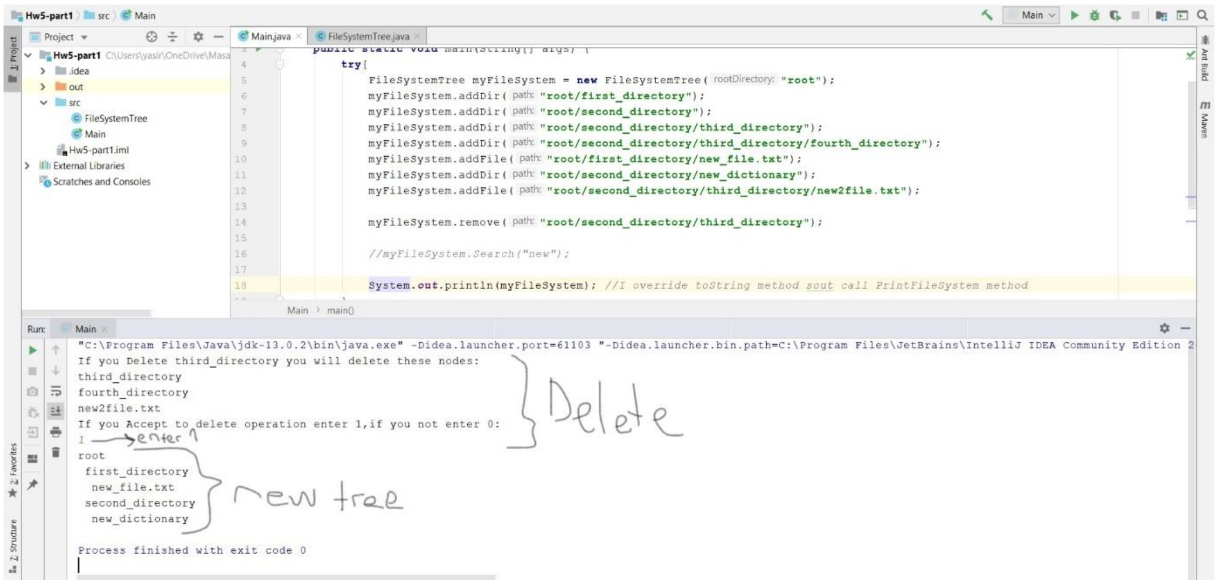
PrintFileSystem method prints my tree acording nodes depths.For example root node printed without any space character but root nodes child printed after 1 space character, so you can understand tree structure after print.



```java
public class Main {

    public static void main(String[] args) {
        try{
            FileSystemTree myFileSystem = new FileSystemTree( rootDirectory: "root");
            myFileSystem.addDir( path: "root/first_directory");
            myFileSystem.addDir( path: "root/second_directory");
            myFileSystem.addDir( path: "root/second_directory/third_directory");
            myFileSystem.addDir( path: "root/second_directory/third_directory/fourth_directory");
            myFileSystem.addFile( path: "root/first_directory/new_file.txt");
            myFileSystem.addDir( path: "root/second_directory/new_dictionary");
            myFileSystem.addFile( path: "root/second_directory/third_directory/new2file.txt");
            //myFileSystem.remove("root/second_directory/third_directory");
            myFileSystem.Search( find: "new");
            System.out.println(myFileSystem); //I override toString method sout call PrintFileSystem method
        }
        catch(IllegalArgumentException x){
            System.out.println("Error:Probably you try to add a file or directory to a file node.");
        }
```

```
file - root/first_directory/new_file.txt
file - root/second_directory/third_directory/new2file.txt
dir - root/second_directory/new_dictionary
root
 first_directory
  new_file.txt
 second_directory
  third_directory
   fourth_directory
   new2file.txt
  new_dictionary
```

Above ScreenShot After with AddDir and AddFile construct my tree I call find method with (“new”) parameter, so it prints all nodes path that has “new” string in their names and their directory or file info. After that I print my tree like a directory file system with space characters.

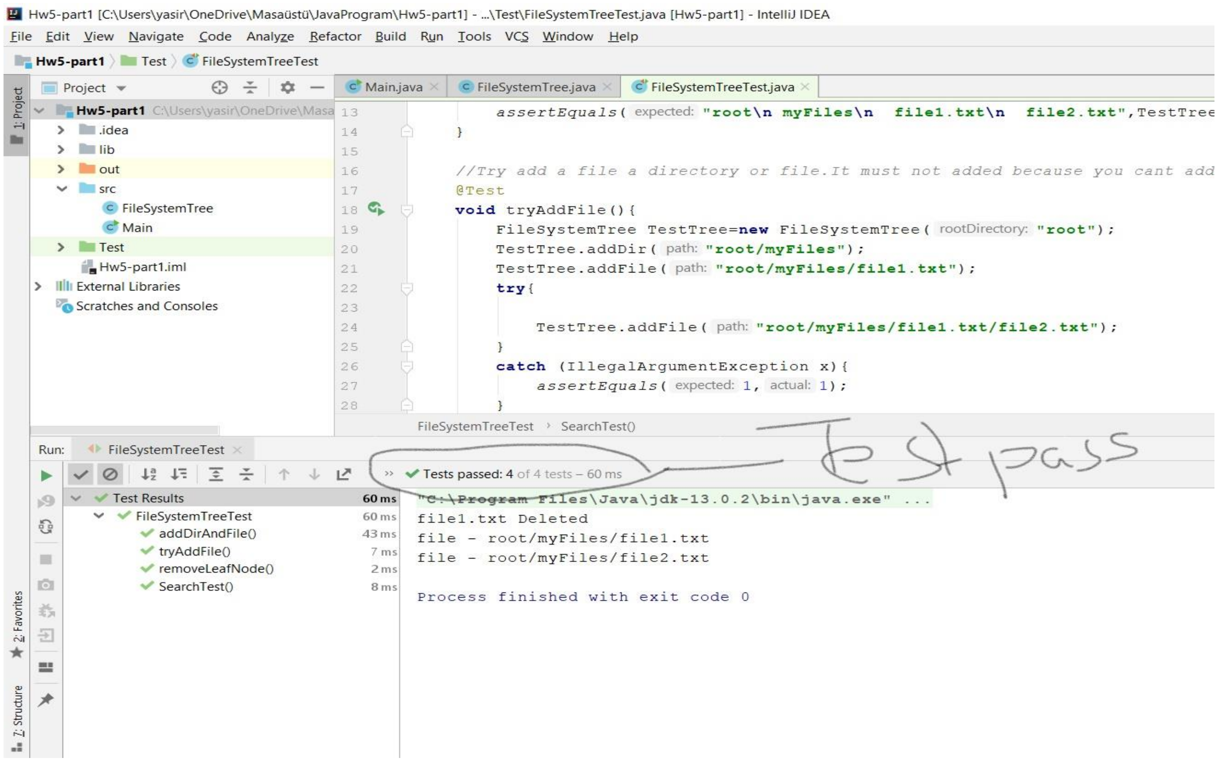


```java
    public static void main(String[] args) {
        try{
            FileSystemTree myFileSystem = new FileSystemTree( rootDirectory: "root");
            myFileSystem.addDir( path: "root/first_directory");
            myFileSystem.addDir( path: "root/second_directory");
            myFileSystem.addDir( path: "root/second_directory/third_directory");
            myFileSystem.addDir( path: "root/second_directory/third_directory/fourth_directory");
            myFileSystem.addFile( path: "root/first_directory/new_file.txt");
            myFileSystem.addDir( path: "root/second_directory/new_dictionary");
            myFileSystem.addFile( path: "root/second_directory/third_directory/new2file.txt");

            myFileSystem.remove( path: "root/second_directory/third_directory");

            //myFileSystem.Search("new");

            System.out.println(myFileSystem); //I override toString method sout call PrintFileSystem method
```

```
"C:\Program Files\Java\jdk-13.0.2\bin\java.exe" -Didea.launcher.port=61103 "-Didea.launcher.bin.path=C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2
If you Delete third_directory you will delete these nodes:
third_directory
fourth_directory
new2file.txt
If you Accept to delete operation enter 1,if you not enter 0:
1
root
 first_directory
  new_file.txt
 second_directory
  new_dictionary

Process finished with exit code 0
```

Above example After with AddDir and AddFile construct mt tree I call remove method for delete third_directory but if you delete this third_directory,fourth_directory and new2file.txt will be deleted so I print deleted file and directories and ask user if you want delete or not. If user enter 1 these nodes will be deleted. After that I print tree to Show these nodes removed.

I use Junit5 test for testing. You can find my test cases in Test folder.



Hw5-part1 [C:\Users\yasir\OneDrive\Masaüstü\JavaProgram\Hw5-part1] - ...\Test\FileSystemTreeTest.java [Hw5-part1] - IntelliJ IDEA

```java
        assertEquals( expected: "root\n myFiles\n  file1.txt\n  file2.txt",TestTree
    }

    //Try add a file a directory or file.It must not added because you cant add
    @Test
    void tryAddFile(){
        FileSystemTree TestTree=new FileSystemTree( rootDirectory: "root");
        TestTree.addDir( path: "root/myFiles");
        TestTree.addFile( path: "root/myFiles/file1.txt");
        try{

            TestTree.addFile( path: "root/myFiles/file1.txt/file2.txt");
        }
        catch (IllegalArgumentException x){
            assertEquals( expected: 1, actual: 1);
        }
```

```
Tests passed: 4 of 4 tests – 60 ms

Test Results                60 ms
  FileSystemTreeTest        60 ms
    addDirAndFile()         43 ms
    tryAddFile()             7 ms
    removeLeafNode()         2 ms
    SearchTest()             8 ms
```

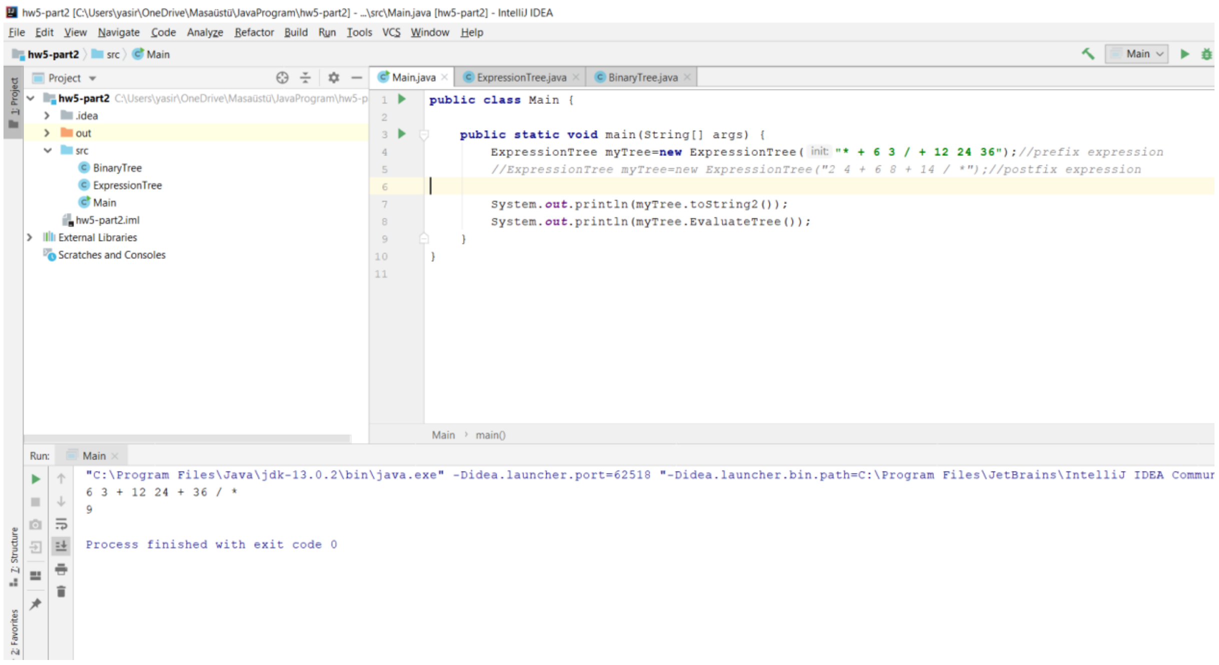
```
"C:\Program Files\Java\jdk-13.0.2\bin\java.exe" ...
file1.txt Deleted
file - root/myFiles/file1.txt
file - root/myFiles/file2.txt

Process finished with exit code 0
```

Test Case Id	Test Scenario	Test Steps	Test Data	Expected Results	Actual Results	Pass/Fail
T01	Add file and Directory	1.Create a Tree object With constructor 2.Add a directory to root 3.Add 2 file to directory	Root directory myFile Directory File1.txt File2.txt	Construct this tree: Root myFile File1.txt File2.txt	As Expected	Pass
T02	Try add a file to a file or directory	1.Create a tree object with constructor 2.Add myfile directory 3.Add file1.txt to myfile 4.Add file2.txt to file1.txt	Root Directory myFile Directory File1.txt File2.txt	Catch IllegelArgument Exception because You can't add a file or directory to a file.	As Expected	Pass
T03	Remove a Leaf node from tree	1.Create a Tree object With constructor 2.Add a directory to root 3.Add 2 file to directory 4.remove file1	Root directory myFile Directory File1.txt File2.txt	It directly delete the leaf node and print screen file1.txt deleted.	As Expected	Pass
T04	Search in Tree	1.Create a Tree object with constructor 2.Add a directory to root 3.Add 2 file to directory 4.Search("file")	Root directory myFile Directory File1.txt File2.txt	file – root/first_directory/New_file.txt File – root/second_directory/Third_directory/new2file.txt Dir – root/second_directory/new_directory	As Expected	Pass

Part-2:

My constructor can take a prefix or postfix string parameter. To understand it is a prefix or postfix I declare a boolean data field and check String first character. If its a operator than it means this is a prefix, if it is a digit then it means argument is a postfix.And I use readBinaryTree method in my constructer for construct my expression Tree.

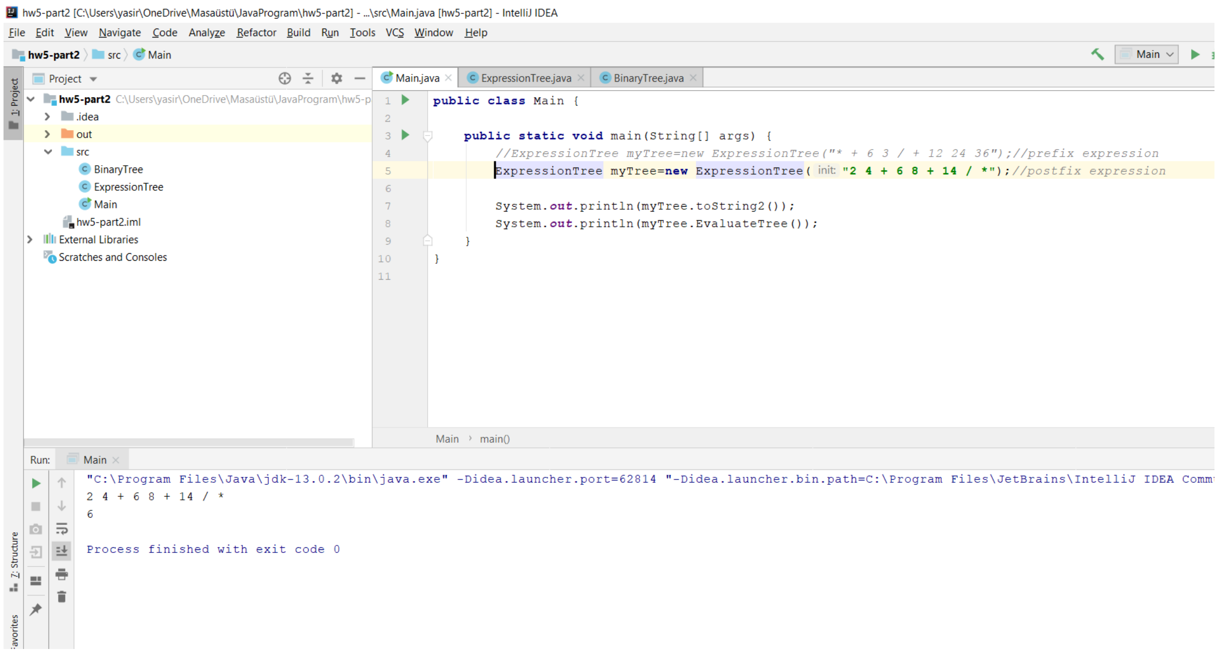
If constructer parameter is prefix:



```java
public class Main {

    public static void main(String[] args) {
        ExpressionTree myTree=new ExpressionTree("* + 6 3 / + 12 24 36");//prefix expression
        //ExpressionTree myTree=new ExpressionTree("2 4 + 6 8 + 14 / *");//postfix expression

        System.out.println(myTree.toString2());
        System.out.println(myTree.EvaluateTree());
    }
}
```

```
"C:\Program Files\Java\jdk-13.0.2\bin\java.exe" -Didea.launcher.port=62518 "-Didea.launcher.bin.path=C:\Program Files\JetBrains\IntelliJ IDEA Commur
6 3 + 12 24 + 36 / *
9

Process finished with exit code 0
```

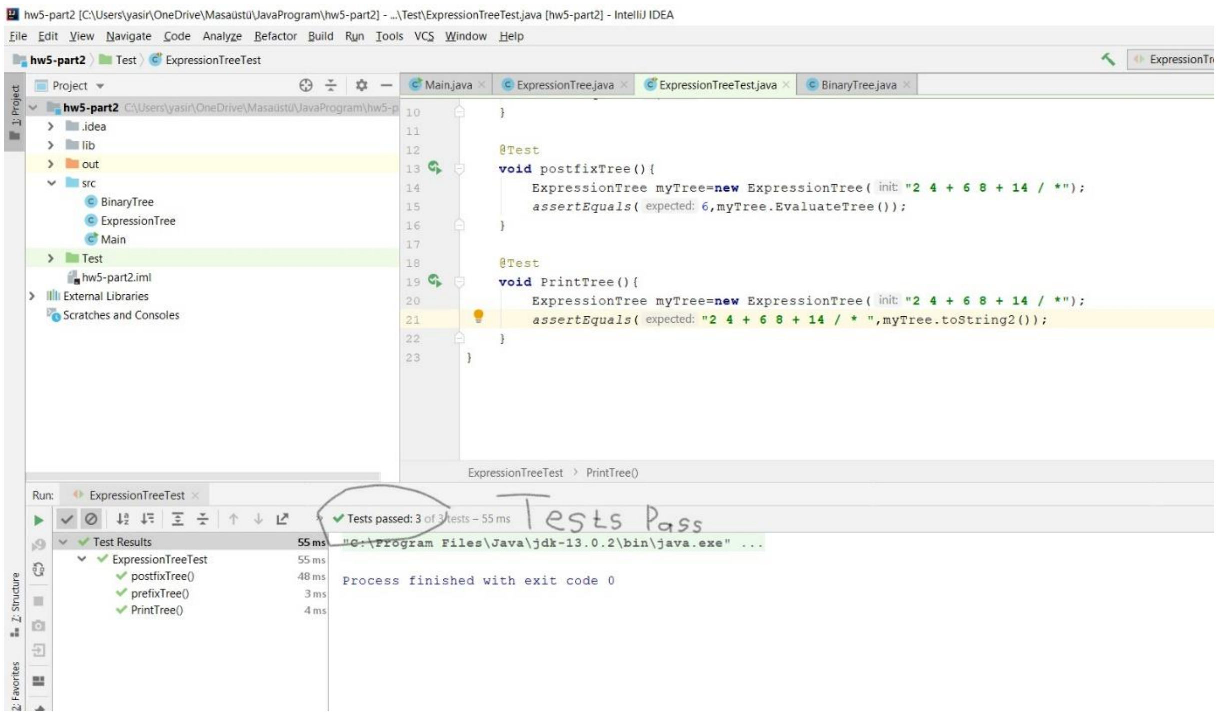
If constructer parameter is postfix:



```java
public class Main {

    public static void main(String[] args) {
        //ExpressionTree myTree=new ExpressionTree("* + 6 3 / + 12 24 36");//prefix expression
        ExpressionTree myTree=new ExpressionTree("2 4 + 6 8 + 14 / *");//postfix expression

        System.out.println(myTree.toString2());
        System.out.println(myTree.EvaluateTree());
    }
}
```

```
"C:\Program Files\Java\jdk-13.0.2\bin\java.exe" -Didea.launcher.port=62814 "-Didea.launcher.bin.path=C:\Program Files\JetBrains\IntelliJ IDEA Comm
2 4 + 6 8 + 14 / *
6

Process finished with exit code 0
```

I use Junit5 for test cases. You can find my test methods in Test Directory.



hw5-part2 [C:\Users\yasir\OneDrive\Masaüstü\JavaProgram\hw5-part2] - ...\Test\ExpressionTreeTest.java [hw5-part2] - IntelliJ IDEA

```java
    }

    @Test
    void postfixTree(){
        ExpressionTree myTree=new ExpressionTree( "2 4 + 6 8 + 14 / *");
        assertEquals( 6,myTree.EvaluateTree());
    }

    @Test
    void PrintTree(){
        ExpressionTree myTree=new ExpressionTree( "2 4 + 6 8 + 14 / *");
        assertEquals( "2 4 + 6 8 + 14 / * ",myTree.toString2());
    }
}
```

```
Test Results                55 ms
  ExpressionTreeTest        55 ms
    postfixTree()           48 ms
    prefixTree()             3 ms
    PrintTree()              4 ms
```

Tests passed: 3 of 3 tests – 55 ms

Tests Pass

```
"C:\Program Files\Java\jdk-13.0.2\bin\java.exe" ...

Process finished with exit code 0
```

Test ID	**Test Scenerio**	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Pass/Fail**
T01	Evaluate Prefix Tree	1.Call constructer with A prefix argument 2.call EvaluateTree method	* + 6 3 / + 12 24 36	9	As Expected	Pass
T02	Evaluate Postfix Tree	1.Call constructer with A postfix argument 2.Call EvaluateTree method	2 4 + 6 8 + 14 / *	6	As Expected	Pass
T03	Print Tree	1.Call constructer and create Tree 2.call toString2 method	2 4 + 6 8 + 14 / *	2	2 4 + 6 8 + 14 / *	Pass

Part-3:

I implement BinaryTree, BinarySearchTree and SearchTree interface in the book and Extend BinarySearchTree as AgeSearchTree. AgeSearchTree has one AgeData root for tree root.It has add, remove, find, youngerThan and olderThan methods. My AgeData class has 2 data field. One for age and one for number of people that same age counter(number_age).AgeData also implement Comparable interface for compare AgeData nodes.I override add, remove and find methods of BinarySearchTree in AgeSearchTree.

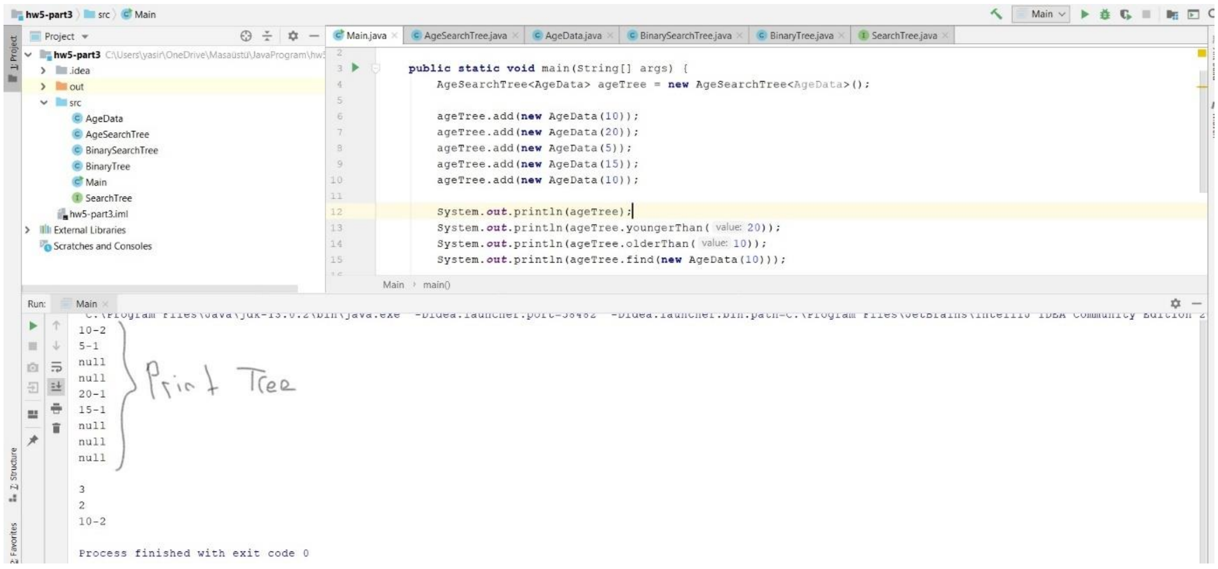
Add method take a comparable item argument and add this into tree.If my tree already has item in my tree than I just increment these node number_age by 1 but if my thee doesn’t have argument age in my tree than it just create a new node and add tree.

Remove method take a comparable item argument and find this item. If this node has more than 1 person of that age than it just decrease number_age by 1 but if this node’s number_age 1 then it find left greater node of this node and replace them and remove node.

Find method take a comparable item argument and find this item and print its age and number_age.If it cant find item in my tree it just return null.

youngerThan method take an integer argument and print how many node in tree that smaller than this argument. My method don’t traverse all tree , It work like that for example it check a node if this node age value greater than my argument than it just go left side,it didnt traverse right side because I know all right side elements bigger than my argument according to binarySearchTree laws. İf checking nodes age smaller than my argument than it count all node in the left side because I know every left side nodes smaller than my argument and move right side.

olderThan method work same youngerThan method, it just counter older Ages not youngers.



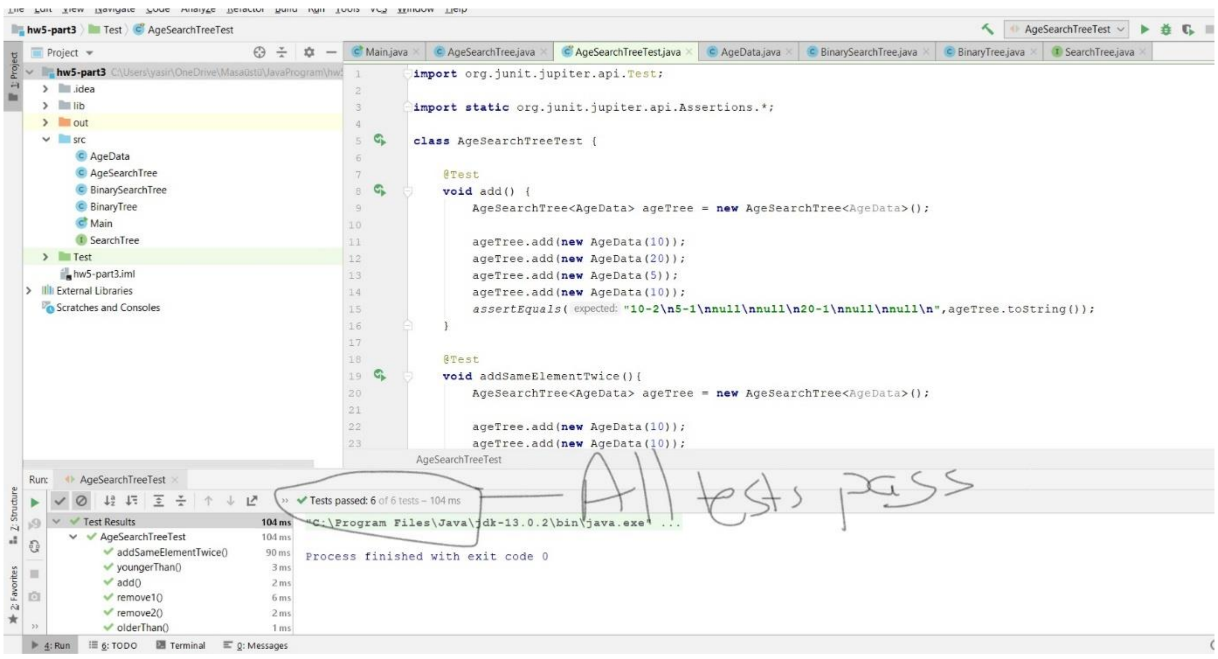
```java
    public static void main(String[] args) {
        AgeSearchTree<AgeData> ageTree = new AgeSearchTree<AgeData>();

        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));

        System.out.println(ageTree);
        System.out.println(ageTree.youngerThan( value: 20));
        System.out.println(ageTree.olderThan( value: 10));
        System.out.println(ageTree.find(new AgeData(10)));
```

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

3
2
10-2

Process finished with exit code 0
```

Above ScreenShot like in pdf I add some AgeData in tree and print tree. After that call youngerThan(20) method and that prints 3 because there is 3 node that has smaller age than

20. After that I call olderThan(10) and it prints 2 because there is 2 nodes in my tree that has bigger age than 10. Lastly, I call find(10) method and it found node that has age value of 10 and return 10-2.Also you can try remove method.

I use Junit5 for test cases. You can find my test methods in Test Directory.



```java
import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.*;

class AgeSearchTreeTest {

    @Test
    void add() {
        AgeSearchTree<AgeData> ageTree = new AgeSearchTree<AgeData>();

        ageTree.add(new AgeData(10));
        ageTree.add(new AgeData(20));
        ageTree.add(new AgeData(5));
        ageTree.add(new AgeData(10));
        assertEquals( expected: "10-2\n5-1\nnull\nnull\n20-1\nnull\nnull\n",ageTree.toString());
    }

    @Test
    void addSameElementTwice(){
        AgeSearchTree<AgeData> ageTree = new AgeSearchTree<AgeData>();

        ageTree.add(new AgeData(10));
        ageTree.add(new AgeData(10));
```

```
Tests passed: 6 of 6 tests – 104 ms

Test Results                 104 ms
  AgeSearchTreeTest          104 ms
    addSameElementTwice()     90 ms
    youngerThan()              3 ms
    add()                      2 ms
    remove1()                  6 ms
    remove2()                  2 ms
    olderThan()                1 ms

"C:\Program Files\Java\jdk-13.0.2\bin\java.exe" ...

Process finished with exit code 0
```

Test ID	Test Scenerio	Test Steps	Test Data	Expected Results	Actual Results	Pass/Fail

T01	Add element to tree	1.Create AgeSearchtree
		2.add AgeData(10)
		3.add AgeData(20)
		4.add AgeData(5)
		5add AgeData(10)
			AgeData(10)
			AgeData(20)
			AgeData(5)
			AgeData(10)
				10-2
				5-1
				Null
				Null
				20-1
				Null
				null
					As expected	Pass

T02	Add same age Twice	1.Create AgeSearchtree 2.add AgeData(10) 3.add AgeData(10)	AgeData(10)	10-2 Null Null	As Expected	Pass

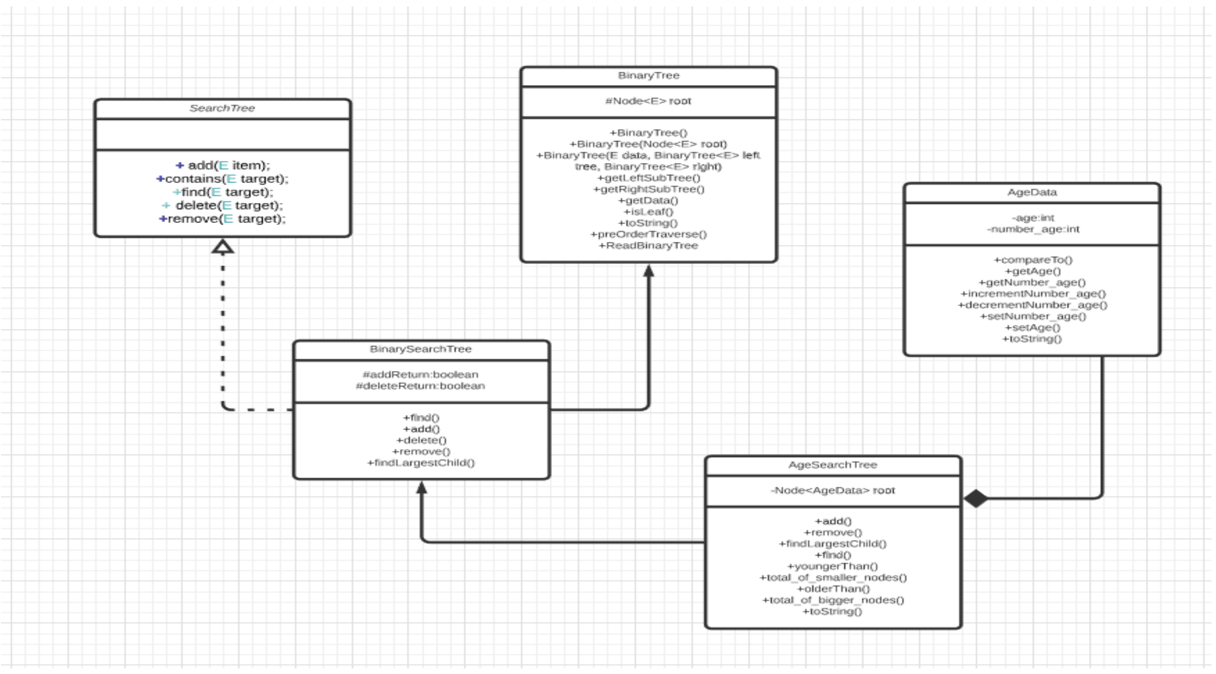
T03	Remove an element that has only 1 person that has same age	1.Create AgeSearchtree
		2.add AgeData(10)
		3.add AgeData(20)
		4.add AgeData(5)
		5.remove AgeData(5)
			AgeData(10)
			AgeData(20)
			AgeData(5)
				10-1
				Null
				20-1
				Null
				Null
					As expected	Pass

T04	Remove an element	1.Create AgeSearchtree 2.add AgeData(10)	AgeData(10)	10-1 Null Null	As Expected	Pass

	that has more than 1 person thas has same age	3.add AgeData(10) 4.remove AgeData(10)				

T05	Find Younger number	1.Create AgeSearchtree
		2.add AgeData(10)
		3.add AgeData(20)
		4. .add AgeData(5)
		5. .add AgeData(15)
		6.call youngerThan(17)
			AgeData(10)
			AgeData(20)
			AgeData(5)
			AgeData(15)
				3	As Expected	Pass

T06	Find older number	1.Create AgeSearchtree
		2.add AgeData(10)
		3.add AgeData(20)
		4. .add AgeData(5)
		5. .add AgeData(15)
		6.call olderThan(3)
			AgeData(10)
			AgeData(20)
			AgeData(5)
			AgeData(15)
				4	As Expected	Pass



Part-4:

AgeData same as in part 3.MaxHeap class has Arraylist of type AgeData and a comparator as data field. MaxHeap has a compare method for compara 2 AgeData object. There are Add,remove,find, youngerThan and olderThan methods.

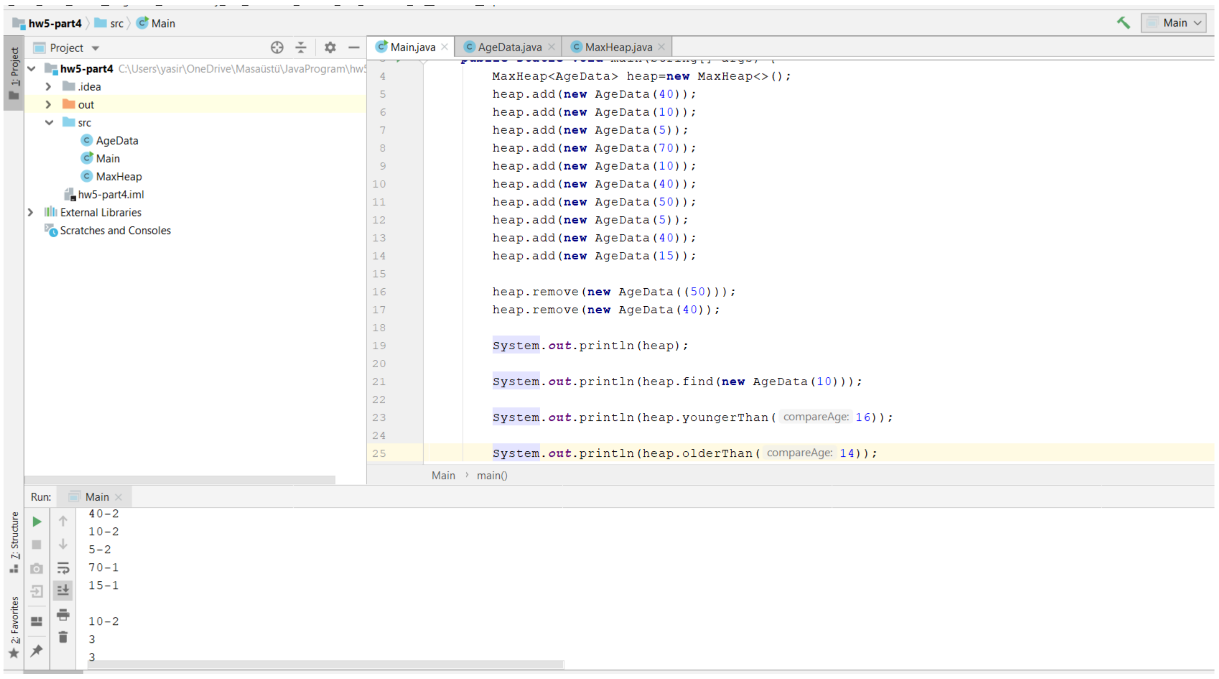
Add method take a AgeData as argument and add this to maxHeap.If there is already this element in tree it increase its number_age by 1 and modify maxHeap structure.If there is not this item in tree already it just add this item to tree and modfy maxHeap.

Remove method take a AgeData as argument. If this argument alreay in tree it just decreate its number_age by 1 and modify maxHeap. If number age became 0 than it delete this node and modify maxHeap.

Find method take a AgeData as argument and find this node and return this node.If it cant find this node so just return null.

YoungerThan method has a integer argument and traverse the tree. It return number of nodes that smaller age than argument.

olderThan method same as YoungerThan method it just return bigger nodes.

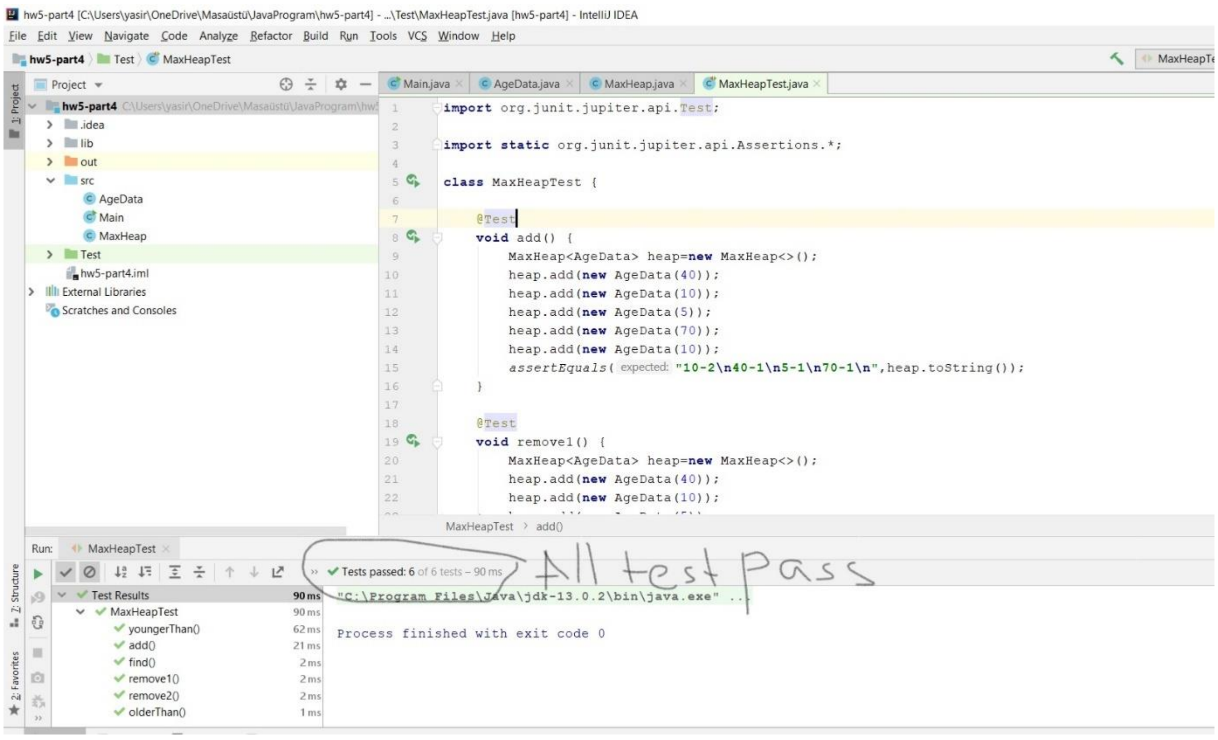


```java
        MaxHeap<AgeData> heap=new MaxHeap<>();
        heap.add(new AgeData(40));
        heap.add(new AgeData(10));
        heap.add(new AgeData(5));
        heap.add(new AgeData(70));
        heap.add(new AgeData(10));
        heap.add(new AgeData(40));
        heap.add(new AgeData(50));
        heap.add(new AgeData(5));
        heap.add(new AgeData(40));
        heap.add(new AgeData(15));

        heap.remove(new AgeData((50)));
        heap.remove(new AgeData(40));

        System.out.println(heap);

        System.out.println(heap.find(new AgeData(10)));

        System.out.println(heap.youngerThan( compareAge: 16));

        System.out.println(heap.olderThan( compareAge: 14));
```

```
40-2
10-2
5-2
70-1
15-1

10-2
3
3
```

Above example I create my heap and add element. After that remove 50 and 40 then print heap.Then call find(AgeData(10)) it return 10-2 and I print it. After that call youngerThan(16) and olderThan(14) both of them print 3.

I use Junit5 for test cases. You can find my test methods in Test Directory.



```java
import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.*;

class MaxHeapTest {

    @Test
    void add() {
        MaxHeap<AgeData> heap=new MaxHeap<>();
        heap.add(new AgeData(40));
        heap.add(new AgeData(10));
        heap.add(new AgeData(5));
        heap.add(new AgeData(70));
        heap.add(new AgeData(10));
        assertEquals( expected: "10-2\n40-1\n5-1\n70-1\n",heap.toString());
    }

    @Test
    void remove1() {
        MaxHeap<AgeData> heap=new MaxHeap<>();
        heap.add(new AgeData(40));
        heap.add(new AgeData(10));
```

```
Tests passed: 6 of 6 tests – 90 ms

Test Results      90 ms
  MaxHeapTest     90 ms
    youngerThan() 62 ms
    add()         21 ms
    find()         2 ms
    remove1()      2 ms
    remove2()      2 ms
    olderThan()    1 ms

"C:\Program Files\Java\jdk-13.0.2\bin\java.exe" ...

Process finished with exit code 0
```

All test Pass

Test ID	Test Scenerio	Test Steps	Test Data	Expected Result	Actual Result	Pass/Fail

T01	Add elements	1.Create heap
		2.Add AgeData(40)
		3.Add AgeData(10)
		4.Add AgeData(5)
		5.Add AgeData(70)
		6.Add AgeData(10)
		7.Print Heap
			AgeData(40)
			AgeData(10)
			AgeData(5)
			AgeData(70)
			AgeData(10)
				10-2
				40-1
				5-1
				70-1
					As expected	Pass

T02	Remove an element thas has 1 number of person in that age	1.Create heap
		2.Add AgeData(40)
		3.Add AgeData(10)
		4.Add AgeData(5)
		5.Add AgeData(70)
		6.Add AgeData(10)
		7.Remove AgeData(70)
		8.Print heap
			AgeData(40)
			AgeData(10)
			AgeData(5)
			AgeData(70)
			AgeData(10)
				10-2
				40-1
				5-1
					As Expected	Pass

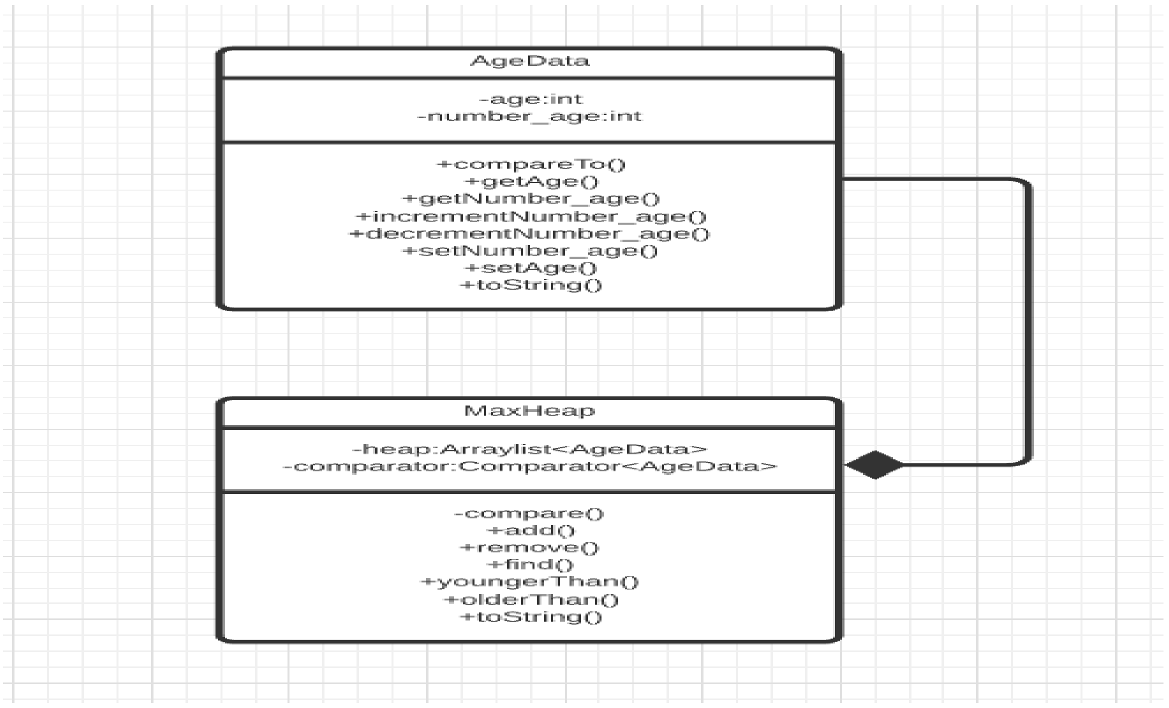
T03	Remove an element that has more than 1 person of that age	1.Create heap
		2.Add AgeData(40)
		3.Add AgeData(10)
		4.Add AgeData(5)
		5.Add AgeData(70)
		6.Add AgeData(10)
		7.Remove AgeData(10)
		8.Print heap
			AgeData(40)
			AgeData(10)
			AgeData(5)
			AgeData(70)
			AgeData(10)
				10-1
				40-1
				5-1
				70-1
					As expected	Pass

T04	Find an element in heap	1.Create heap
		2.Add AgeData(40)
		3.Add AgeData(10)
		4.Add AgeData(5)
		5.Add AgeData(70)
		6.Add AgeData(10)
		7.call find(AgeData(10))
			AgeData(40)
			AgeData(10)
			AgeData(5)
			AgeData(70)
			AgeData(10)
				10-2	As expected	Pass

T05	Find youngerThan number	1.Create heap
		2.Add AgeData(40)
		3.Add AgeData(10)
		4.Add AgeData(5)
		5.Add AgeData(70)
		6.Add AgeData(10)
		7.call youngerThan(15)
			AgeData(40)
			AgeData(10)
			AgeData(5)
			AgeData(70)
			AgeData(10)
				2	As Expected	Pass

T06	Find olderThan number	1.Create heap
		2.Add AgeData(40)
		3.Add AgeData(10)
		4.Add AgeData(5)
		5.Add AgeData(70)
		6.Add AgeData(10)
			AgeData(40)
			AgeData(10)
			AgeData(5)
			AgeData(70)
			AgeData(10)
				3	As Expected	Pass

		7.call olderThan(8)				


AgeData
-age:int
-number_age:int
+compareTo()
+getAge()
+getNumber_age()
+incrementNumber_age()
+decrementNumber_age()
+setNumber_age()
+setAge()
+toString()
MaxHeap
-heap:Arraylist<AgeData>
-comparator:Comparator<AgeData>
-compare()
+add()
+remove()
+find()
+youngerThan()
+olderThan()
+toString()