GIT Department of Computer Engineering

CSE 222/505 - Spring 2020

Homework #5 Report

Elif Goral – 171044003

Q1)

Problem Solution Approach

In the first question, I created a file system hierarchy consisting of FileSystemTree, FileNode and Main class. While creating my FileNode, I kept a string variable to keep the name of the file or folder in it, a List <FileNode> type to see the children of the node I am currently in, a parent of the node I am currently in, and the number of children of the node I am currently in.

When we come to my FileSystemTree class, I only have the root variable of type FileNode. This root shows the root of my tree. I created three kinds of constructors. The first one takes root as a parameter and my new root is root come as a parameter. My second constructor takes String data as a parameter. I create root with the data given in this constructor. In my third and final constructor, I assign null to root.

My AddFile method takes String as a parameter and it expresses the string path. In the method, I split the path according to the ‘/ 'sign and throw it into an array. If the penultimate element of the path array is file, it sends an exception stating that it cannot put a file on the file. If the penultimate element is not a file, it checks if root is null. If null, it adds direct path sequentially. But if it is not null, it scans the tree in order and tries to find the path and adds it when it finds it.

My AddDir method also takes the String parameter in the same way as addFile. The parameter that it takes expresses path and we divide path into pieces. Then, we check If the penultimate element of the path array is file, it sends an exception stating that it cannot put a directory on the file. If not, we check if the tree is empty. If it is empty, the specified path is added to the tree in order. If not, we are looking for the path and adding it to the place we find.

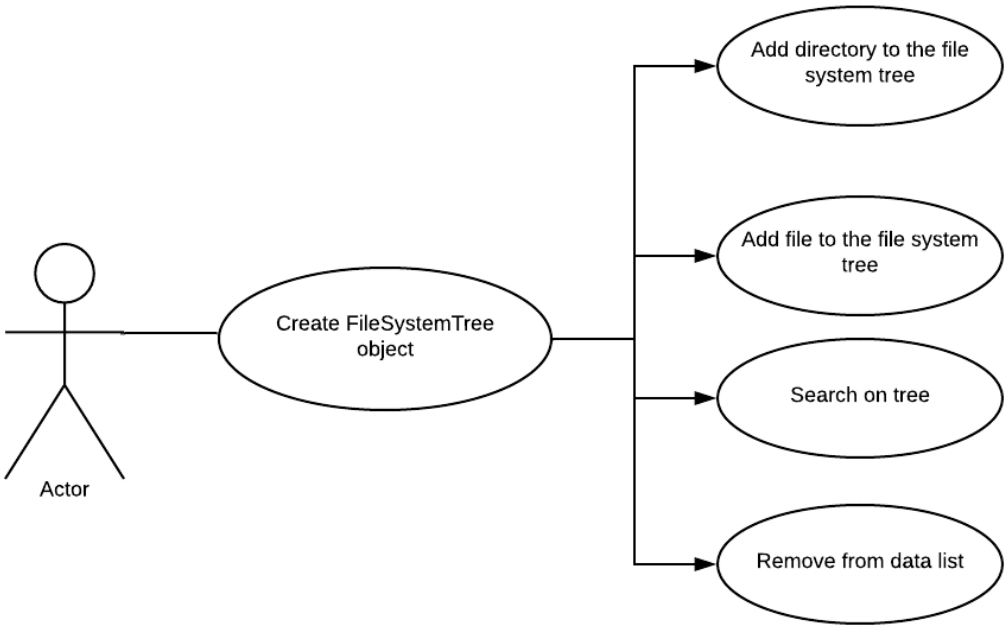
My Remove method also takes the path as a string, and we divide it into an array. Then we find the element to be deleted by travel around the

tree. If there is no child, we delete it directly. If there is a child, we ask the user if he should delete it.

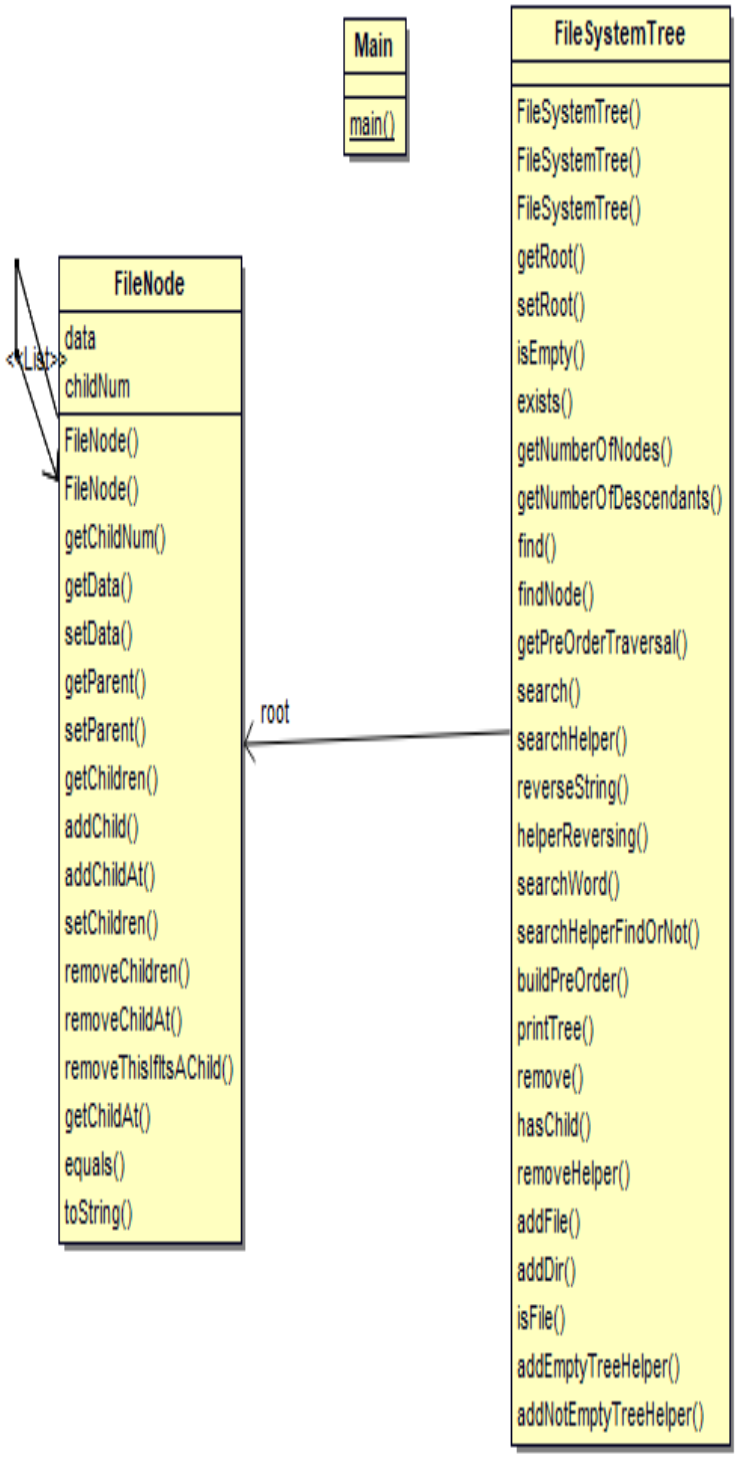
My search method takes the string .This string is the word to search. Then we start scanning the tree. We send it to our function which is searchHelperFindOrNot, which checks if it is in node or nor . If he found, he went to his parent and returned the full path. Since it adds in reverse, we reverse and print the string. In this way, we go around the whole tree.

The Find method also checks found or not. It returns true if it found it, and false if it did not find it.

Use Case Diagram



Class Diagram



Test Cases and Results

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Results	Actual Results	Pass/ Fail
T01	*Check one parameter constructor with a valid path*	*Call the one parameter constructor with a valid path*	*"root"*	"root"	"root"	*Pass*
T02	*Check addDir method call with a valid path*	*Call the method with a valid path*	*"root/first_directory"*	*Root* *First_directory*	*Root* *First_directory*	pass
T03	Check addDir method another subtree from root.	*Call the method with a valid path*	*"Root/second_directory"*	*root* *first_directory* *second_directory*	*root* *first_directory* *second_directory*	*pass*
T04	Check addFile Method above the directory with a valid path	*Call the method with a valid path*	*"Root/first_directory/ New_file.txt"*	*root* *first_directory* *new_file.txt* *second_directory*	*root* *first_directory* *new_file.txt* *second_directory*	pass
T05	Check addDir method above the directory with a valid path	Call the method with a valid path	*"Root/second_directory /new_directory"*	*root* *first_directory* *new_file.txt* *second_directory* new_directory	*root* *first_directory* *new_file.txt* *second_directory* new_directory	pass
T06	Check addFile method above the directory with a valid path	Call the method with a valid path	*"Root/second_directory /new_directory/ New_file.doc"*	*root* *first_directory* *new_file.txt* *second_directory* new_directory new_file.doc	*root* *first_directory* *new_file.txt* *second_directory* new_directory new_file.doc	pass
T07	Check the addDir method above the file with a valid path	Call the method with a valid path	"Root/first_directory/ New_file.txt/ aa_directory"	*Exception: we can not add directory after txt/doc file*	*Exception: we can not add directory after txt/doc file*	pass

T08	Check the addFile method above the file with a valid path.	Call the method with a valid path	*"Root/second_directory /new_directory/ New_file.doc/aa.txt"*	*Exception: we can not add file after txt/doc file*	*Exception: we can not add file after txt/doc file*	pass
T09	Check the search method	Call the search method with a valid string	"new"	file - root/first_directory/new_file.txt/ dir - root/second_directory/new_directory/ file - root/second_directory/new_directory/new_file.doc/	file - root/first_directory/new_file.txt/ dir - root/second_directory/new_directory/ file - root/second_directory/new_directory/new_file.doc/	pass
T10	Check the remove method with invalid path which is root	Call the remove method with invalid path which is root.	"root"	Exception: root can not be remove.	Exception: root can not be remove.	pass
T11	Check the remove method with valid path.	Call the remove method with valid path	"root/first_directory/ New_file.txt"	*root* *first_directory* *second_directory* *new_directory* *new_file.doc*	*root* *first_directory* *second_directory* *new_directory* *new_file.doc*	pass
T12	Check the remove method with removed node of tree has subtree,program give us choice. If we say yes,program removed specified node and its subtree,other wise removed nothing.	Call the remove method with valid path	"root/second_directory/ new_directory"	*Directory which you want to delete has child file or directories* *Do you want to continue the delete process?* *y: yes* *n: no* ***y ->*** *root* *first_directory* *second_directory* ***n->*** *delete process is canceled.* *root* *first_directory* *second_directory* *new_directory* *new_file.doc*	*Directory which you want to delete has child file or directories* *Do you want to continue the delete process?* *y: yes* *n: no* ***y ->*** *root* *first_directory* *second_directory* ***n->*** *delete process is canceled.* *root* *first_directory* *second_directory* *new_directory* *new_file.doc*	pass

T01 :

```
constructing with a root...
printing the tree...
root
```

T02:

```
add directory method with argument 'root/first_directory'
printing the tree...
root
first_directory
```

T03:

```
add directory method with argument 'root/second_directory'
printing the tree...
root
first_directory
second_directory
```

T04:

```
add file method with argument 'root/first_directory/new_file.txt'
printing the tree...
root
first_directory
new_file.txt
second_directory
```

T05:

```
add directory method with argument 'root/second_directory/new_directory'
printing the tree...
root
first_directory
new_file.txt
second_directory
new_directory
```

T06:

```
add file method with argument 'root/second_directory/new_directory/new_file.doc'
printing the tree...
root
first_directory
new_file.txt
second_directory
new_directory
new_file.doc
```

T07:

```
add directory method with argument 'root/first_directory/new_file/aa_directory'
java.lang.Exception: we can not add directory after txt/doc file
printing the tree...
root
first_directory
new_file.txt
second_directory
new_directory
new_file.doc
```

T08:

```
add file method with argument 'root/second_directory/new_directory/new_file.doc/aa.txt'
java.lang.Exception: we can not add file after txt/doc file
printing the tree...
root
first_directory
new_file.txt
second_directory
new_directory
new_file.doc
```

T09:

```
searching new...
file -root/first_directory/new_file.txt/
dir -root/second_directory/new_directory/
file -root/second_directory/new_directory/new_file.doc/
```

T10:

```
removing root..
java.lang.Exception: root can not be remove!
printing the tree...
root
first_directory
new_file.txt
second_directory
new_directory
new_file.doc
```

T11:

```
removing root/first_directory/new_file.txt
printing the tree...
root
first_directory
second_directory
new_directory
new_file.doc
```

T12:

```
removing root/second_directory/new_directory
Directory which you want to delete has child file or directories
Do you want to continue the delete process?
y: yes
n: no
y
input: y
printing the tree...
root
first_directory
second_directory
```

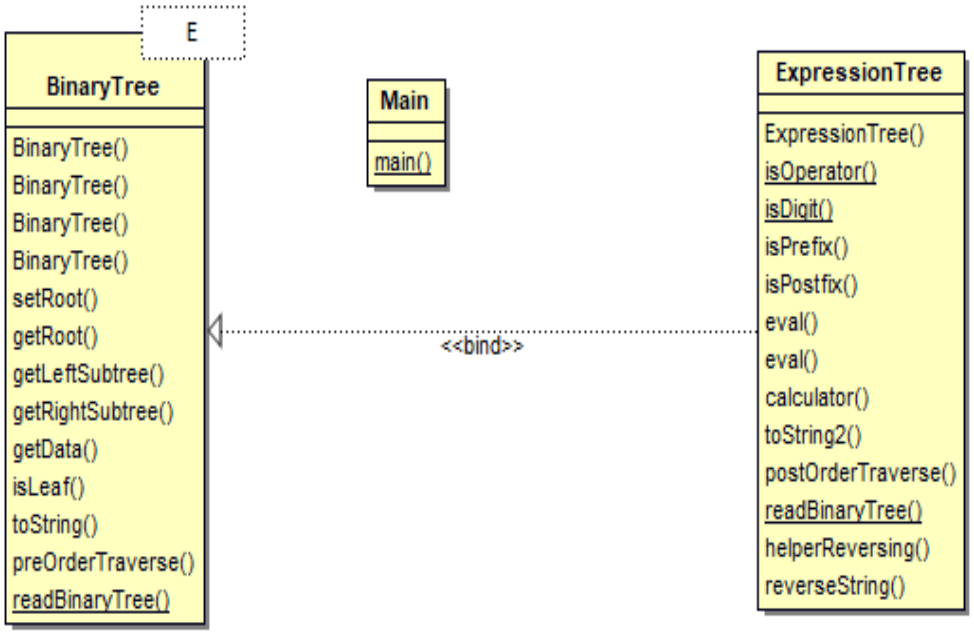
```
removing root/second_directory/new_directory
Directory which you want to delete has child file or directories
Do you want to continue the delete process?
y: yes
n: no
n
input: n
delete process is canceled.
printing the tree...
root
first_directory
second_directory
new_directory
new_file.doc
```

Q2)

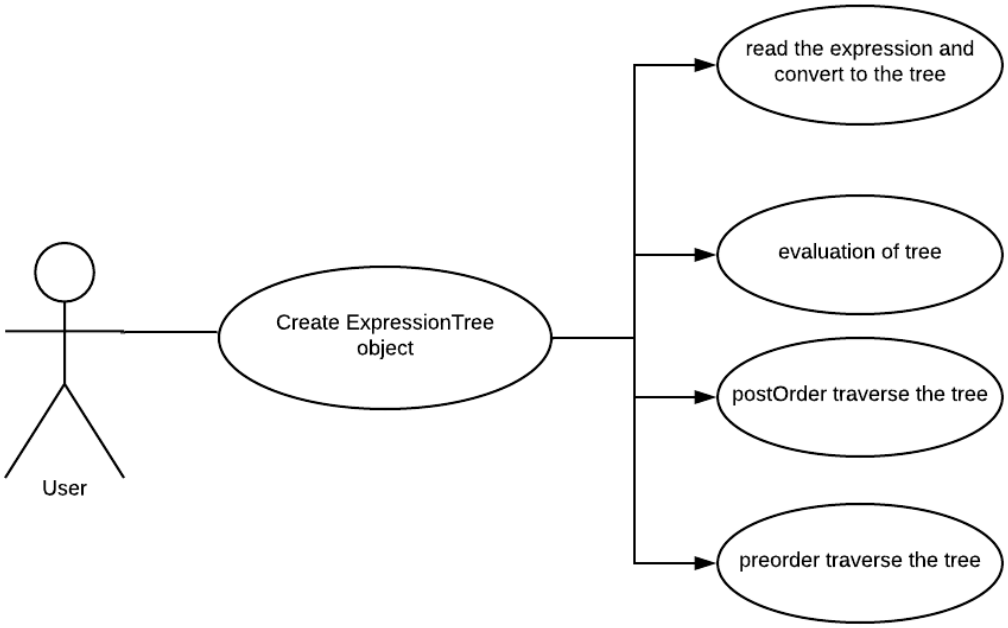
Problem Solution Approach

In second question, firstly I create a constructor which takes a string which Express the our expression. On constructor, I control the expression is prefix or not. If expression is prefix, I send the expression to the readBinaryTree method and assgin its root's to our root. If expression is postfix, firstly I reverse the expression and send to readBinaryTree method and assign its root's to our root.Thus I create a tree. In readBinaryTree method, I control the character is digit or not. If character is digit , I create a new subtree which has no leftchild and rightchild. Otherwise read and write to the tree. Then evalute the expression on tree.

Class Diagram



Use Case Diagram



Test Cases and Results

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Results	Actual Results	Pass/ Fail
T01	Check one parameter constructor with a valid path	Call the one parameter constructor with a valid path.	"+ + 10 * 5 15 20"	expression is prefix Expression : + + 10 * 5 15 20	expression is prefix Expression : + + 10 * 5 15 20	Pass
T02	Check toString2 method with prefix expression.	Call the toString2 method with prefix expression.	"+ + 10 * 5 15 20"	postOrderTraverse: 10 5 15 * + 20 +	postOrderTraverse: 10 5 15 * + 20 +	Pass

T03	*Check the eval method*	*Call the eval method after converting prefix expression to the tree.*	***"+ + 10 * 5 15 20"***	*eval : 105*	*eval : 105*	**Pass**
T04	***Check one parameter constructor with a valid path***	*Call the one parameter constructor with a valid path.*	*" * + 2 3 / + 4 5 3 "*	expression is prefix Expression : * + 2 3 / + 4 5 3	expression is prefix Expression : * + 2 3 / + 4 5 3	**Pass**
T05	***Check toString2 method with prefix expression.***	*Call the toString2 method with prefix expression.*	*" * + 2 3 / + 4 5 3 "*	postOrderTraverse: 2 3 + 4 5 + 3 / *	postOrderTraverse: 2 3 + 4 5 + 3 / *	**Pass**
T06	*Check the eval method*	*Call the eval method after converting prefix expression to the tree.*	*" * + 2 3 / + 4 5 3 "*	*eval : 15*	*eval : 15*	**Pass**
T07	***Check one parameter constructor with a valid path***	*Call the one parameter constructor with a valid path.*	*"10 5 15 * + 20 +"*	*expression is postfix Expression : 10 5 15 * + 20 +*	*expression is postfix Expression : 10 5 15 * + 20 +*	**Pass**
T08	***Check toString2 method with postfix expression.***	*Call the toString2 method with postfix expression*	*"10 5 15 * + 20 +"*	*preOrderTraverse : + 20 + * 15 5 10*	*preOrderTraverse : + 20 + * 15 5 10*	**Pass**
T09	*Check the eval method*	*Call the eval method after converting postfix expression to the tree.*	*"10 5 15 * + 20 +"*	*eval : 105*	*eval : 105*	**Pass**
T10	***Check one parameter constructor with a valid path***	*Call the one parameter constructor with a valid path.*	*"2 3 + 4 5 + 3 / * "*	*expression is postfix Expression : 2 3 + 4 5 + 3 / **	*expression is postfix Expression : 2 3 + 4 5 + 3 / **	**pass**
T11	***Check toString2 method with postfix expression.***	*Call the toString2 method with postfix expression*	*"2 3 + 4 5 + 3 / * "*	*preOrderTraverse : * / 3 + 5 4 + 3 2*	*preOrderTraverse : * / 3 + 5 4 + 3 2*	**pass**
T12	*Check the eval method*	*Call the eval method after converting postfix expression to the tree.*	*"2 3 + 4 5 + 3 / * "*	*eval : 15*	*eval : 15*	**pass**

Test data 1:

```
expression is prefix
Expression         : + + 10 * 5 15 20
postOrderTraverse: 10 5 15 * + 20 +
eval               : 105
```

Test data 2:

```
expression is prefix
Expression         : * + 2 3 / + 4 5 3
postOrderTraverse: 2 3 + 4 5 + 3 / *
eval               : 15
```

Test data 3:

```
expression is postfix
Expression         : 10 5 15 * + 20 +
preOrderTraverse : + 20 + * 15 5 10
eval               : 105
```

Test data 4:

```
expression is postfix
Expression         : 2 3 + 4 5 + 3 / *
preOrderTraverse : * / 3 + 5 4 + 3 2
eval               : 15
```

Q3)

Problem Solution Approach

In third question, I create AgeData, AgeSearchTree, BinaryTree, BinarySearchTree and Main class and SearchTree interface. My AgeData class implements comparable<AgeData>. And that class contains age and numberOfPerson data fields. And lastly has compareTo method which is compare the ages. İf our age greater than parameter object's age, returns 1,if our age smaller than parameter object's age, returns -1,other wise return 0.

On my AgeSearchTree class, I have no data field. My constructor only assign null to our root. Then I override a add method. Which takes generic data. And I have add method helper which is firstly control the data instance of AgeData or not. After that control the node is null or not. İf it is null, addReturn become true and returns new node. İf it is not null, we compare the two data and if compareTo method returns 0,which means that age already exist, Then I increase the number of person. If compare result is smaller than 0, I add to left, otherwise I add right.

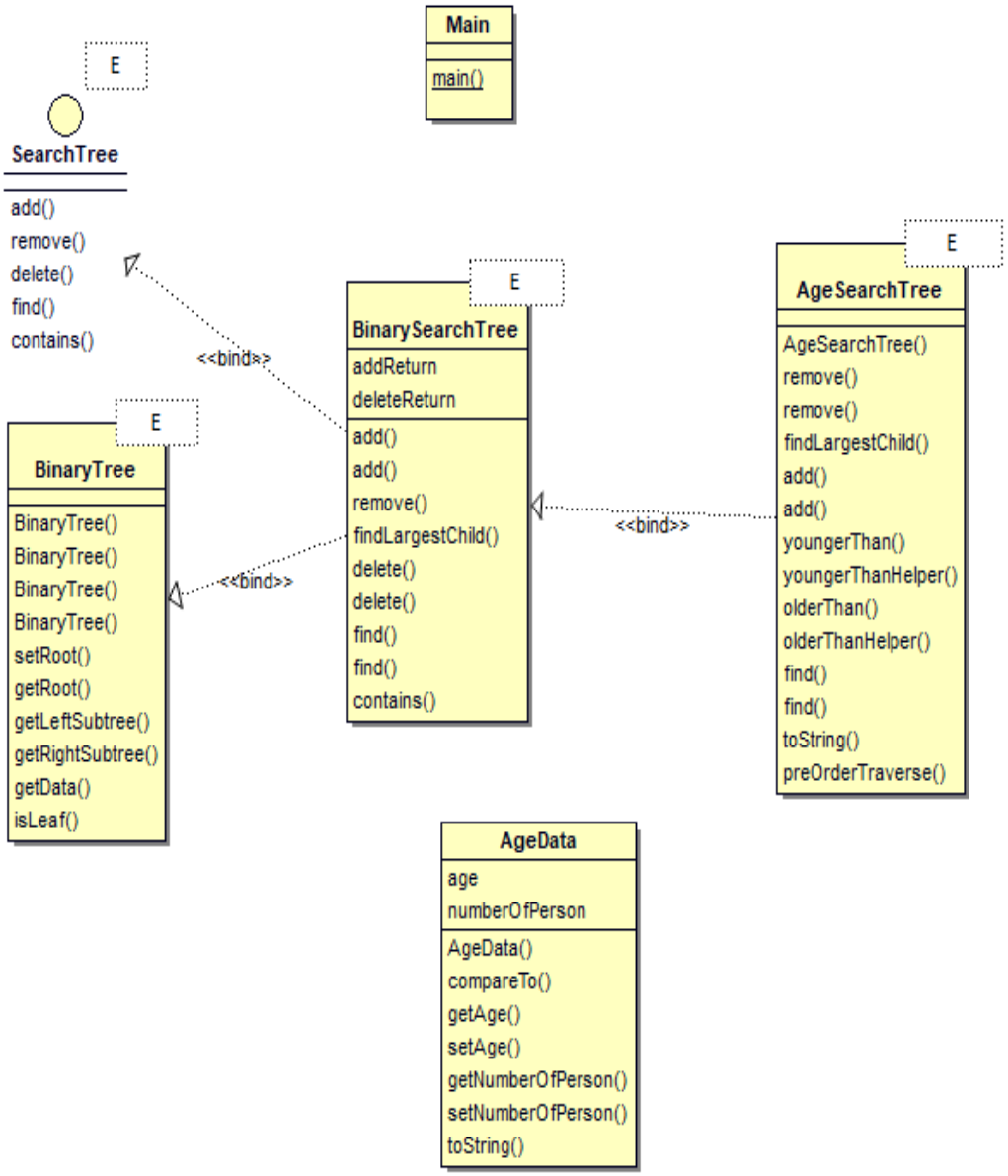
On my remove method, I take a generic data.And I have a helper method for remove. İf node is null, deleteReturn becomes null and we throw an exception which Express the data is not on tree. Otherwise we calculate the compare result with two data. İf compare result is smaller than 0, I call the method again started the current node's left child. İf compare result is greater than 0, I call the method again started the current node's right child. Otherwise which means I found the data. And check the number of data is equal to 1 or not. If it is not equal, I ust decrease the number of person,Otherwise I check current node's left child is null or not.if it is null,return right child, if node's right child is null, I return leftchild, otherwise which means our current node has left and right child, I control the leftchild's right child is null or not. İf it is null our new node is our left node. Otherwise I will find largest child and my new node will be that.

On my youngerThan method, takes an integer value. That method send the age to my helper method of youngerthan method. It checks the all

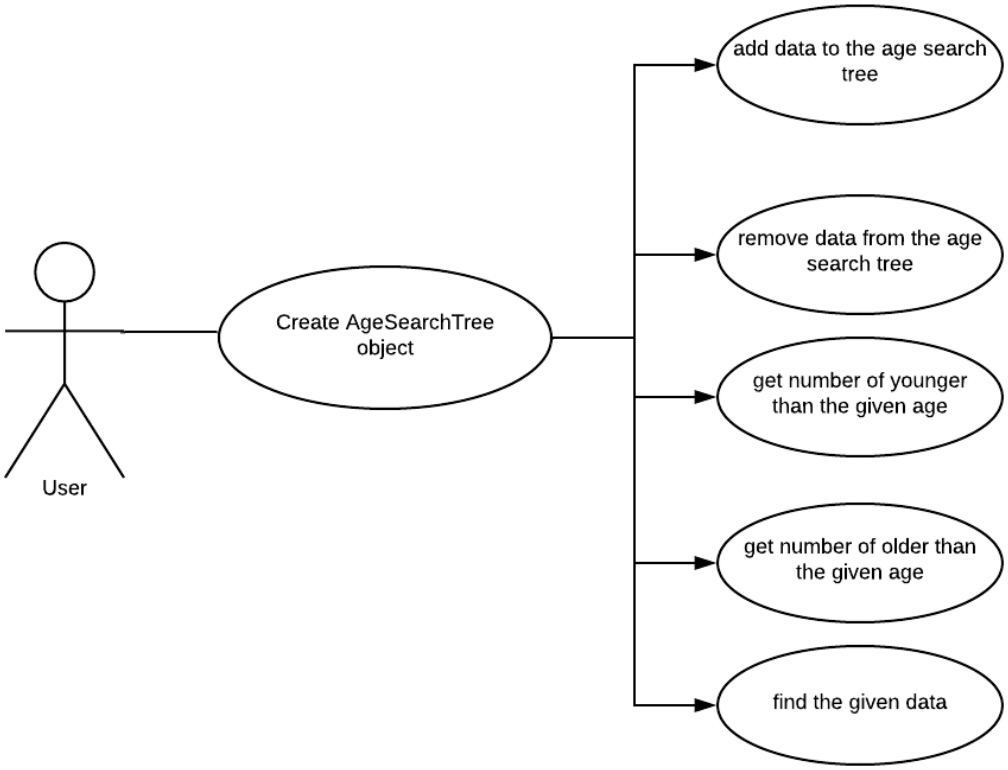
tree. If there is a age younger than given age,increase the younger number and end of the tree it returns the younderCounter. My olderThan method Works samely.

My last important method is find. That method travel the tree and use compareTo method. compareTo result is equal to 0, returns current node, if compareTo result is smaller than 0,returns current node's left child, otherwise returns current node's right child. If data is not on tree,throws an exception which Express the data is not on tree.

Class Diagram



Use Case Diagram



Test cases and results

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Results	Actual Results	Pass/ Fail
T01	*Check the constructor*	*Call the constructor*	*No data*	*No result*	*No result*	*pass*

T02	*Check the toString() method when the tree is empty*	*Call the toString() method when tree is empty.*	*No data*	*null*	*null*	*pass*
T03	*Check the add method with AgeData object*	*Call the add method with AgeData object*	*AgeData(10)*	*No result*	*No result*	*pass*
T04	*Check the toString() method after the add method*	*Call the toString() method after the add method*	*No data*	*10 – 1* *null* *null*	*10 – 1* *null* *null*	*pass*
T05	*Check the add method with AgeData object*	*Call the add method with AgeData object*	*AgeData(20)*	*No result*	*No result*	*pass*
T06	*Check the toString() method after the add method*	*Call the toString() method after the add method*	*No data*	*10 – 1* *null* *20 - 1* *null* *null*	*10 – 1* *null* *20 - 1* *null* *null*	*pass*
T07	*Check the add method with AgeData object*	*Call the add method with AgeData object*	*AgeData(5)*	*No result*	*No result*	*pass*
T08	*Check the toString() method after the add method*	*Call the toString() method after the add method*	*No data*	*10 – 1* *5 - 1* *null* *null* *20 - 1* *null* *null*	*10 – 1* *5 - 1* *Null* *null* *20 - 1* *null* *null*	*pass*
T09	*Check the add method with AgeData object*	*Call the add method with AgeData object*	*AgeData(15)*	*No result*	*No result*	*pass*
T10	*Check the toString() method after the add method*	*Call the toString() method after the add method*	*No data*	*10-1* *5-1* *null* *null* *20-1* *15-1* *null* *null* *null*	*10-1* *5-1* *null* *null* *20-1* *15-1* *null* *null* *null*	*pass*
T11	*Check the add method with AgeData object*	*Call the add method with AgeData object*	*AgeData(10)*	*No result*	*No result*	*pass*

T12	Check the toString() method after the add method	Call the toString() method after the add method	No data	10-2 5-1 null null 20-1 15-1 null null null	10-2 5-1 null null 20-1 15-1 null null null	pass
T13	Check the youngerThan() method with exist value	Call youngerThan method()	20	4	4	pass
T14	Check the olderThan() method with exist value	Call olderThan() method	20	0	0	pass
T15	Check the youngerThan() method with exist value	Call youngerThan method()	15	3	3	Pass
T16	Check the olderThan() method with exist value	Call olderThan() method	15	1	1	pass
T17	Check the youngerThan() method with exist value	Call youngerThan method()	10	1	1	Pass
T18	Check the olderThan() method with exist value	Call olderThan() method	10	2	2	pass
T19	Check the youngerThan() Method with exist value	Call youngerThan method()	5	0	0	Pass
T20	Check the olderThan() method with exist value	Call olderThan() method	5	4	4	pass
T21	Check the youngerThan() Method with no exist value	Call youngerThan method()	7	1	1	Pass
T22	Check the olderThan() method with no exist value	Call olderThan() method	7	4	4	Pass
T23	Check the find method with exist value	Call find method with exist value	AgeData(10)	10 - 2	10 - 2	Pass

T24	*Check the find method with no exist value*	*Call find method with no exist value*	*AgeData(2)*	*Exception: 2 - 1 is not on tree.*	*Exception: 2 - 1 is not on tree.*	*Pass*
T25	*Check the remove method with exist value*	*Call the remove method with exist value*	*AgeData(20)*	*No result*	*No result*	*Pass*
T26	*Check the toString method after remove method*	*Call the toString() method after remove method*	*No data*	*10-2 5-1 null null 15-1 null null*	*10-2 5-1 null null 15-1 null null*	*Pass*
T27	*Check the remove method with no exist value*	*Call the remove method with no exist value*	*AgeData(1)*	*No result*	*No result*	*Pass*
T28	*Check the toString method after remove method*	*Call the toString() method after remove method*	*No data*	*1 is not on tree.*	*1 is not on tree.*	*Pass*
T29	*Check the remove method with exist value whice has greater than one number of person.*	*Call the remove method with exist value*	*AgeData(10)*	*No result*	*No result*	*Pass*
T30	*Check the toString method after remove method*	*Call the toString() method after remove method*	*No data*	*10-1 5-1 null null 15-1 null null*	*10-1 5-1 null null 15-1 null null*	*Pass*
T31	*Check the remove method with exist value*	*Call the remove method with exist value*	*AgeData(10)*	*No result*	*No result*	*Pass*
T32	*Check the toString method after remove method*	*Call the toString() method after remove method*	*No data*	*5-1 null 15-1 null null*	*5-1 null 15-1 null null*	*Pass*

```
constructing with no parameter constructor.
Tree:
null
```

```
add AgeData(10)
Tree:
10-1
null
null

add AgeData(20)
Tree:
10-1
null
20-1
null
null

add AgeData(5)
Tree:
10-1
5-1
null
null
20-1
null
null
```

```
add AgeData(15)
Tree:
10-1
5-1
null
null
20-1
15-1
null
null
null

add AgeData(10)
Tree:
10-2
5-1
null
null
20-1
15-1
null
null
null
```

```
number of younger than 20 :  4

number of older than 20   :  0

number of younger than 15 :  3

number of older than 15   :  1

number of younger than 10 :  1

number of older than 10   :  2

number of younger than 5  :  0

number of older than 5    :  4

number of younger than 7  :  1

number of older than 7    :  4
```

```
finding 10....
10 - 2

finding 2....
java.lang.Exception: 2 - 1 is not on tree.
```

```
removing 1...
java.lang.Exception: 1 is not on tree.
Tree:
10-2
5-1
null
null
15-1
null
null

removing 10...
Tree:
10-1
5-1
null
null
15-1
null
null

removing 10...
Tree:
5-1
null
15-1
null
null
```

Q4)

Problem Solution Approach

In fourth question, I create AgeData, MaxHeap and Main classes. In AgeData class has age and numberOfPerson data fields and that class implements comparator<AgeData>. Accordingly I create a compare method too. In that method ; if first object's number of person is greater thean second object's number of person, returns 1; if first object's number of person is samller thean second object's number of person, returns -1;otherwise returns 0.

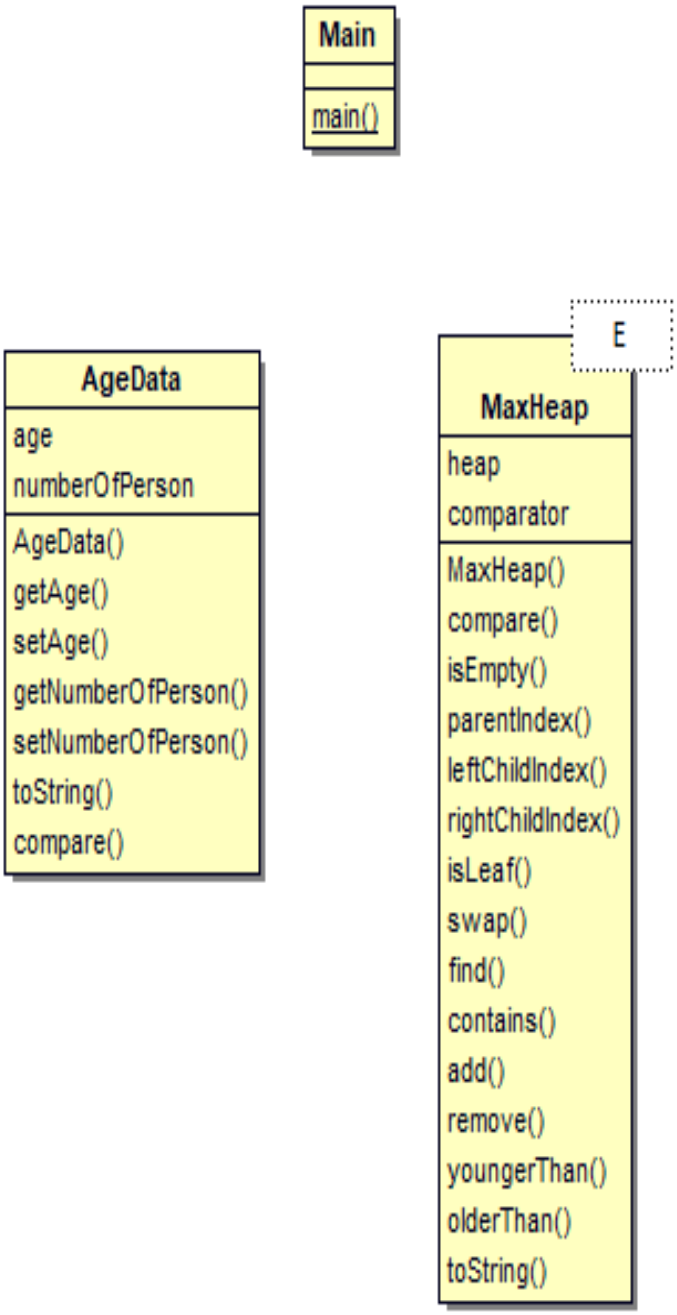
On my maxHeap class, I have a arraylist which Express the heap and comparator. For comparing, I create a compare method in maxHeap class too. In that class, I have an add method which takes a generic data. In that method firstly I check the data instanceof AgeData or not. İf is instanceof AgeData, I control heap's size is 0 or not. If it is 0, I add directly. Then I control the tree which contains the data or not. İf it contains the data, I start to travel the heap. When I found the data,decrease the numberofPerson. Otherwise I add the data to the heap. After that swap the values if the heap's current data is greater than parent's data until the root.

On my remove method, firstly I control the data instance of AgeData or not. İf data instanceof AgeData then control the tree contains the data or not. If tree does not contains the data, throw an exception which Express the data is not on tree.Otherwise, I travel the tree until find the data After found the data, if number of person is greater than 1, we just decrease the number of person. If the number of person is equal to 1, we put the most right bottom data to the removed node of tree, comparing the data to the upward. If compare result is 1, swap the values.

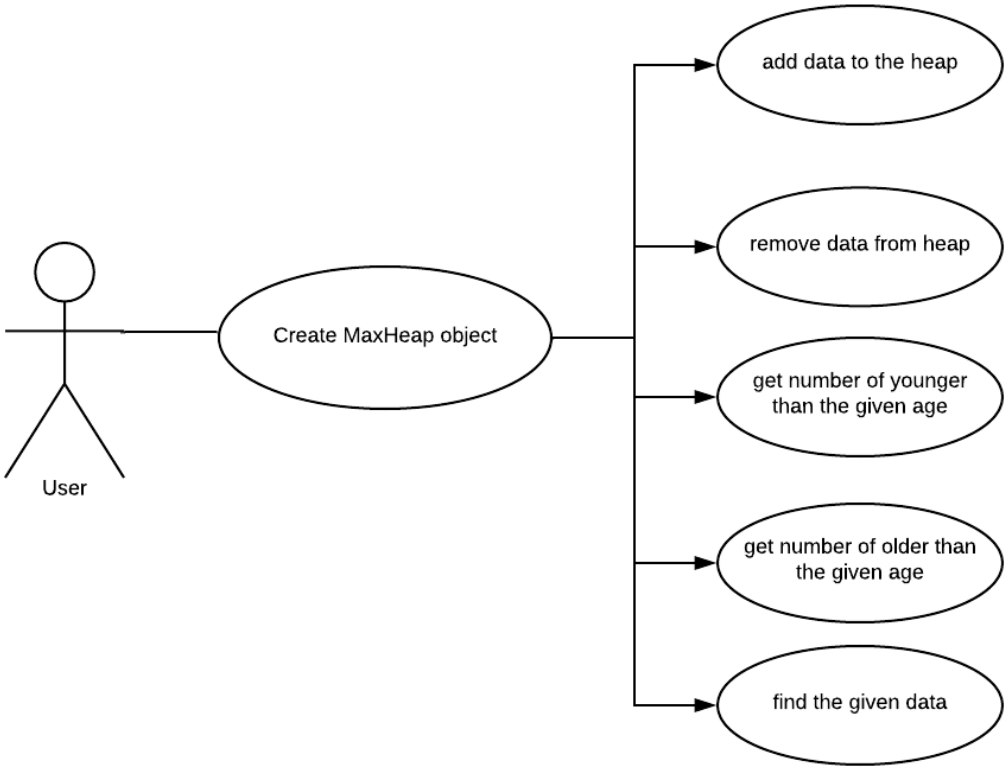
On my find method, I travel the tree. If I found the data, I return it; otherwse I throw an exception which Express the data is not on tree.

On my olderThan and youngerThan method, I travel the all tree and calculate the number of younger and older people, and return it.

Class Diagram



Use Case Diagram



Test Cases and Results

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Results	Actual Results	Pass/ Fail
T01	*Check the no parameter constructor*	*Call the no parameter constructor*	*No data*	*No result*	*No result*	*pass*
T02	*Check the add() method with valid argument*	*Call the add method with valid argument*	*AgeData(10)*	*No result*	*No result*	*pass*

T03	*Check the toString() method*	*Call the toString() method*	*No data*	*10 - 1*	*10 - 1*	*Pass*
T04	*Check the add() method with valid argument*	*Call the add method with valid argument*	*AgeData(5)*	*No result*	*No result*	*Pass*
T05	*Check the toString() method*	*Call the toString() method*	*No data*	*10 - 1* *5 - 1*	*10 - 1* *5 - 1*	*Pass*
T06	*Check the add() method with valid argument*	*Call the add method with valid argument*	*AgeData(70)*	*No result*	*No result*	*Pass*
T07	*Check the toString() method*	*Call the toString() method*	*No data*	*10 - 1* *5 - 1* *70 - 1*	*10 - 1* *5 - 1* *70 - 1*	*Pass*
T08	*Check the add() method with valid argument*	*Call the add method with valid argument*	*AgeData(10)*	*No result*	*No result*	*Pass*
T09	*Check the toString() method*	*Call the toString() method*	*No data*	*10 – 2* *5 - 1* *70 - 1*	*10 - 2* *5 - 1* *70 - 1*	*Pass*
T10	*Check the add() method with valid argument*	*Call the add method with valid argument*	*AgeData(50)*	*No result*	*No result*	*Pass*
T11	*Check the toString() method*	*Call the toString() method*	*No data*	*10 - 2* *5 - 1* *70 - 1* *50 - 1*	*10 - 2* *5 - 1* *70 - 1* *50 - 1*	*Pass*
T12	*Check the add() method with valid argument*	*Call the add method with valid argument*	*AgeData(5)*	*No result*	*No result*	*Pass*
T13	*Check the toString() method*	*Call the toString() method*	*No data*	*10 - 2* *5 – 2* *70 - 1* *50 - 1*	*10 - 2* *5 - 2* *70 - 1* *50 - 1*	*Pass*
T14	*Check the add() method with valid argument*	*Call the add method with valid argument*	*AgeData(15)*	*No result*	*No result*	*Pass*
T15	*Check the toString() method*	*Call the toString() method*	*No data*	*10 - 2* *5 - 2* *70 - 1* *50 - 1* *15 - 1*	*10 - 2* *5 - 2* *70 - 1* *50 - 1* *15 - 1*	*Pass*

T16	*Check the find method with exist value*	*Call the find method with exist value*	*AgeData(10)*	*10 - 2*	*10 - 2*	*Pass*
T17	*Check the find method with exist value*	*Call the find method with exist value*	*AgeData(70)*	*70 - 1*	*70 - 1*	*Pass*
T18	*Check the find method with no-exist value*	*Call the find method with no-exist value*	*AgeData(23)*	*Exception: 23-1 is not on heap*	*Exception: 23-1 is not on heap*	*Pass*
T19	*Check the youngerThan method with exist value*	*Call the youngerThan method with exist value*	*10*	*2*	*2*	*Pass*
T20	*Check the olderThan method with exist value*	*Call the olderThan method with exist value*	*10*	*3*	*3*	*Pass*
T21	*Check the youngerThan method with exist value*	*Call the youngerThan method with exist value*	*5*	*0*	*0*	*Pass*
T22	*Check the olderThan method with exist value*	*Call the olderThan method with exist value*	*5*	*5*	*5*	*Pass*
T23	*Check the youngerThan method with exist value*	*Call the youngerThan method with exist value*	*70*	*6*	*6*	*Pass*
T24	*Check the olderThan method with exist value*	*Call the olderThan method with exist value*	*70*	*0*	*0*	*Pass*
T25	*Check the youngerThan method with exist value*	*Call the youngerThan method with exist value*	*50*	*5*	*5*	*Pass*
T26	*Check the olderThan method with exist value*	*Call the olderThan method with exist value*	*50*	*1*	*1*	*Pass*
T27	*Check the youngerThan method with exist value*	*Call the youngerThan method with exist value*	*15*	*4*	*4*	*Pass*

T28	*Check the olderThan method with exist value*	*Call the olderThan method with exist value*	*15*	*2*	*2*	*Pass*
T29	*Check the youngerThan method with non exist value*	*Call the youngerThan method with non exist value*	*23*	*5*	*5*	*Pass*
T30	*Check the olderThan method with non exist value*	*Call the olderThan method with non exist value*	*23*	*2*	*2*	*Pass*
T31	*Check the remove method with exist value which has greater than one number of people*	*Call the removed method with exist value*	*AgeData(10)*	*No result*	*No result*	*Pass*
T32	*Check the toString() method after remove process*	*call the toString() method after remove process*	*No data*	*10 - 1* *5 - 2* *70 - 1* *50 - 1* *15 - 1*	*10 - 1* *5 - 2* *70 - 1* *50 - 1* *15 – 1*	*Pass*
T33	*Check the remove method with exist value which has greater than one number of people*	*Call the removed method with exist value*	*AgeData(5)*	*No result*	*No result*	*Pass*
T34	*Check the toString() method after remove process*	*call the toString() method after remove process*	*No data*	*10 - 1* *5 - 1* *70 - 1* *50 - 1* *15 - 1*	*10 - 1* *5 - 2* *70 - 1* *50 - 1* *15 – 1*	*Pass*
T35	*Check the remove method with non exist value*	*Call the removed method with non exist value*	*AgeData(6)*	*Exception: 6 - 1 is not on tree*	*Exception: 6 - 1 is not on tree.*	*pass*
T36	*Check the toString() method after failed remove process*	*call the toString() method after failed remove process*	*No data*	*10 - 1* *5 - 1* *70 - 1* *50 - 1* *15 – 1*	*10 - 1* *5 - 1* *70 - 1* *50 - 1* *15 – 1*	*Pass*

T37	*Check the remove method with exist value which has one people*	*Call the remove method*	*AgeData(5)*	*No result*	*No result*	*pass*
T38	*Check the toString() method after removed process*	*Call the toString method after remove method*	*No data*	*10 - 1* *70 - 1* *50 - 1* *15 – 1*	*10 - 1* *70 - 1* *50 - 1* *15 – 1*	*pass*

```
Add AgeData(10)
heap:
10 - 1

Add AgeData(5)
heap:
10 - 1
5 - 1

Add AgeData(70)
heap:
10 - 1
5 - 1
70 - 1

Add AgeData(10)
heap:
10 - 2
5 - 1
70 - 1
```

```
Add AgeData(50)
heap:
10 - 2
5 - 1
70 - 1
50 - 1

Add AgeData(5)
heap:
10 - 2
5 - 2
70 - 1
50 - 1

Add AgeData(15)
heap:
10 - 2
5 - 2
70 - 1
50 - 1
15 - 1
```

```
find AgeData(10)
10 - 2

find AgeData(70)
70 - 1

find AgeData(23)
java.lang.Exception: 23 - 1 is not on heap.
23 - 1
```

```
number of younger than 10        :  2
number of older than 10          :  3

number of younger than 5         :  0
number of older than 5           :  5

number of younger than 70        :  6
number of older than 70          :  0

number of younger than 50        :  5
number of older than 50          :  1

number of younger than 15        :  4
number of older than 15          :  2
```

```
removing 10
heap:
10 - 1
5 - 2
70 - 1
50 - 1
15 - 1

removing 5
heap:
10 - 1
5 - 1
70 - 1
50 - 1
15 - 1

removing 6
java.lang.Exception: 6 - 1 is not on tree.
heap:
10 - 1
5 - 1
70 - 1
50 - 1
15 - 1

removing 5
heap:
10 - 1
15 - 1
70 - 1
50 - 1
```