# Project 4
# Binary Trees

By: Alexander J Sanna
Due: December 2nd, 2022
Computer Science 2400 - 002
015193607

# Project Specification:

This Project required the implementation of 4 separate interfaces. However, the BinaryTreeInterface extended both the Tree Interface as well as the Tree Iterator Interface. Below are the JavaDoc files for all three of these, along with Expression Tree Interface, an extension of Binary Tree Interface.

## Interface TreeInterface<T>

public interface **TreeInterface<T>**

### *Method Summary*

All Methods | Instance Methods | Abstract Methods

| Modifier and Type | Method | Description |
|---|---|---|
| void | clear() | clear() scrubs the tree clean |
| int | getHeight() | getHeight() returns the number of levels of the tree. |
| int | getNumberOfNodes() | getNumberOfNodes() counts the number of Nodes in the tree. |
| T | getRootData() | getRootData() this method will return the reference held within the tree. |
| boolean | isEmpty() | isEmpty() checks if there is data within the tree. |

### *Method Details*

#### getRootData

```
T getRootData()
```

getRootData() this method will return the reference held within the tree.

**Returns:**
T the data reference held within the tree

#### getHeight

```
int getHeight()
```

getHeight() returns the number of levels of the tree.

**Returns:**
int number of levels

#### getNumberOfNodes

```
int getNumberOfNodes()
```

getNumberOfNodes() counts the number of Nodes in the tree.

**Returns:**
int, number of nodes in tree.

#### isEmpty

```
boolean isEmpty()
```

isEmpty() checks if there is data within the tree.

**Returns:**
boolean: if the tree contains data.

#### clear

```
void clear()
```

clear() scrubs the tree clean

## Interface BinaryTreeInterface<T>

**All Superinterfaces:**
TreeInterface<T>, TreeIteratorInterface<T>

public interface **BinaryTreeInterface<T>**
extends TreeInterface<T>, TreeIteratorInterface<T>

### *Method Summary*

All Methods | Instance Methods | Abstract Methods

| Modifier and Type | Method | Description |
|---|---|---|
| void | setTree(T rootData) | SetTree method Allows for a tree to be created with root data T. |
| void | setTree(T rootData, BinaryTreeInterface<T> leftTree, BinaryTreeInterface<T> rightTree) | setTree method Allows for a tree to be made with 2 children. |

**Methods inherited from interface TreeInterface**
clear, getHeight, getNumberOfNodes, getRootData, isEmpty

**Methods inherited from interface TreeIteratorInterface**
getInorderIterator, getLevelOrderIterator, getPostorderIterator, getPreorderIterator

### *Method Details*

#### setTree

```
void setTree(T rootData)
```

SetTree method Allows for a tree to be created with root data T.

**Parameters:**
rootData -

#### setTree

```
void setTree(T rootData,
             BinaryTreeInterface<T> leftTree,
             BinaryTreeInterface<T> rightTree)
```

setTree method Allows for a tree to be made with 2 children.

**Parameters:**
rootData -
leftTree -
rightTree -

## Class ExpressionTree

java.lang.Object
  BinaryTree<String>
    ExpressionTree

**All Implemented Interfaces:**
BinaryTreeInterface<String>, ExpressionTreeInterface<String>, TreeInterface<String>, TreeIteratorInterface<String>

public class **ExpressionTree**
extends BinaryTree<String>
implements ExpressionTreeInterface<String>

### *Constructor Summary*

Constructors

| Constructor | Description |
|---|---|
| ExpressionTree(String[] postfix) | |

### *Method Summary*

All Methods | Instance Methods | Concrete Methods

| Modifier and Type | Method | Description |
|---|---|---|
| double | evaluate() | This is the evaluate method. |
| void | postorder() | |

**Methods inherited from class BinaryTree**
clear, getHeight, getInorderIterator, getLevelOrderIterator, getNumberOfNodes, getPostorderIterator, getPreorderIterator, getRootData, getRootNode, isEmpty, setRootNode, setTree, setTree

**Methods inherited from class java.lang.Object**
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

**Methods inherited from interface BinaryTreeInterface**
setTree, setTree

**Methods inherited from interface TreeInterface**
clear, getHeight, getNumberOfNodes, getRootData, isEmpty

**Methods inherited from interface TreeIteratorInterface**
getInorderIterator, getLevelOrderIterator, getPostorderIterator, getPreorderIterator

### *Constructor Details*

#### ExpressionTree

```
public ExpressionTree(String[] postfix)
```

### *Method Details*

#### evaluate

```
public double evaluate()
```

**Description copied from interface: ExpressionTreeInterface**
This is the evaluate method. every expression tree can evaluate to a final answer

**Specified by:**
evaluate in interface ExpressionTreeInterface<String>

**Returns:**
the value of evaluation

#### postorder

```
public void postorder()
```

All of these methods Specified by the interfaces were implemented in the classes described in the Following section.

Testing and Implementation:

For the implementation I ran with a Binary Node class, a binary tree class and an Expression tree class. The binary node served as the implementation of the binary tree class as it held references to the data of each binary tree as well as a reference to the tree's children. For the expression Tree, i had to utilize a stack to ensure that the constructor performed correctly and did its job in storing the correct data in the correct spot of the tree. This made the postorder method and the evaluate method that much easier.

What I learned?

I learned a lot about the tree adt as well as recursion, which was not at all easy for me. The recursive methods in this project easily took me over 10 hours to understand and implement, I just could not wrap my head around this.

However, in the end, it all worked out and I am really proud of how this project turned out.