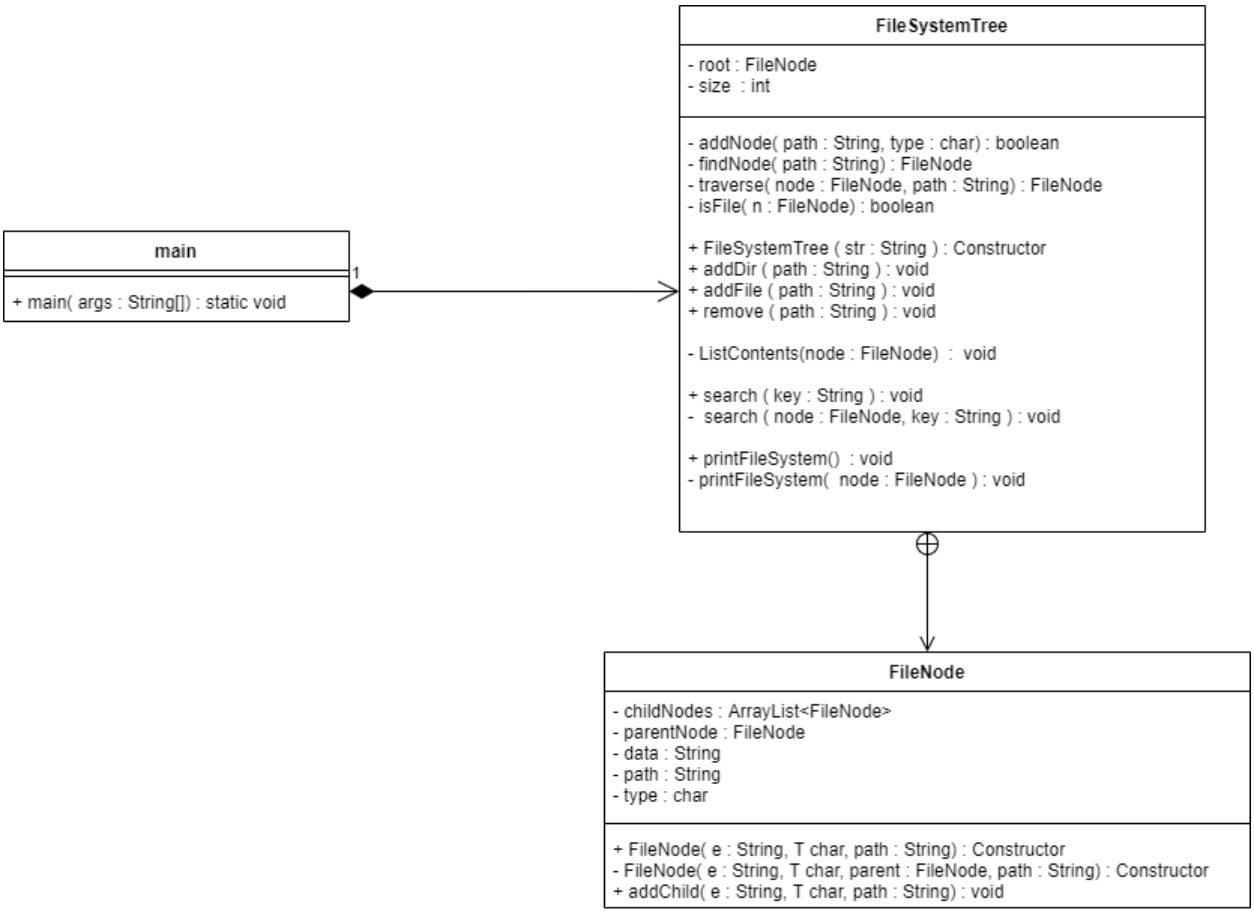
GIT Department of Computer Engineering
CSE 222/505 - Spring 2020
Homework 5 Report

Muhammed Eyüp

1801042679

Q1

Class Diagram:



Problem solutions approach:

In this question the problem was to find a way to organize a file system and the best way was to organize it using a tree where the outer directory (folder) is the parent node and the files are the leaves of the tree and each node has a type which identifies it as being a directory or a file and each node has an array list to store all of its children and a node to link it to its parent.

The system also enables the user to modify the system by adding and removing a node provided it doesn't exist in the same path when adding and it exists in the path when removing

The system also warns the user when attempting to remove a non-empty directory by listing all the contents of that directory and asking for confirmation.

Test cases:

Test Scenario	Test Case	Result
Adding a new node	Node doesn't exist in the path	Adds a new node
	Node already exists in the path	Doesn't add a node and gives a warning
	Add a node after a file	
Removing a node	Node exists in the path and doesn't have children	Deletes node
	Node doesn't exist in the path	Throws NoSuchElement Exception
	Node exists in the path but has children	Lists Children and asks for confirmation
Search for nodes with given key	Node with given key exists in the tree	Prints the node type and its path
	Node doesn't exist in the tree	Returns
Print the tree	Print the tree	Prints the tree

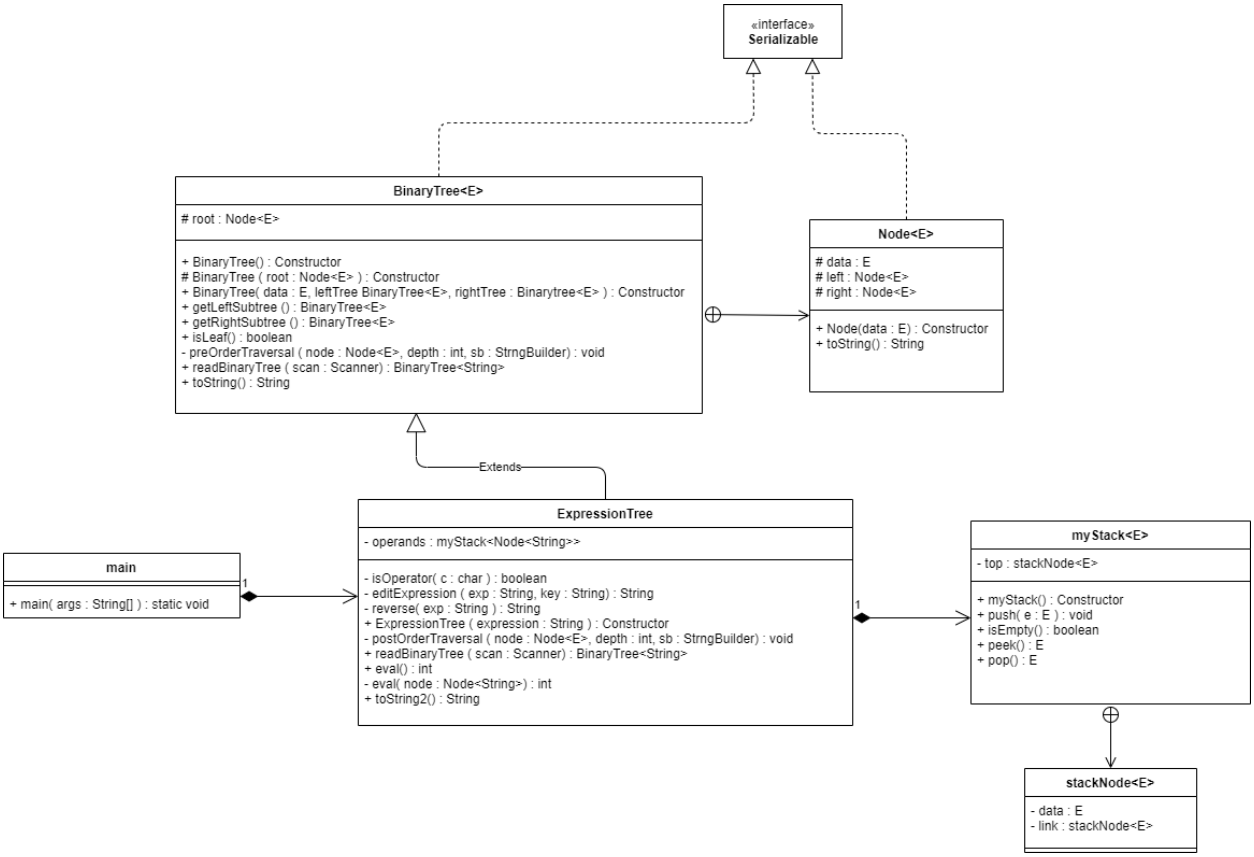
Running commands and results:

```
Directory added successfully: root
Directory added successfully: root/new_directory
Directory added successfully: root/Second_new_Directory
Directory added successfully: root/Second_new_Directory/AB
Directory/File already exists in the path: root/Second_new_Directory/AB
File added successfully: root/new_file.txt
File added successfully: root/new_directory/new_doc_file.doc
File added successfully: root/Second_new_Directory/c
Searching for files with the word new
dir - root/new_directory
File - root/new_directory/new_doc_file.doc
dir - root/Second_new_Directory
File - root/new_file.txt

root new_directory new_doc_file.doc Second_new_Directory AB c new_file.txt

Removing root/Second_new_Directory...

Warning : Folder not empty
Contents
AB c
Remove Directory Y/N: y

Removing root/new_directory/new_doc_file.doc...

root new_directory new_file.txt
```

```
Directory added successfully: root
Directory added successfully: root/new_directory
Directory added successfully: root/Second_new_Directory
Directory added successfully: root/Second_new_Directory/AB
Directory/File already exists in the path: root/Second_new_Directory/AB
File added successfully: root/new_file.txt
File added successfully: root/new_directory/new_doc_file.doc
File added successfully: root/Second_new_Directory/c
Searching for files with the word new
dir - root/new_directory
File - root/new_directory/new_doc_file.doc
dir - root/Second_new_Directory
File - root/new_file.txt

root new_directory new_doc_file.doc Second_new_Directory AB c new_file.txt

Removing root/Second_new_Directory...

Warning : Folder not empty
Contents
AB c
Remove Directory Y/N: n

Removing root/new_directory/new_doc_file.doc...

root new_directory Second_new_Directory AB c new_file.txt
```

Q2

Class Diagram:



Problem solutions approach:

The problem was to create an expression tree that can handle both prefix and postfix expressions and to do that we used a binary tree such that every operator is a node that has two children nodes and these children are leaves if they are operands or branches if they are operators

Test cases:

Test Scenario	Test case	Result
Adding nodes	Postfix expression	Expression is divided and each piece is added as a node
	Prefix expression	Expression is reversed the added to the tree
Evaluating an expression	Evaluating an expression	Returns the result as an integer
Printing the tree	Preorder traverse	Traverses the tree in preorder manner and prints the nodes
	Post order traverse	Traverses the tree in post order manner and prints the nodes

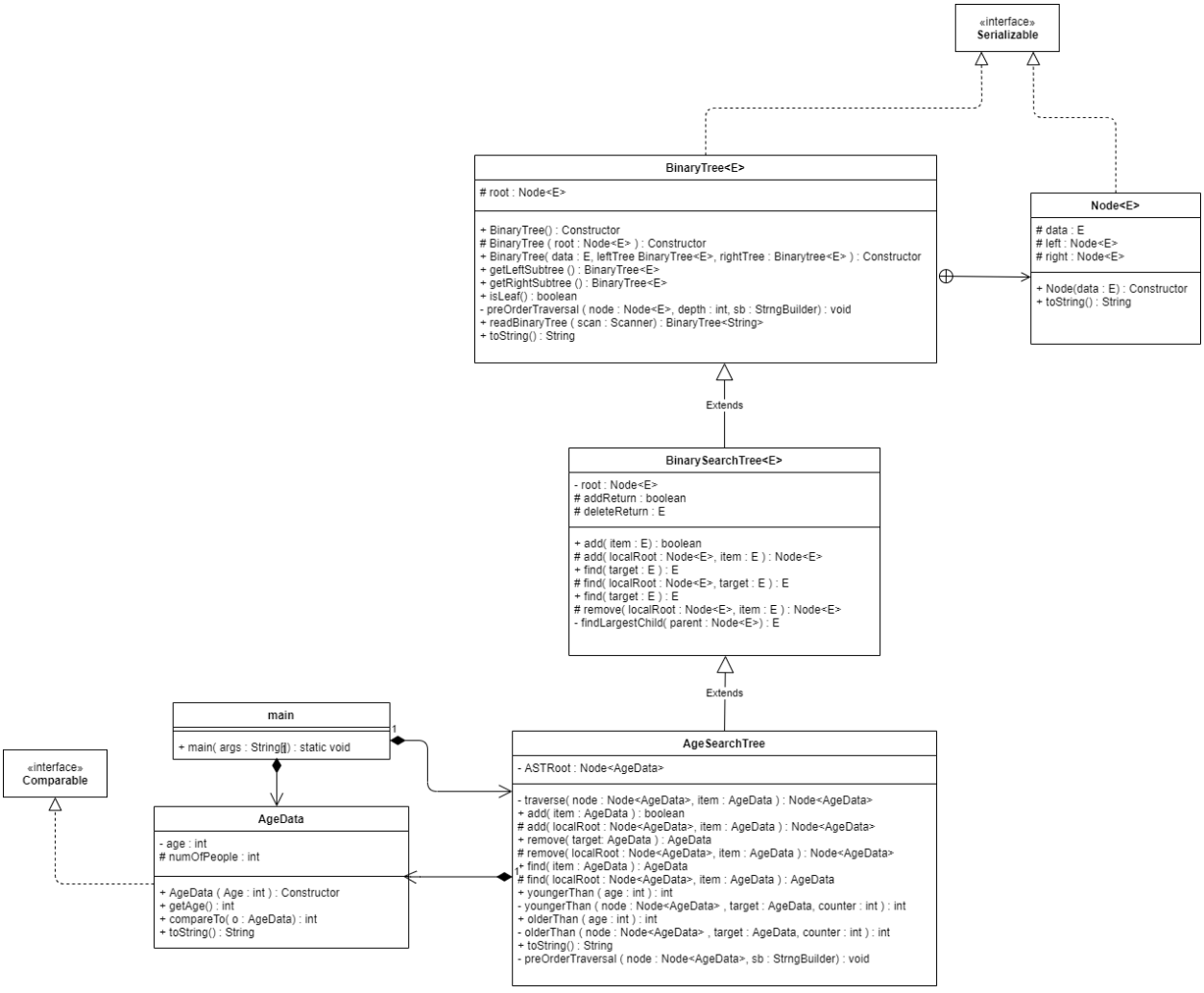
Running commands and results:

```
preorder traverse:
+
 +
  10
   null
   null
  *
   5
    null
    null
   15
    null
    null
 20
  null
  null
```

```
postorder traverse:
      null
   null
10
         null
    null
5
       null
    null
15
*
+
   null
  null
20
+

result : 105
```

Q3

Class diagram:



Problem solutions approach:

In this question we used a binary search tree to group people with different ages the tree allows us to sort them find an age group in a short time and compare age groups

Test Cases:

Test Scenario	Test Case	Result
Adding an age group	age group doesn't exist	Adds a new node with that age group in the appropriate place in the tree and sets the number of people to 1
	Age group already exists	Finds the node of that age group in the tree and increase the number of people by 1
Removing an age group	age group doesn't exist	Throws a NoSuchElement exception
	Age group has more than 1 person	Decreases number of people in that age group by 1
	Age group has 1 person	Deletes the node
Finding an age group	age group doesn't exist	Throws a NoSuchElement exception
	age group exists	Returns the age groups data
Printing an age group	Printing an age group	Prints the age and number of people

Running commands and results:

```
Successfully added age group: 4

Successfully added age group: 12

Successfully added age group: 17

Age group 12 already exists number of people increased by 1:(Age:12)
number of people younger than 15 is: 3
Age group has more than one person number of people decreased by 1:(Age:12)
Removed age group of age 17
12 - 1

15 - 1
4 - 1
null
12 - 1
null
null
null
```

```
Successfully added age group: 4

Successfully added age group: 12

Successfully added age group: 17

Age group 12 already exists number of people increased by 1:(Age:12)
number of people younger than 15 is: 3
Exception in thread "main" java.util.NoSuchElementException: Age group doesn't exist: 50

        at AgeSearchTree.remove(AgeSearchTree.java:92)
        at AgeSearchTree.remove(AgeSearchTree.java:59)
        at main.main(main.java:16)
```

```
Successfully added age group: 4

Successfully added age group: 12

Successfully added age group: 17

Age group 12 already exists number of people increased by 1:(Age:12)
number of people younger than 15 is: 3
Age group has more than one person number of people decreased by 1:(Age:12)
Removed age group of age 17
Exception in thread "main" java.util.NoSuchElementException: could not find Age group: 50

        at AgeSearchTree.find(AgeSearchTree.java:113)
        at AgeSearchTree.find(AgeSearchTree.java:99)
        at main.main(main.java:19)
```