**Data Structures Workspace Technical Manual**

* For individuals looking to continue development on the data structures workspace this manual will provide the most detail with regards to the design structure of certain implementations, UML diagrams for understanding how the various data structures should operate and interact with other classes, and further information on future development considerations. Note that not all diagrams are completed for each implemented. For certain data structures this is not necessary since the implementation is straightforward, but if future development requires increased complexity then more diagrams and models of the implementation may become necessary. See the table of contents, or the table of figures to navigate to specific data structures or UML diagrams.

Table of Contents

[Table of Figures 1](#_Toc99893914)

[Getting Started 1](#_Toc99893915)

[Purpose 1](#_Toc99893916)

[Structure of JUnit Tests 1](#_Toc99893917)

[Single Linked List (int) 1](#_Toc99893918)

[Double Linked List (int) 1](#_Toc99893919)

[Single Linked List (Generic) 1](#_Toc99893920)

[Double Linked List (Generic) 1](#_Toc99893921)

[Stack (Int) 1](#_Toc99893922)

[Queue (Int) 1](#_Toc99893923)

[Binary Search Tree (int) 1](#_Toc99893924)

[Data Sets Tested 1](#_Toc99893925)

[Use Case Diagrams 1](#_Toc99893926)

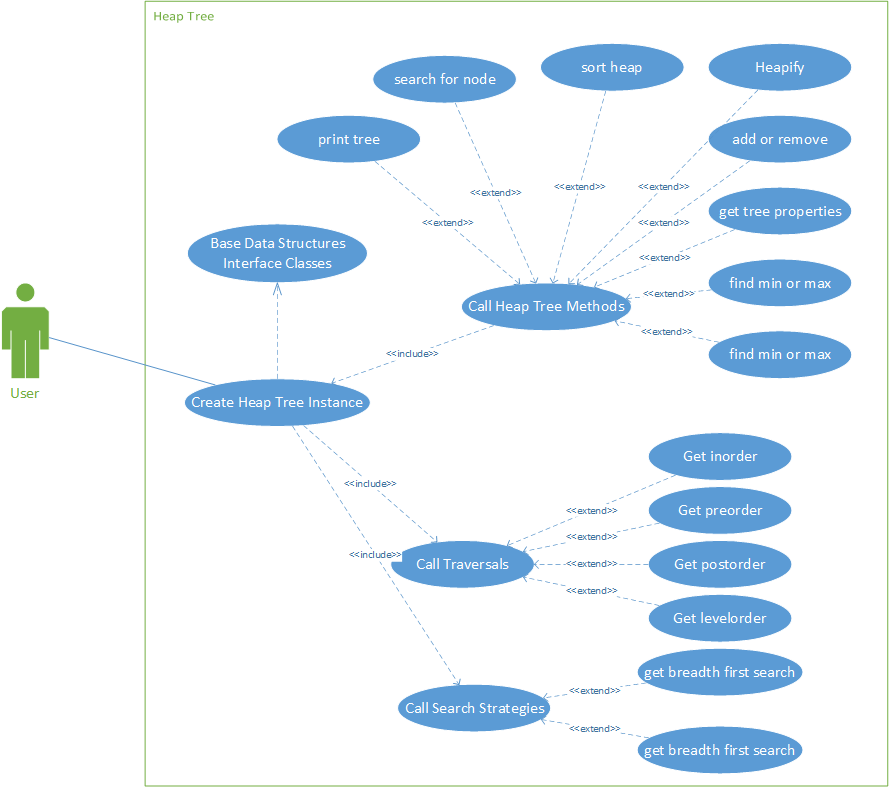
[Single Linked List 1](#_Toc99893927)

[Doubly Linked List 1](#_Toc99893928)

[Binary Search Tree 1](#_Toc99893929)

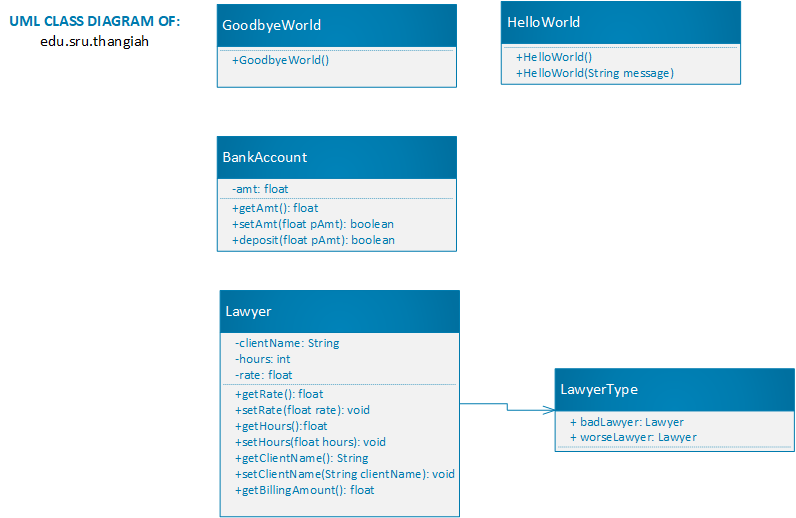
[General Tree 1](#_Toc99893930)

[Heap Tree 1](#_Toc99893931)

[ 1](#_Toc99893932)

[Class Diagrams: 1](#_Toc99893933)

[Edu.sru.thangiah 1](#_Toc99893934)

[ 1](#_Toc99893935)

[Edu.sru.thangiah.arrays 1](#_Toc99893936)

[Edu.sru.thangiah.datastructures 1](#_Toc99893937)

[Edu.sru.thangiah.datastructures.generic 1](#_Toc99893938)

[Edu.sru.thangiah.datastructures.generic.matrix 1](#_Toc99893939)

[Edu.sru.thangiah.datastructures.generic.queue 1](#_Toc99893940)

[Edu.sru.thangiah.datastructures.generic.stack 1](#_Toc99893941)

[Edu.sru.thangiah.datastructure.generic.linkedlist 1](#_Toc99893942)

[Edu.sru.thangiah.datastructures.generic.linkedlist 1](#_Toc99893943)

[Edu.sru.thangiah.datastructures.tree.binarytree 1](#_Toc99893944)

[edu.sru.thangiah.datastructures.generic.tree.binarytree 1](#_Toc99893945)

[Edu.sru.thangiah.datastructures.tree.generaltree 1](#_Toc99893946)

[edu.sru.thangiah.datastructures.tree.heaptree 1](#_Toc99893947)

[edu.sru.thangiah.datastructures.generic.tree.heaptree 1](#_Toc99893948)

[edu.sru.thangiah.datastructures.generic.tree.avltree 1](#_Toc99893949)

[edu.sru.thangiah.datastructures.tree.avltree 1](#_Toc99893950)

[edu.sru.thangiah.datastructures.tree.redblacktree 1](#_Toc99893951)

[edu.sru.thangiah.datastructures.generic.tree.redblacktree 1](#_Toc99893952)

[edu.sru.thangiah.datastructures.stack 1](#_Toc99893953)

[sru.edu.thangiah.datastructures.queue 1](#_Toc99893954)

[edu.sru.thangiah.datastructurs.matrix 1](#_Toc99893955)

[edu.sru.thangiah.datastructures.search.minimax 1](#_Toc99893956)

[edu.sru.thangiah.datastructures.hashtable 1](#_Toc99893957)

[edu.sru.thangiah.datastructures.vector 1](#_Toc99893958)

[edu.sru.thangiah.datastructures.inheritance 1](#_Toc99893959)

[edu.sru.thangiah.datastructures.interfaceex 1](#_Toc99893960)

[edu.sru.thangiah.datastructures.polymorphism 1](#_Toc99893961)

[edu.sru.thangiah.datastructures.recursion 1](#_Toc99893962)

[edu.sru.thangiah.datastructures.sorting 1](#_Toc99893963)

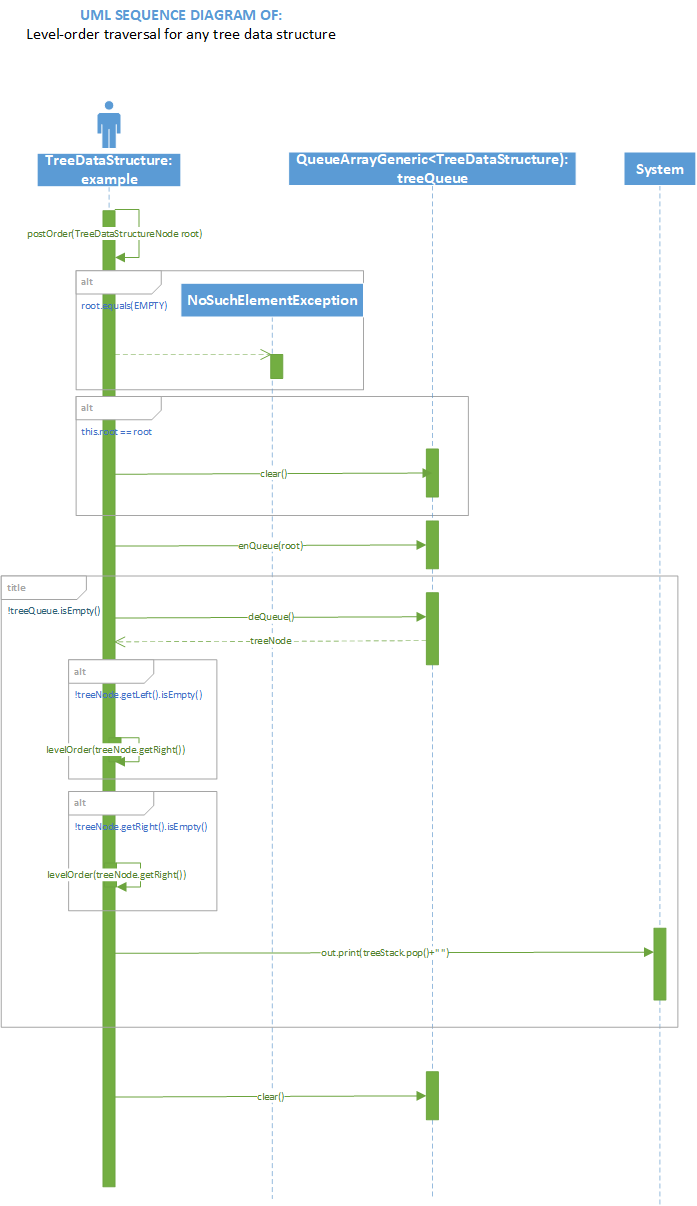
[Sequence Diagrams: 1](#_Toc99893964)

[In-Order Traversal for any Tree Data Structure 1](#_Toc99893965)

[Pre-Order Traversal for any Tree Data Structure 1](#_Toc99893966)

[Post-Order Traversal for any Tree Data Structure 1](#_Toc99893967)

[Level-Order Traversal for any Tree Data Structure 1](#_Toc99893968)

[ 1](#_Toc99893969)

[Hash Table Add Method (Array) 1](#_Toc99893970)

[Hash Table Add Method (ArrayList) 1](#_Toc99893971)

[Hash Table add method (Vector) 1](#_Toc99893972)

[UML Statechart Diagrams 1](#_Toc99893973)

[In-order Traversal for any Tree Data Structure 1](#_Toc99893974)

[Pre-Order Traversal for any Tree Data Structure 1](#_Toc99893975)

[Post-Order Traversal for any Tree Data Structure 1](#_Toc99893976)

[Level Order Traversal for any Tree Data Structure 1](#_Toc99893977)

[Hash Table Add Method for any Container (Array, ArrayList, Vector) 1](#_Toc99893978)

[UML Activity Diagrams 1](#_Toc99893979)

[In-Order Traversal for any Tree Data Structure 1](#_Toc99893980)

[Pre-Order Traversal for any Tree Data Structure 1](#_Toc99893981)

[Post-Order Traversal for any Tree Data Structure 1](#_Toc99893982)

[Level-Order Traversal for any Tree Data Structure 1](#_Toc99893983)

[Hash Table Add Method for any Container (Array, ArrayList, Vector) 1](#_Toc99893984)

# Table of Figures

[Figure 1: Use Case Diagram of Single Linked List 10](#_Toc100619532)

[Figure 2: Use Case Diagram of Doubly Linked List 11](#_Toc100619533)

[Figure 3: Use Case Diagram of Binary Search Tree 12](#_Toc100619534)

[Figure 4: Use Case Diagram of General Tree Implementation 13](#_Toc100619535)

[Figure 5 Heap Tree Use Case Diagram 14](#_Toc100619536)

[Figure 6 Class diagram of edu.sru.thangiah package 15](#_Toc100619537)

[Figure 7 Class diagram of edu.sru.thangiah.arrays 16](#_Toc100619538)

[Figure 8 Class diagram of edu.sru.thangiah.datastructures 17](#_Toc100619539)

[Figure 9 Class diagram of edu.sru.thangiah.datastructures.generic 18](#_Toc100619540)

[Figure 10 Class diagram of edu.sru.thangiah.datastructures.generic.matrix 19](#_Toc100619541)

[Figure 11 Class Diagram of edu.sru.thangiah.datastructures.generic.queue 20](#_Toc100619542)

[Figure 12 Class diagram of edu.sru.thangiah.datastructures.generic.stack 21](#_Toc100619543)

[Figure 13: Class Diagram of Int Single Linked List Implementation 22](#_Toc100619544)

[Figure 14: Class Diagram of Doubly Linked List Implementation 23](#_Toc100619545)

[Figure 15: Class Diagram of Generic Single Linked List Implementation 24](#_Toc100619546)

[Figure 16: Class Diagram of Generic Doubly Linked List Implementation 25](#_Toc100619547)

[Figure 17: Class Diagram of Int Binary Tree Implementation 26](#_Toc100619548)

[Figure 18: Class Diagram of Generic Binary Tree Implementation 27](#_Toc100619549)

[Figure 19: Class Diagram of Int General Tree Implementation 29](#_Toc100619550)

[Figure 20: Class Diagram of Heap Tree Implementation 31](#_Toc100619551)

[Figure 21: Class Diagram of Generic Heap Tree Implementation 32](#_Toc100619552)

[Figure 22: Class Diagram of Generic AVL Tree 33](#_Toc100619553)

[Figure 23:Class Diagram of Int AVL Tree 34](#_Toc100619554)

[Figure 24: Class Diagram of Int Red Black Tree Implementation 35](#_Toc100619555)

[Figure 25 Class Diagram of Generic Red Black Tree Implementation 36](#_Toc100619556)

[Figure 26 : Class diagram of int stack implementation 37](#_Toc100619557)

[Figure 27 : Class diagram of int queue implementation 38](#_Toc100619558)

[Figure 28 : Class diagram of int matrix implementation 39](#_Toc100619559)

[Figure 29 : Class diagram of Tic Tac Toe minimax implementation 40](#_Toc100619560)

[Figure 30 : Class diagram of the Vector Data Structure 41](#_Toc100619561)

[Figure 31 : Class diagram of the different examples of inheritance in the inheritance package 42](#_Toc100619562)

[Figure 32 : Class diagram of the interface example 43](#_Toc100619563)

[Figure 33 : Class diagram of the polymorphism example 44](#_Toc100619564)

[Figure 34 : Class diagram of the recursion example 44](#_Toc100619565)

[Figure 35 : Class diagram of the sorting package classes 45](#_Toc100619566)

[Figure 36 : Class diagram of int hash table implementation 46](#_Toc100619567)

[Figure 37 Sequence Diagram of In-Order Traversal for any Tree Data Structure 47](#_Toc100619568)

[Figure 38 Sequence Diagram of Pre-Order Traversal for any Tree Data Structure 48](#_Toc100619569)

[Figure 39 Sequence Diagram of Post Order Traversal for any Tree Data Structure 49](#_Toc100619570)

[Figure 40 Sequence Diagram of Level Order Traversal for any Tree Data Structure 50](#_Toc100619571)

[Figure 41: Sequence Diagram of Add Method for Array Hash Table Implementation 51](#_Toc100619572)

[Figure 42 Sequence Diagram of Hash Table Add Method for ArrayList Hash Table Implementation 52](#_Toc100619573)

[Figure 43 Sequence Diagram of Add Method for Vector Hash Table Implementation 53](#_Toc100619574)

[Figure 44 MiniMax Tic Tac Toe Sequence Diagram 54](#_Toc100619575)

[Figure 45 Statechart diagram of In-Order traversal for any Tree Data Structure Implementation 55](#_Toc100619576)

[Figure 46 State Chart Diagram of Pre-Order Traversal for any Tree Data Structure Implementation 56](#_Toc100619577)

[Figure 47 Statechart diagram of Post-Order Traversal for any Tree Data Structure Implementation 57](#_Toc100619578)

[Figure 48 Statechart diagram of Level Order Traversal for any Tree Data Structure Implementation 58](#_Toc100619579)

[Figure 49 Statechart Diagram of Add Method for any Hash Table Implementation (Array, ArrayList, Vector) 59](#_Toc100619580)

[Figure 50 State Chart Diagram of Minimax Tic Tac Toe Game 60](#_Toc100619581)

[Figure 51 Activity Diagram of In-Order Tree Traversal for any Tree Data Structure 61](#_Toc100619582)

[Figure 52 Activity Diagram of Pre-Order Traversal for any Tree Data Structure Implementation 62](#_Toc100619583)

[Figure 53 Activity Diagram of Post-Order Traversal for any Tree Data Structure Implementation 63](#_Toc100619584)

[Figure 54 Activity Diagram of Level-Order Traversal for any Tree Data Structure 64](#_Toc100619585)

[Figure 55 Activity Diagram of Hash Table Add Method for any Container (Array, ArrayList, Vector) 65](#_Toc100619586)

[Figure 56 Activity Diagram of Minimax Tic Tac Toe Game 66](#_Toc100619587)

# Getting Started

## Purpose

The Data Structures workspace contained in this repository provides a large number of Java implementations for commonly used and studied data structures. The intent is to help guide prospective users trying to learn or brush up on their knowledge of data structures by providing thoroughly documented code with default int and generic implementations. Users can expect to find generic and int implementations of arrays, stacks, matrices, vectors, sorting algorithms, linked lists, queues, binary search trees, heap trees, AVL trees, 2-3-4 trees, redblack trees, and hashing algorithms. Users can expect a wide variety of methods which demonstrate the useability of these data structures and also interface/abstract classes which can be inherited to easily build additional data structures examples.

# Structure of JUnit Tests

## Single Linked List (int)

The Single Linked List tests are implemented in order to test each method of the SingleLinkedList.java class. The types of lists created to test each case include an empty list, a populated list, and a list with a single node. The empty list tests adding and removal of nodes when there is no existing data. The populated list interacts with all methods to ensure they are working properly in a traditional scenario. The single node list ensures that removal from the structure resets the structure back to an empty list.

## Double Linked List (int)

The Double Linked List tests are implemented in order to test each method of the DoubleLinkedList.java class. The types of lists created to test each case include an empty list, a populated list, and a list with a single node. These tests are essentially the same as the Single Linked List tests, just more specific to Double Linked List methods and being able to retrieve previous node in the list.

## Single Linked List (Generic)

The Single Linked List tests are implemented in order to test each method of the SingleLinkedListGeneric.java class. The types of lists created include the original integer lists, along with lists to test String data nodes and double data nodes. These added data type lists are tested using the same methods as the integer list.

## Double Linked List (Generic)

The Double Linked List tests are implemented in order to test each method of the DoubleLinkedList.java class. The types of lists created to test each case include an empty list, a populated list, and a list with a single node. These tests are essentially the same as the Single Linked List tests, just more specific to Double Linked List methods and being able to retrieve previous node in the list.

## Stack (Int)

The Stack tests are implemented in order to test each method in the Stack.java class. The types of stacks created in order to test these methods thoroughly include an empty stack, a stack with one node, and a populated stack. The empty stack is used to test adding, removing, and getting nodes or data when there is none. The stack with one node ensures that the stack is reset when removing the single node. The populated stack is tested with all methods and cases for each method.

## Queue (Int)

The Queue tests are implemented in order to test each method in the QueueArray.java class. The types of queues created in order to test these methods thoroughly include an empty queue, a queue with one node, and a populated queue. The empty queue is used to test adding, removing, and getting nodes or data when there is none. The queue with one node ensures that the queue is reset when removing the single node. The populated queue is tested with all methods and cases for each method.

## Binary Search Tree (int)

The Binary Search Tree tests are laid out to test for all the methods in BinarySearchTree.java. The data sets used for our tests include an empty tree, a one node tree, an initialized tree with five nodes, a complete tree, a full tree, an empty tree used to test the add/insert methods, and a right/left leaning tree. Diagrams of how all of these trees look are included in this section. Since an empty tree simply does not have any initialized nodes, we omit the visualization of this tree.

# Data Sets Tested

The data sets and tree examples used in our JUnit tests are standardized across all data structure Junit tests to ensure uniformity among all the tests. Subsections for each data structure describes the kinds of initialized data structures tested for as well as diagrams depicting what these structures should look like.

# Use Case Diagrams

## Single Linked List

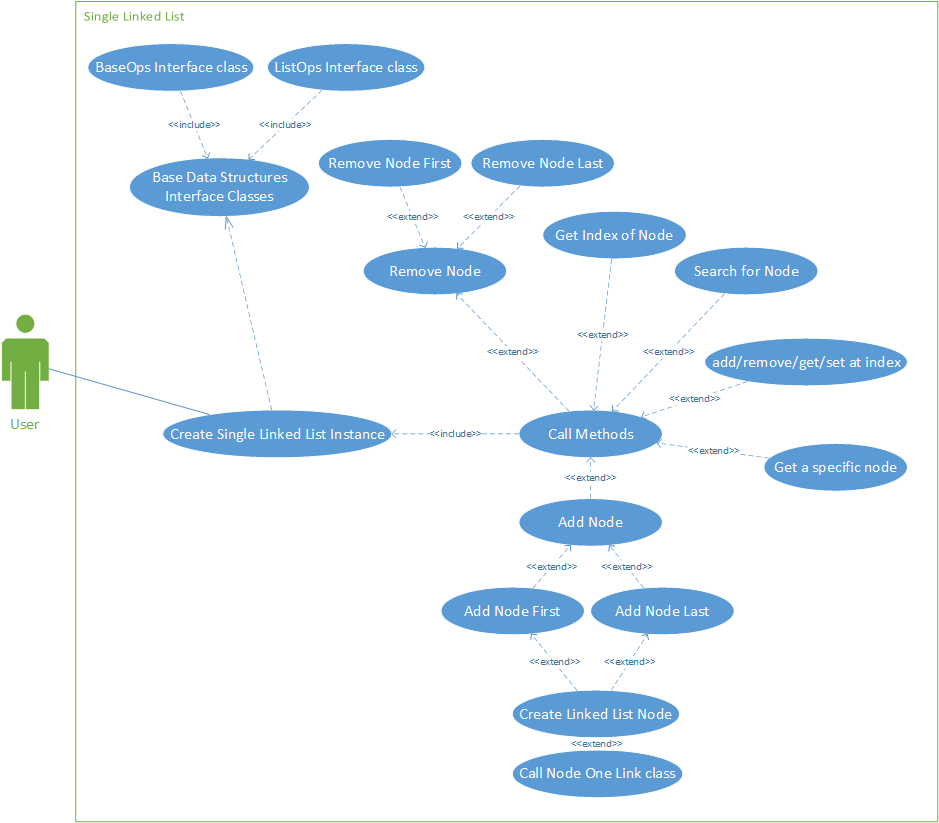
****

Figure 1: Use Case Diagram of Single Linked List

## Doubly Linked List

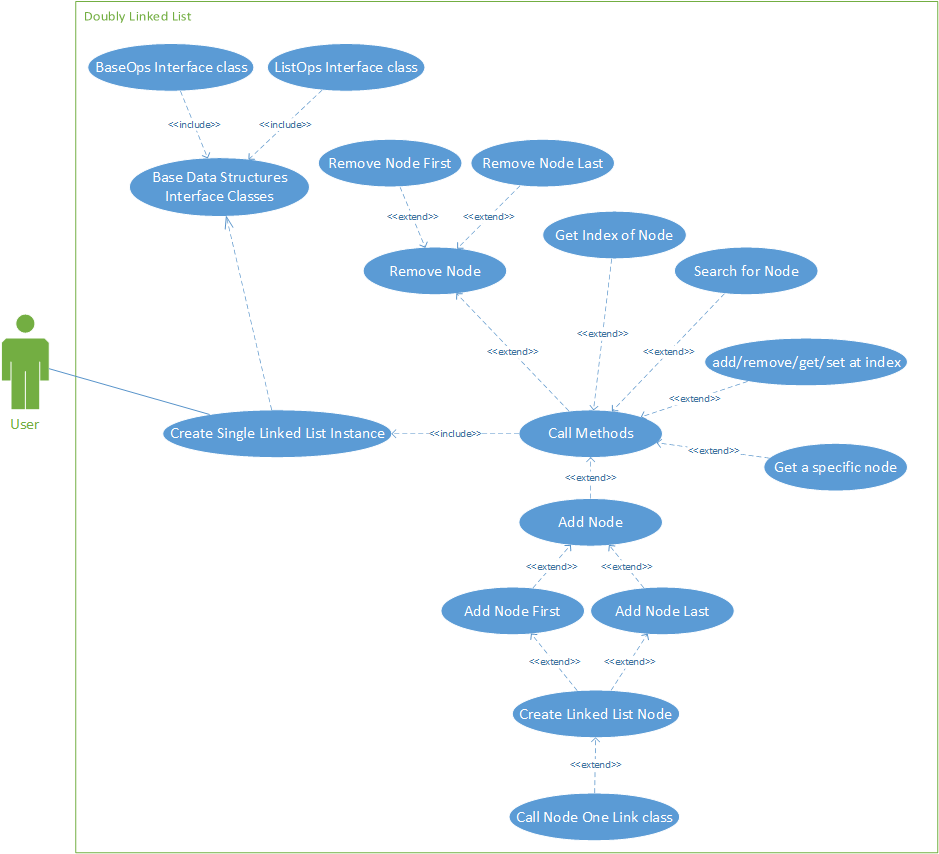
****

Figure 2: Use Case Diagram of Doubly Linked List

## Binary Search Tree

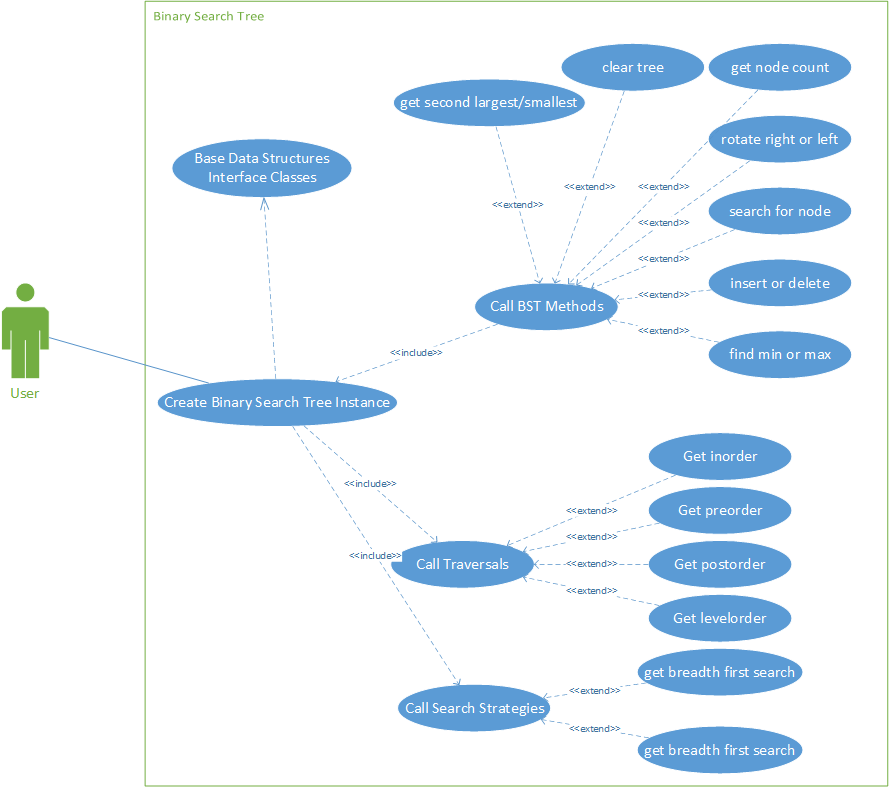
****

Figure 3: Use Case Diagram of Binary Search Tree

## General Tree

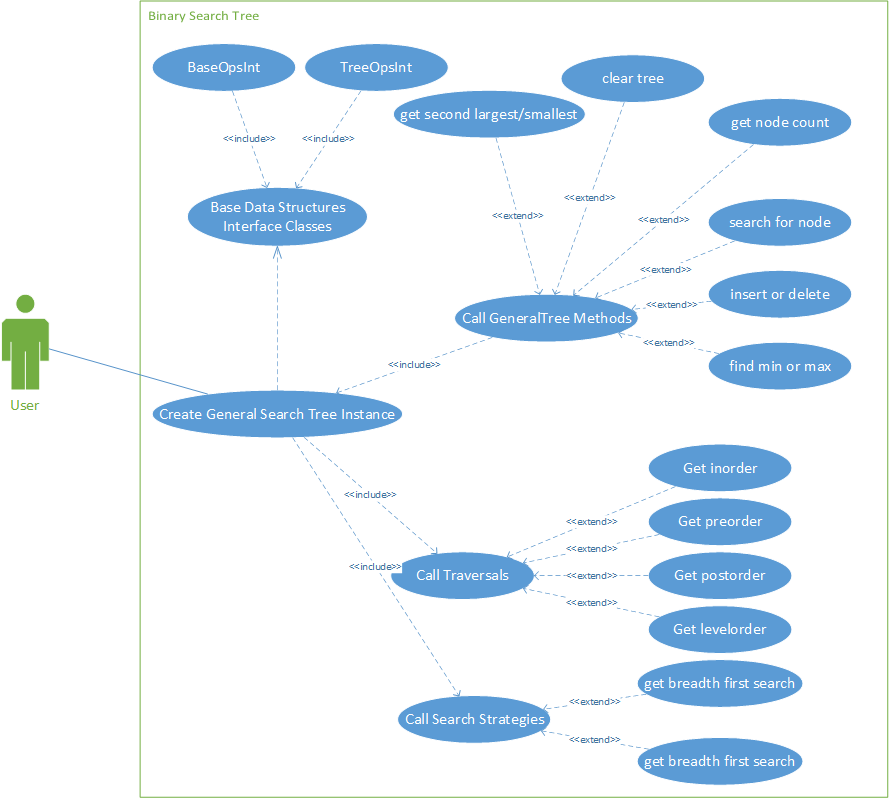
****

Figure 4: Use Case Diagram of General Tree Implementation

## Heap Tree

## 

Figure Heap Tree Use Case Diagram

# Class Diagrams:

## Edu.sru.thangiah

## 

Figure 6 Class diagram of edu.sru.thangiah package

## Edu.sru.thangiah.arrays

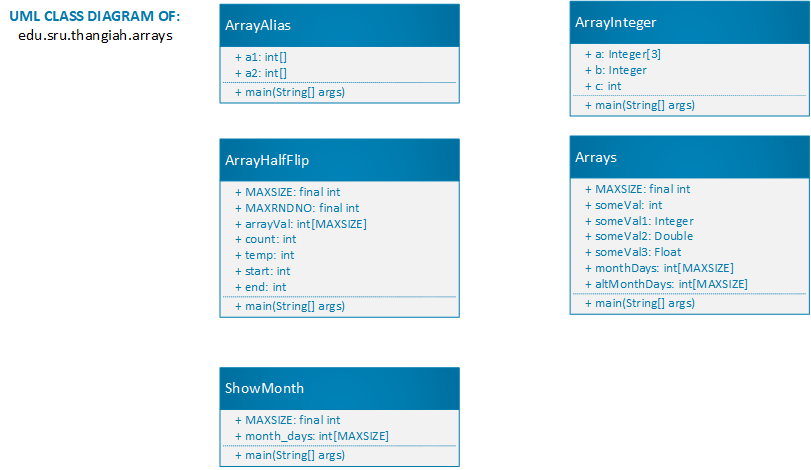


Figure 7 Class diagram of edu.sru.thangiah.arrays

## Edu.sru.thangiah.datastructures

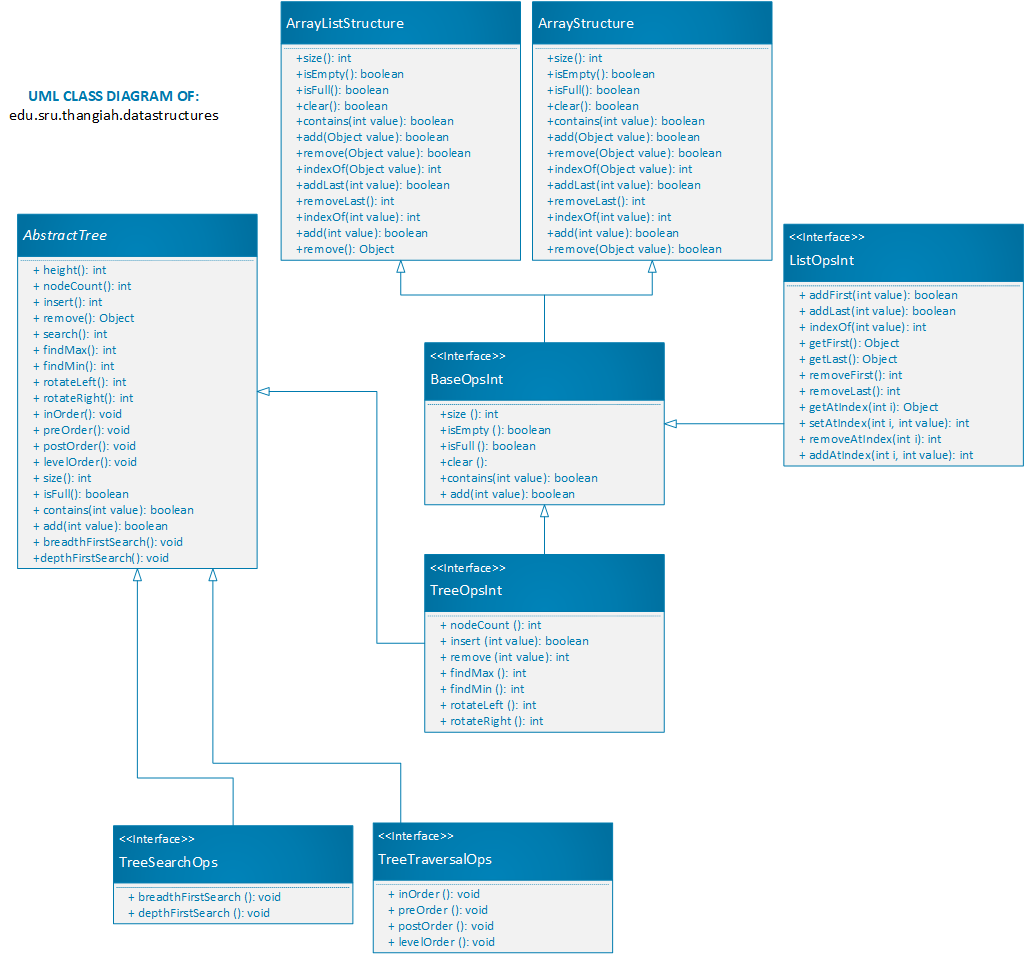


Figure 8 Class diagram of edu.sru.thangiah.datastructures

## Edu.sru.thangiah.datastructures.generic

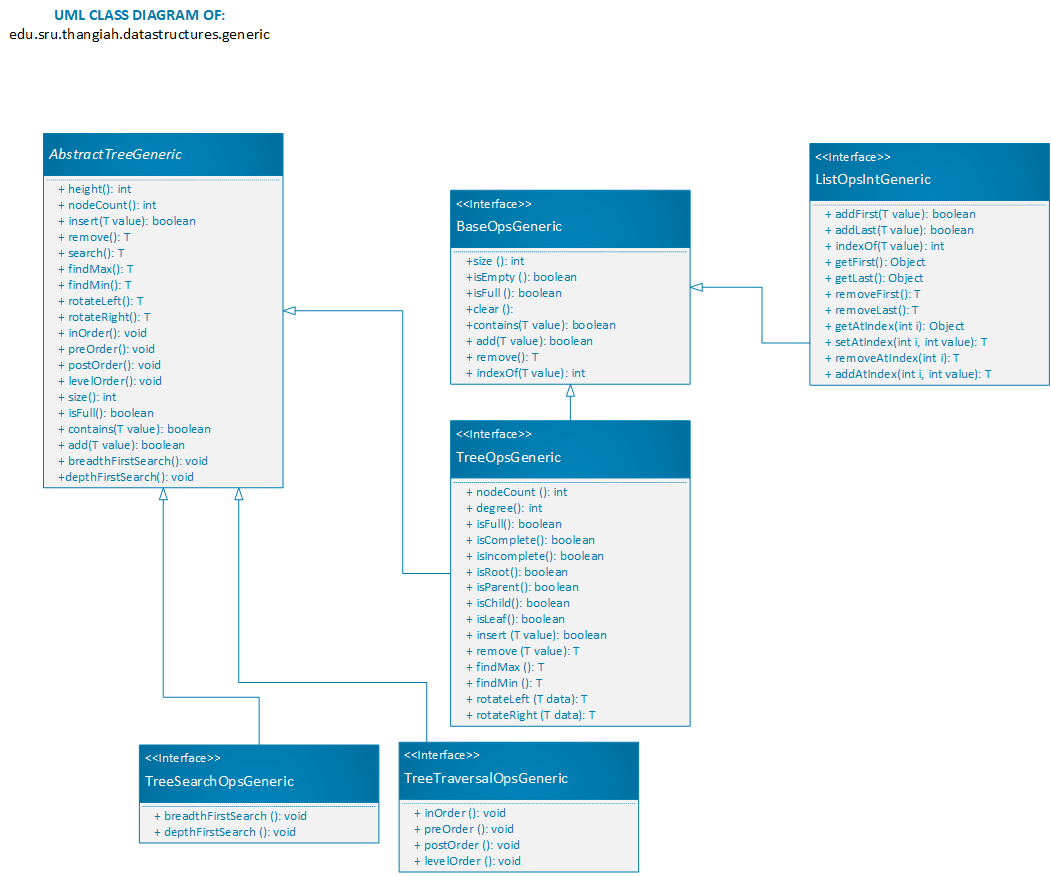


Figure 9 Class diagram of edu.sru.thangiah.datastructures.generic

## Edu.sru.thangiah.datastructures.generic.matrix

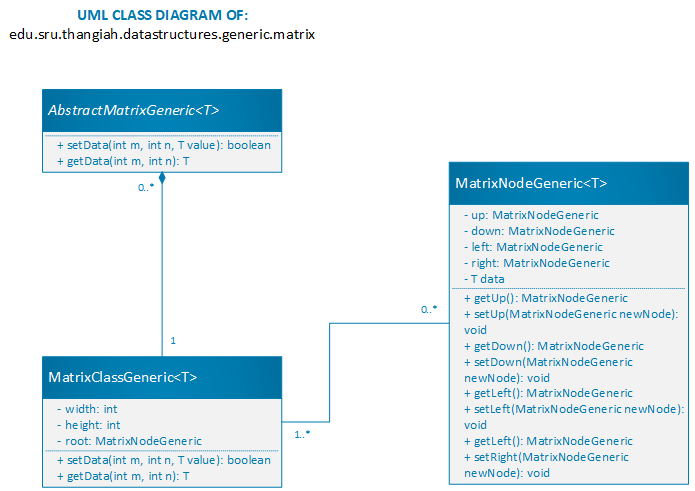


Figure 10 Class diagram of edu.sru.thangiah.datastructures.generic.matrix

## Edu.sru.thangiah.datastructures.generic.queue

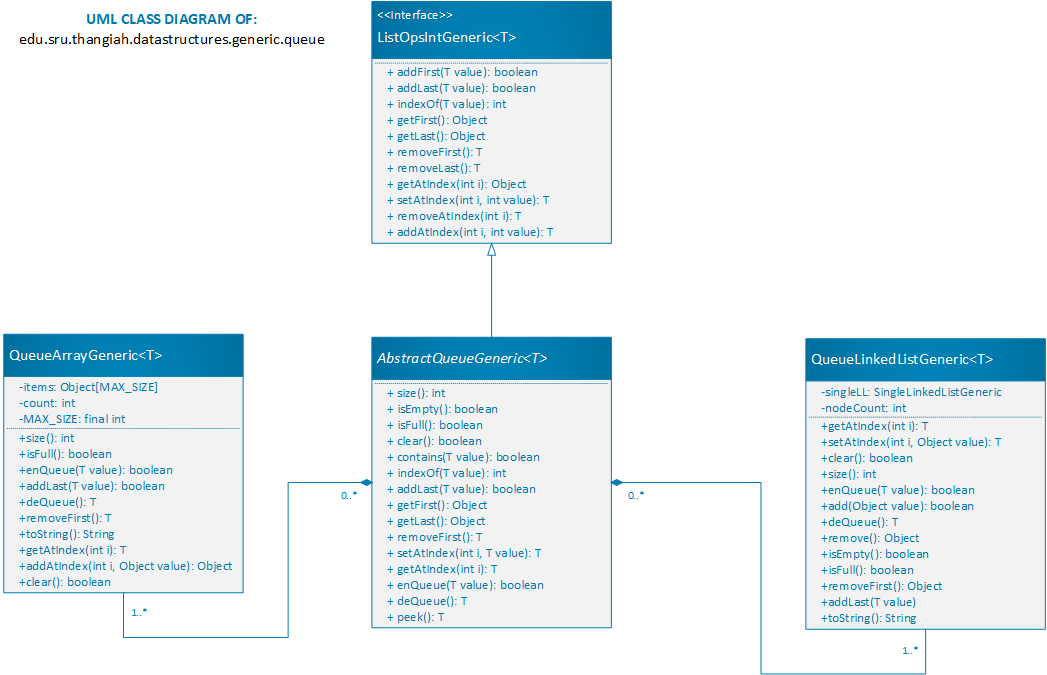


Figure 11 Class Diagram of edu.sru.thangiah.datastructures.generic.queue

## Edu.sru.thangiah.datastructures.generic.stack

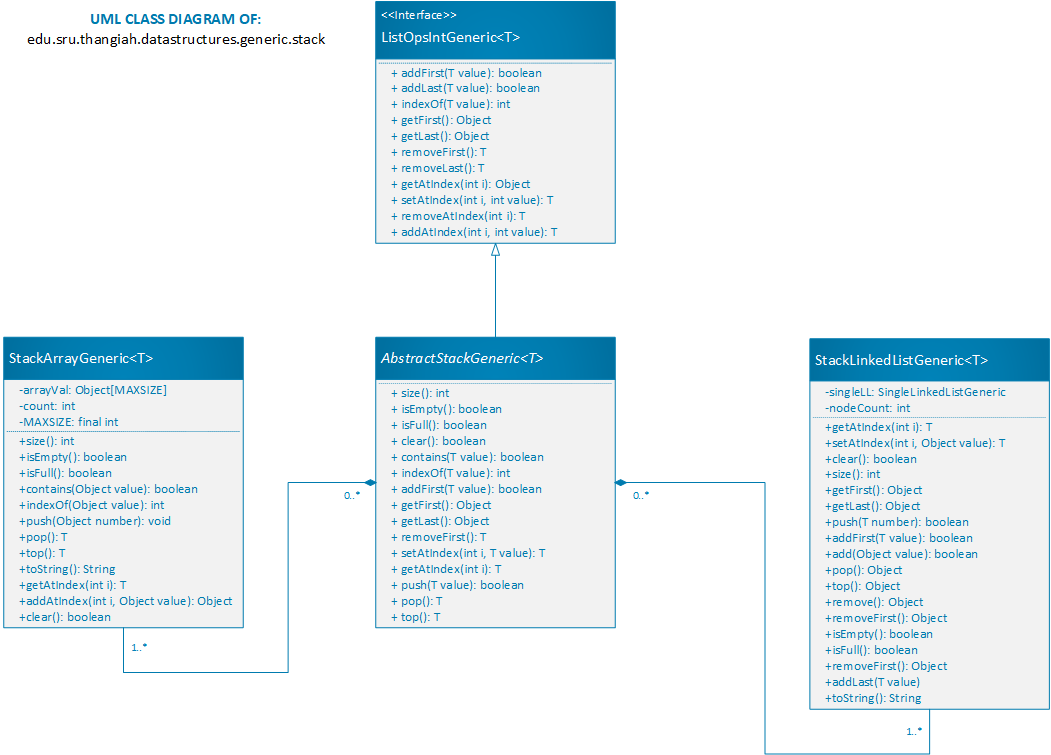


Figure 12 Class diagram of edu.sru.thangiah.datastructures.generic.stack

## Edu.sru.thangiah.datastructure.generic.linkedlist

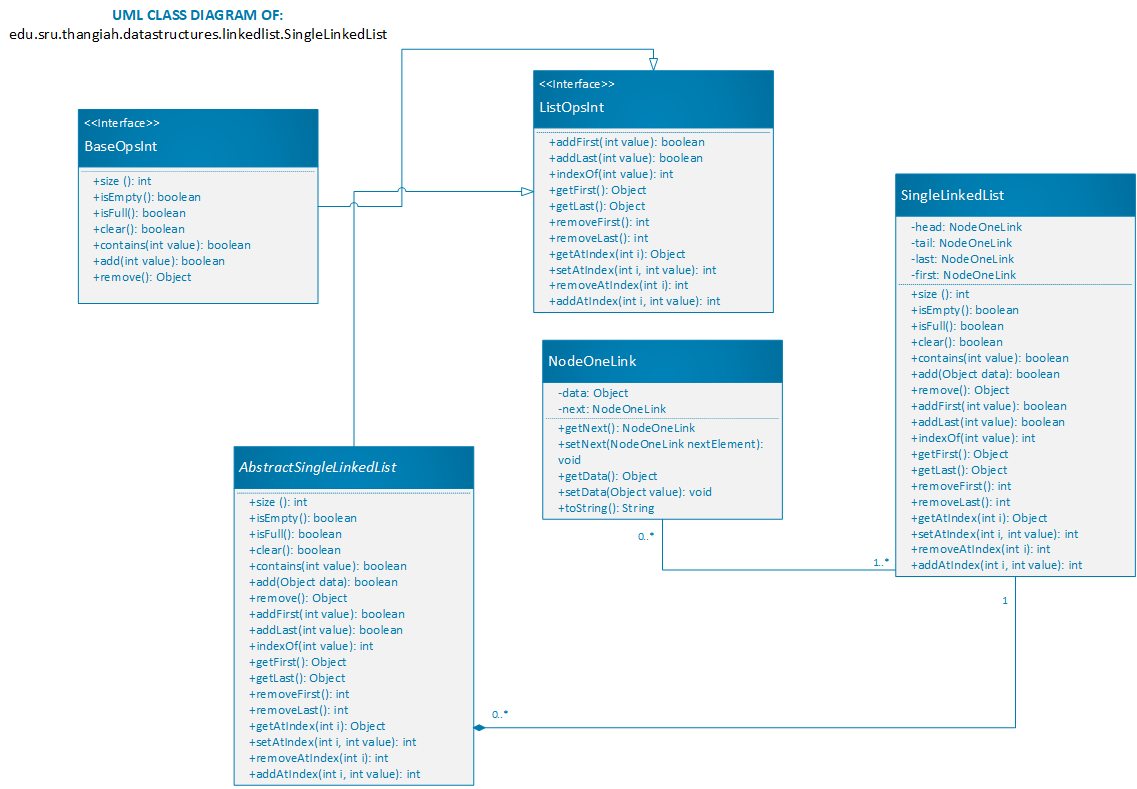
****

Figure 13: Class Diagram of Int Single Linked List Implementation

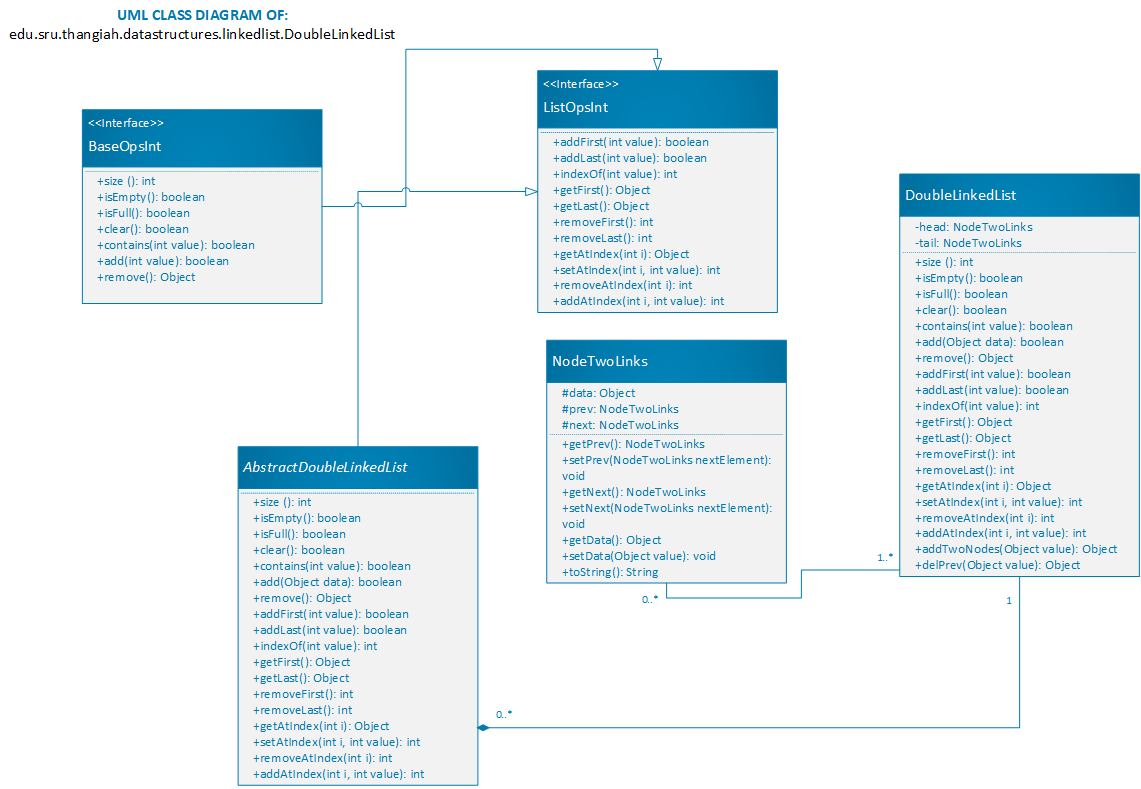
****

Figure 14: Class Diagram of Doubly Linked List Implementation

## Edu.sru.thangiah.datastructures.generic.linkedlist

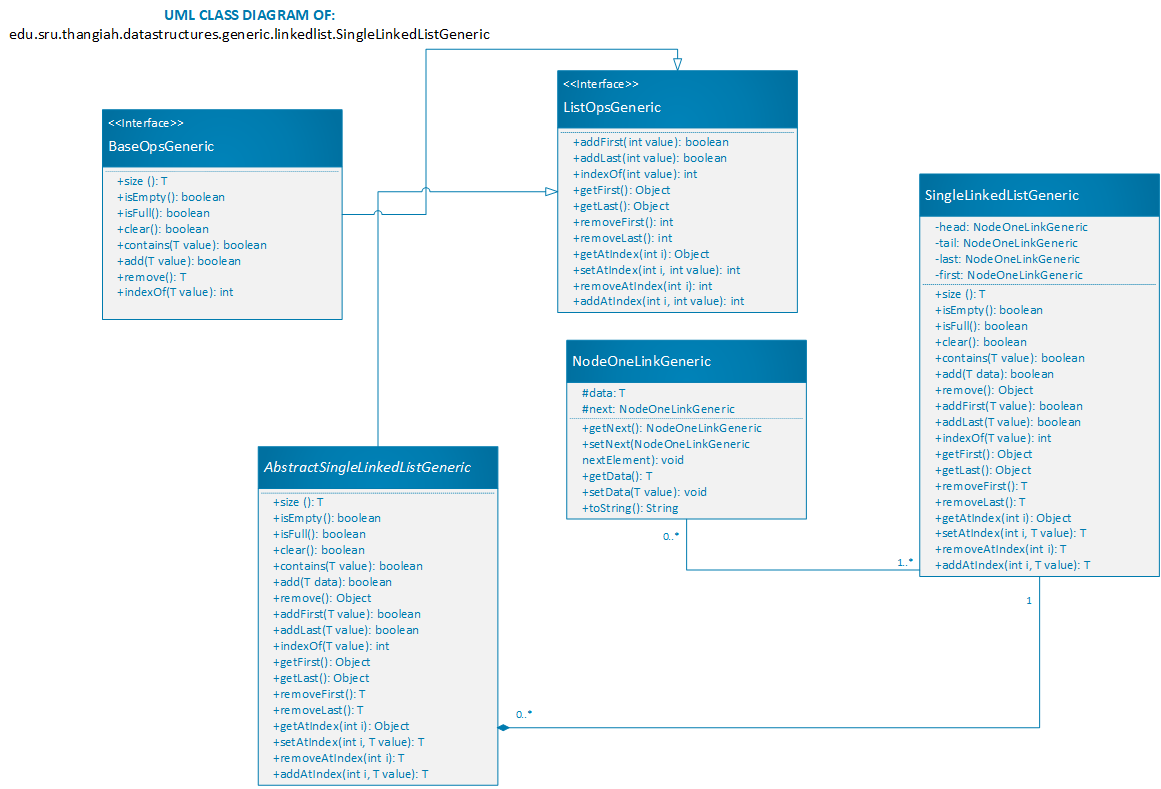
****

Figure 15: Class Diagram of Generic Single Linked List Implementation

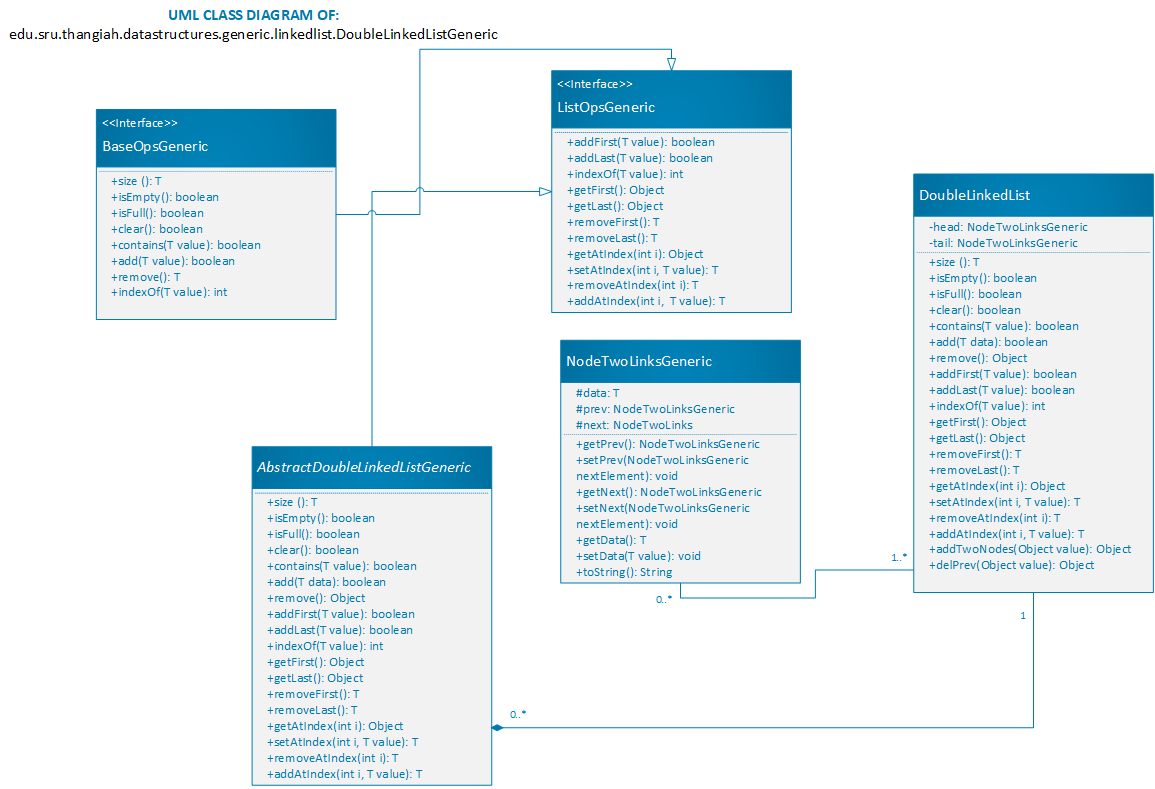
****

Figure 16: Class Diagram of Generic Doubly Linked List Implementation

## Edu.sru.thangiah.datastructures.tree.binarytree

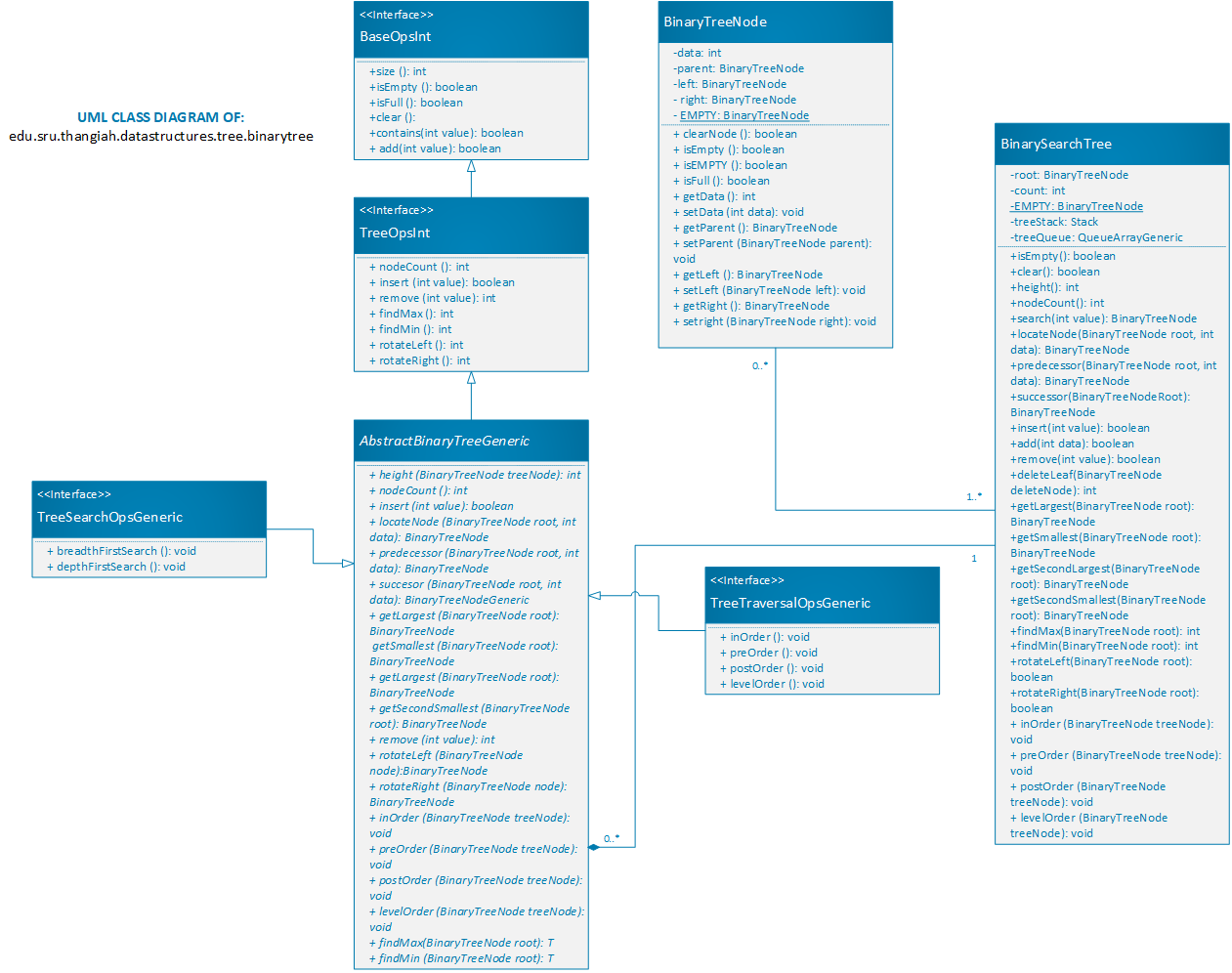
****

Figure 17: Class Diagram of Int Binary Tree Implementation

## edu.sru.thangiah.datastructures.generic.tree.binarytree

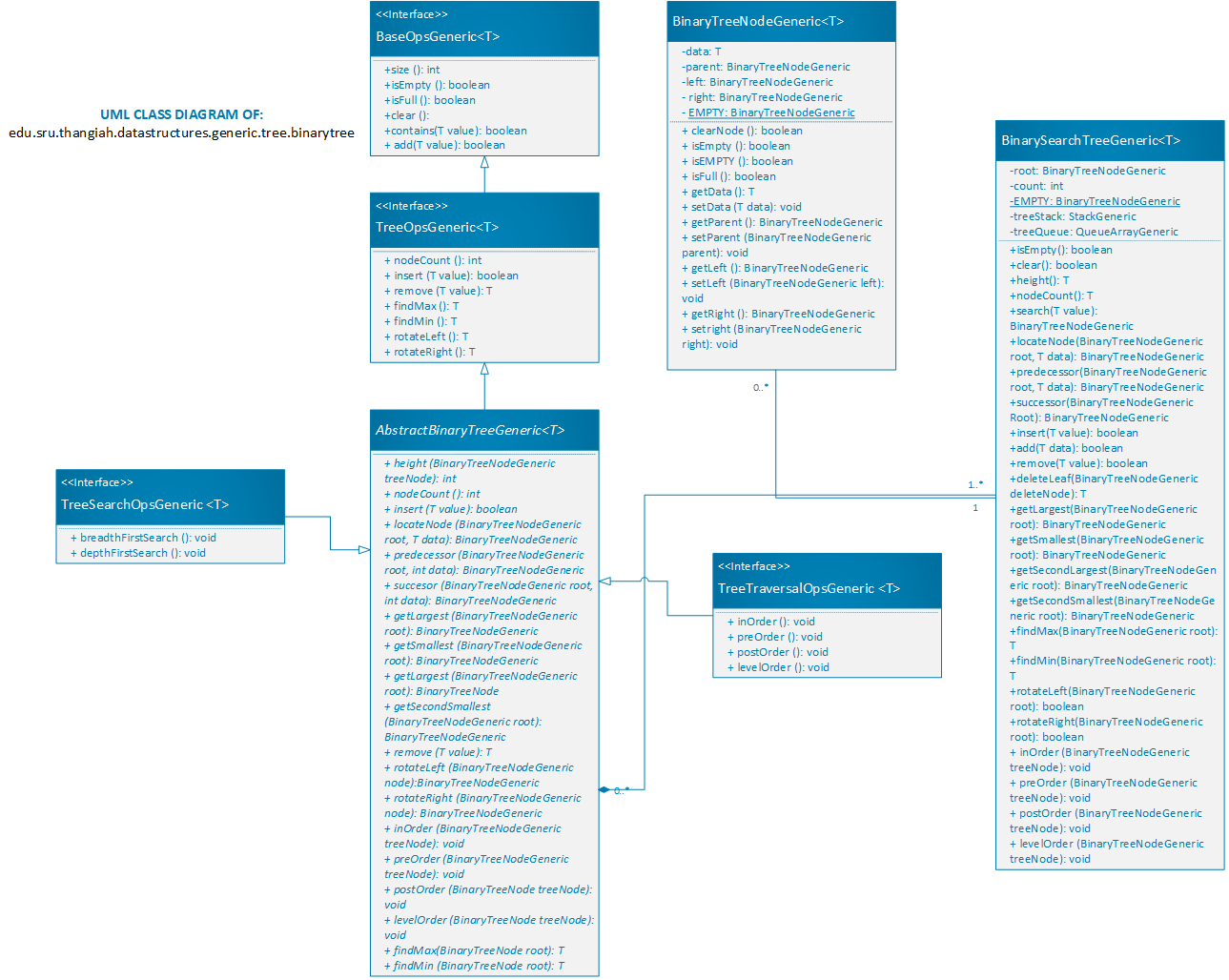
****

Figure 18: Class Diagram of Generic Binary Tree Implementation

## Edu.sru.thangiah.datastructures.tree.generaltree

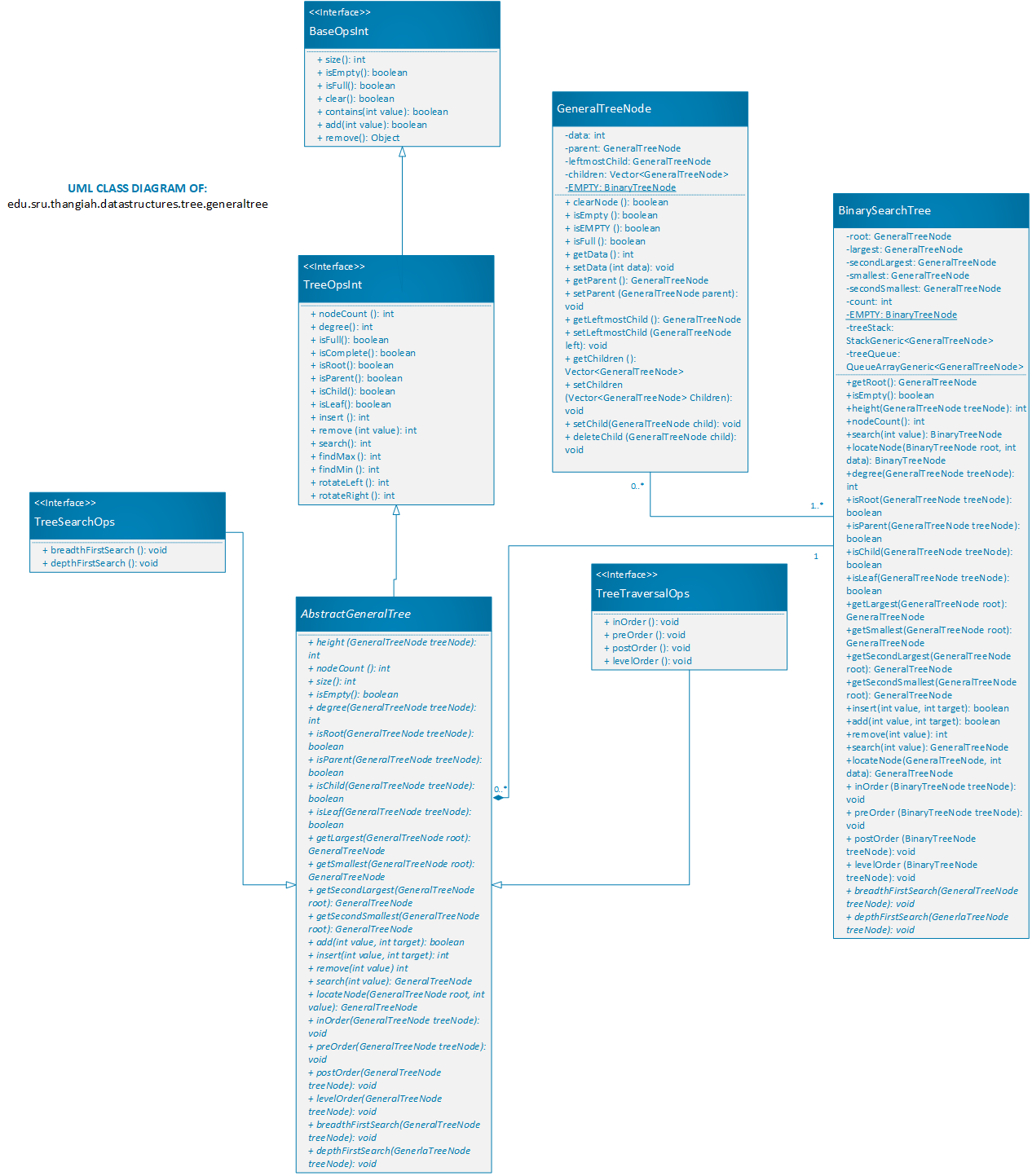
****

Figure 19: Class Diagram of Int General Tree Implementation

## edu.sru.thangiah.datastructures.tree.heaptree

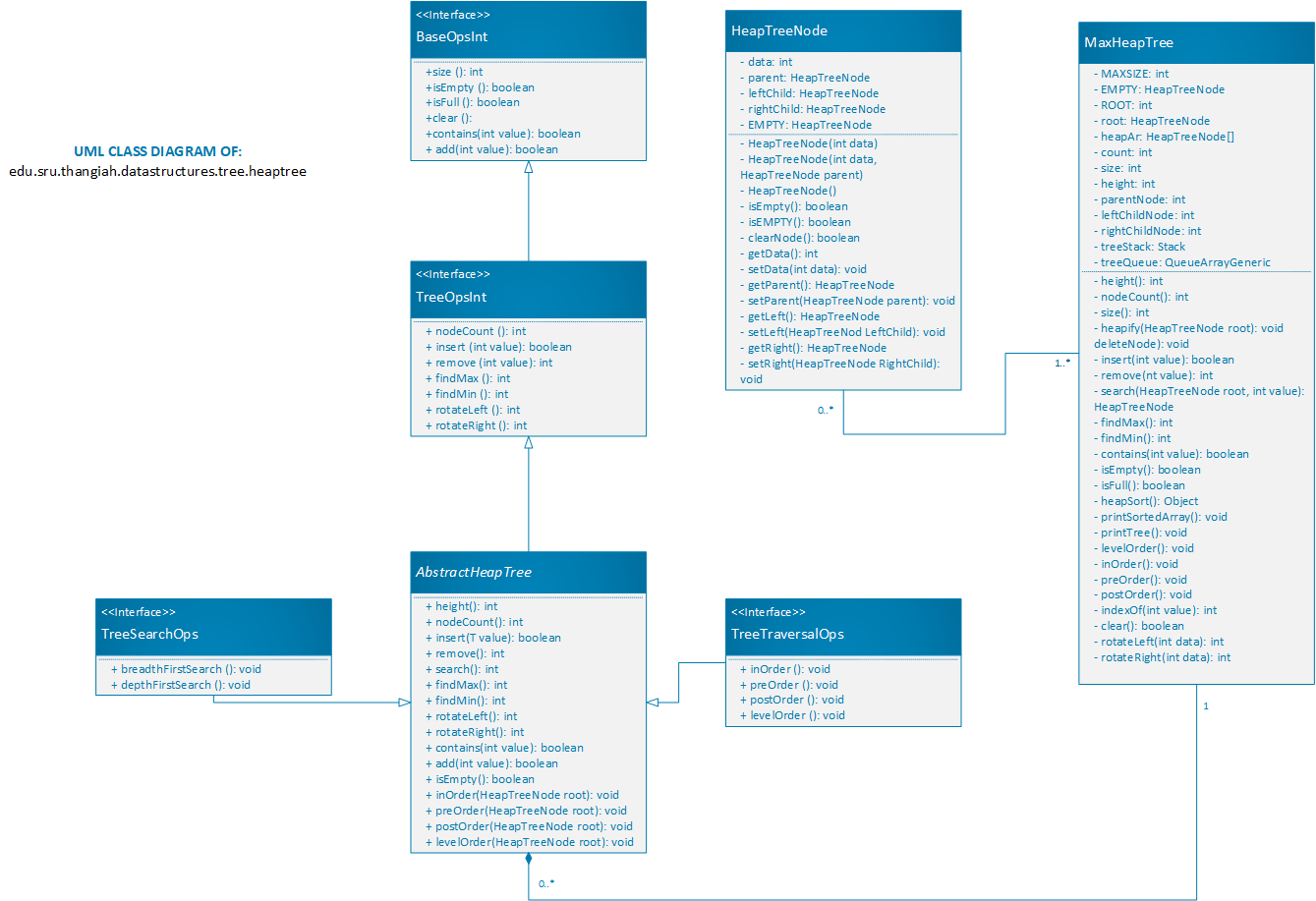
****

Figure 20: Class Diagram of Heap Tree Implementation

## edu.sru.thangiah.datastructures.generic.tree.heaptree

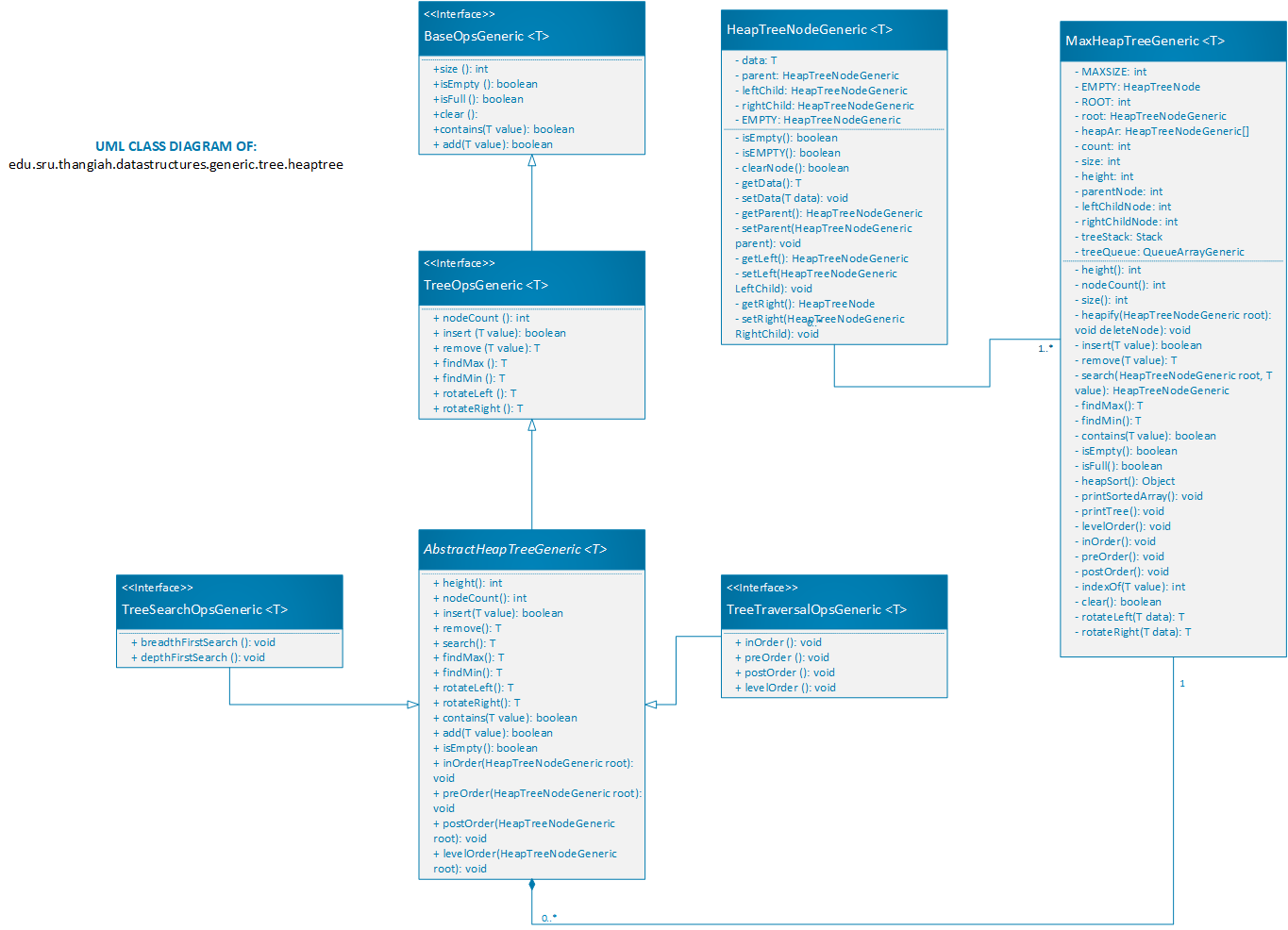
****

Figure 21: Class Diagram of Generic Heap Tree Implementation



Figure 22: Class Diagram of Generic AVL Tree

## edu.sru.thangiah.datastructures.generic.tree.avltree



Figure 23:Class Diagram of Int AVL Tree

## edu.sru.thangiah.datastructures.tree.avltree

## edu.sru.thangiah.datastructures.tree.redblacktree



Figure 24: Class Diagram of Int Red Black Tree Implementation

## edu.sru.thangiah.datastructures.generic.tree.redblacktree



Figure 25 Class Diagram of Generic Red Black Tree Implementation

## edu.sru.thangiah.datastructures.stack

Timeline

Description automatically generated with medium confidence

Figure 26 : Class diagram of int stack implementation

## sru.edu.thangiah.datastructures.queue

Graphical user interface, application

Description automatically generated

Figure 27 : Class diagram of int queue implementation

## edu.sru.thangiah.datastructurs.matrix

Graphical user interface, application

Description automatically generated with medium confidence

Figure 28 : Class diagram of int matrix implementation

## edu.sru.thangiah.datastructures.search.minimax

Graphical user interface, application

Description automatically generated

Figure 29 : Class diagram of Tic Tac Toe minimax implementation

## edu.sru.thangiah.datastructures.hashtable

## edu.sru.thangiah.datastructures.vector



Figure : Class diagram of the Vector Data Structure

## edu.sru.thangiah.datastructures.inheritance



Figure : Class diagram of the different examples of inheritance in the inheritance package

## edu.sru.thangiah.datastructures.interfaceex



Figure : Class diagram of the interface example

## edu.sru.thangiah.datastructures.polymorphism



Figure : Class diagram of the polymorphism example

## edu.sru.thangiah.datastructures.recursion



Figure : Class diagram of the recursion example

## edu.sru.thangiah.datastructures.sorting



Figure : Class diagram of the sorting package classes

Graphical user interface

Description automatically generated with medium confidence

Figure 36 : Class diagram of int hash table implementation

# Sequence Diagrams:

## In-Order Traversal for any Tree Data Structure

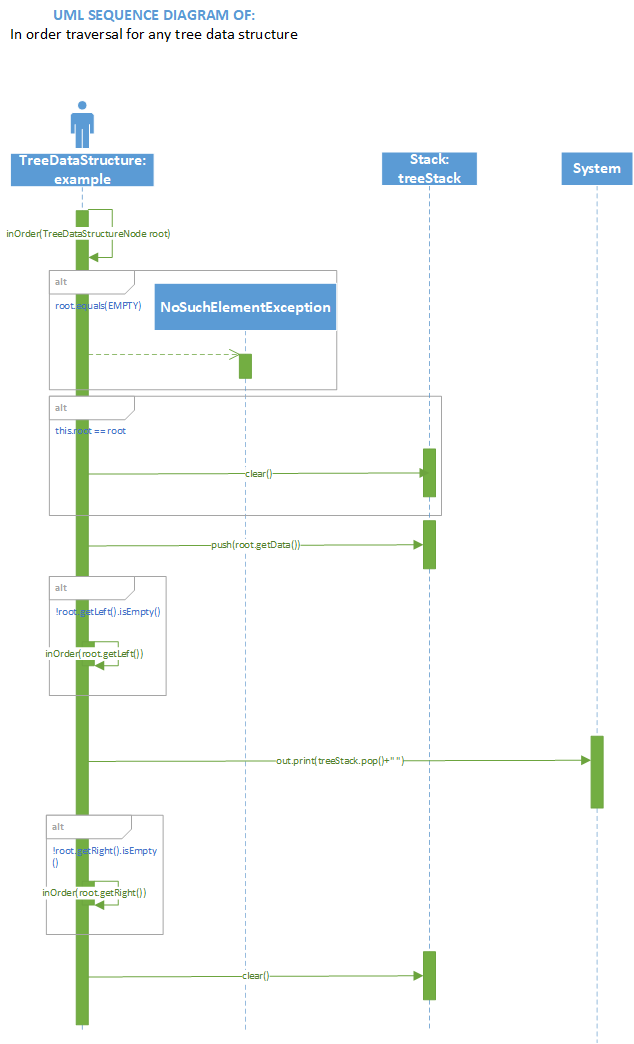


Figure 37 Sequence Diagram of In-Order Traversal for any Tree Data Structure

## Pre-Order Traversal for any Tree Data Structure

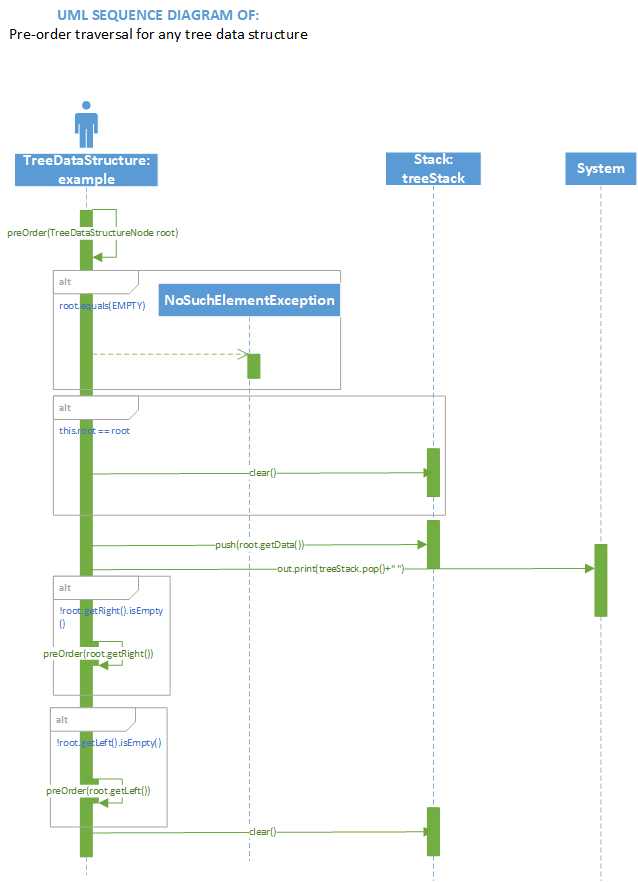


Figure 38 Sequence Diagram of Pre-Order Traversal for any Tree Data Structure

## Post-Order Traversal for any Tree Data Structure

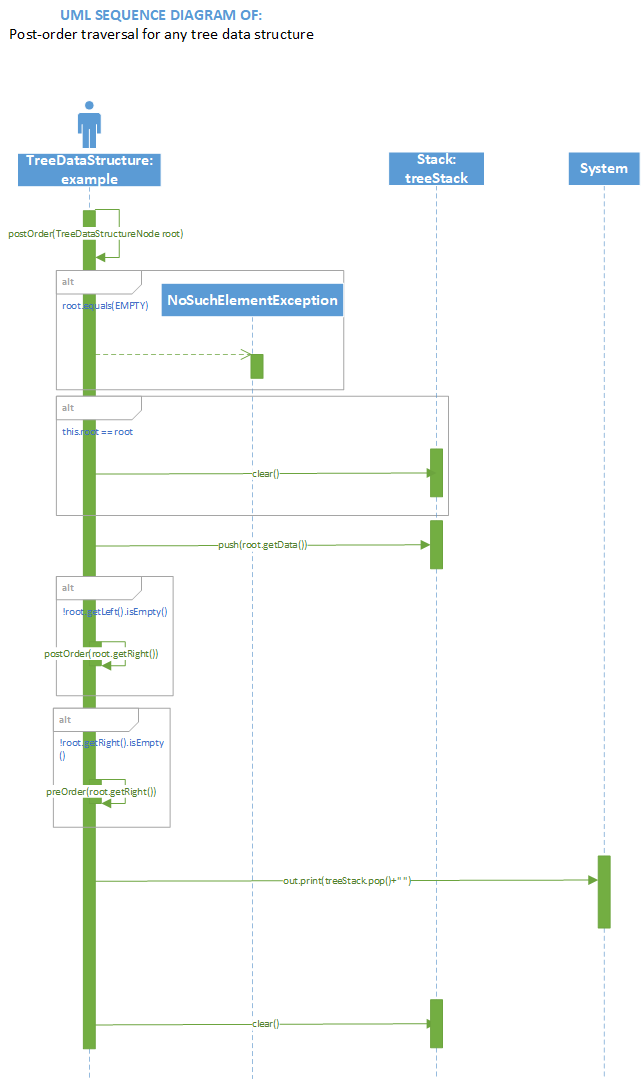


Figure 39 Sequence Diagram of Post Order Traversal for any Tree Data Structure

## Level-Order Traversal for any Tree Data Structure

## 

Figure 40 Sequence Diagram of Level Order Traversal for any Tree Data Structure

## Hash Table Add Method (Array)

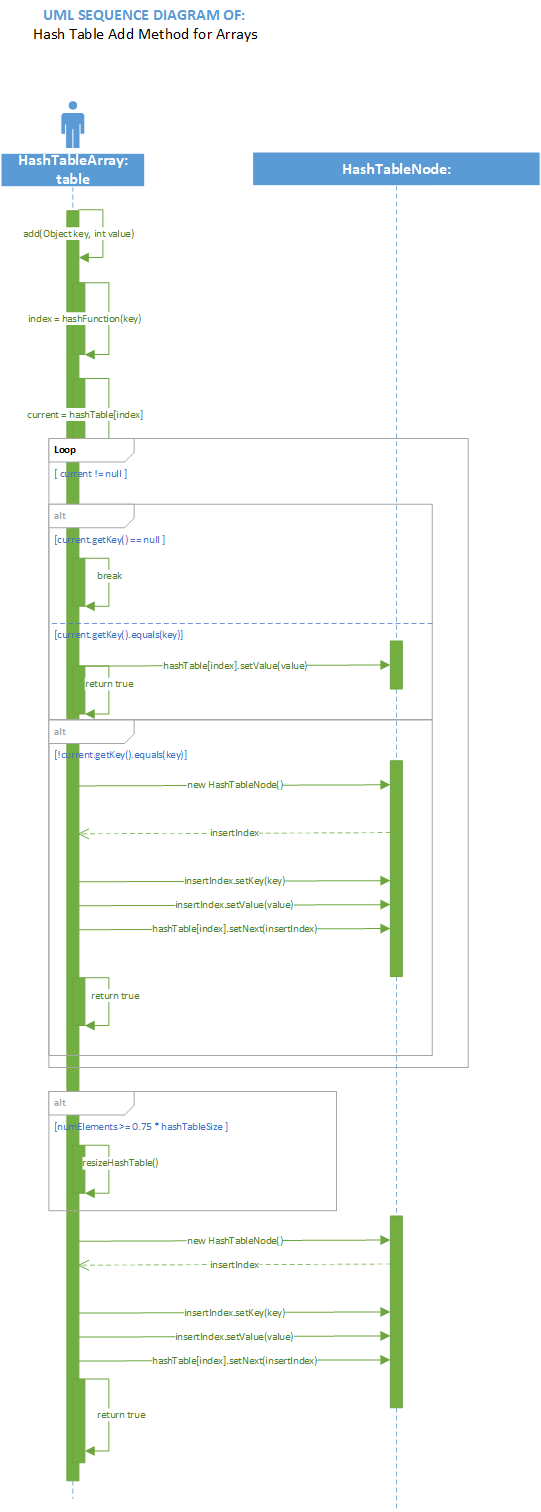


Figure : Sequence Diagram of Add Method for Array Hash Table Implementation

## Hash Table Add Method (ArrayList)

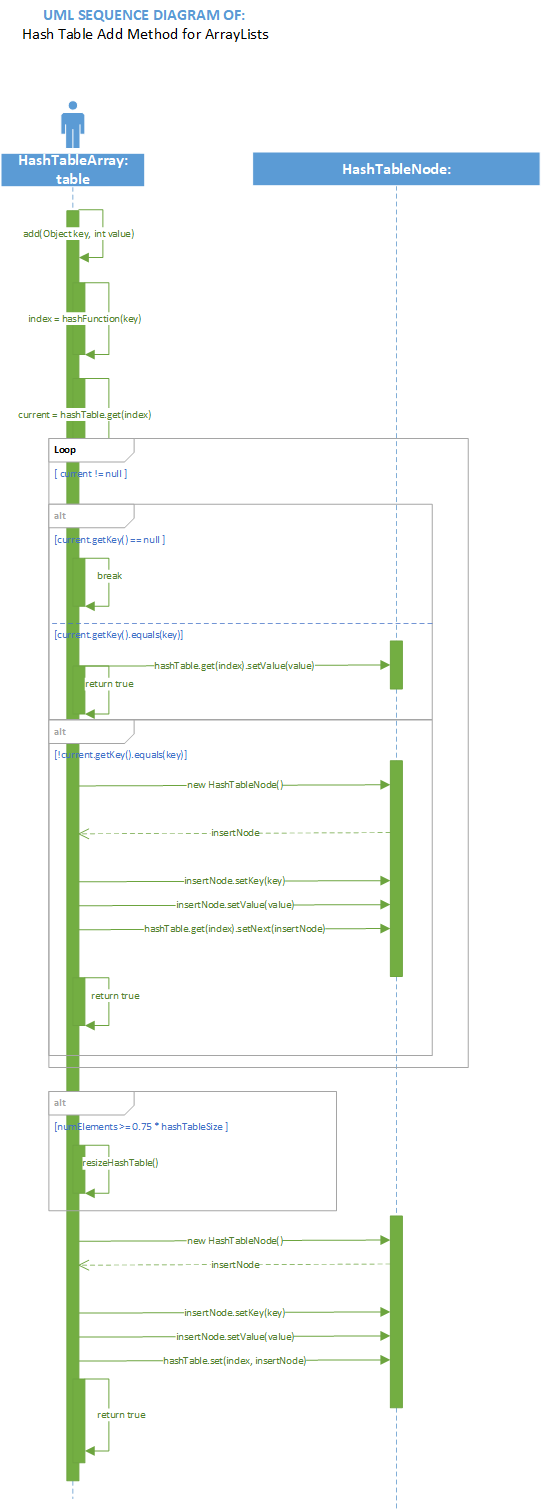


Figure Sequence Diagram of Hash Table Add Method for ArrayList Hash Table Implementation

## Hash Table add method (Vector)

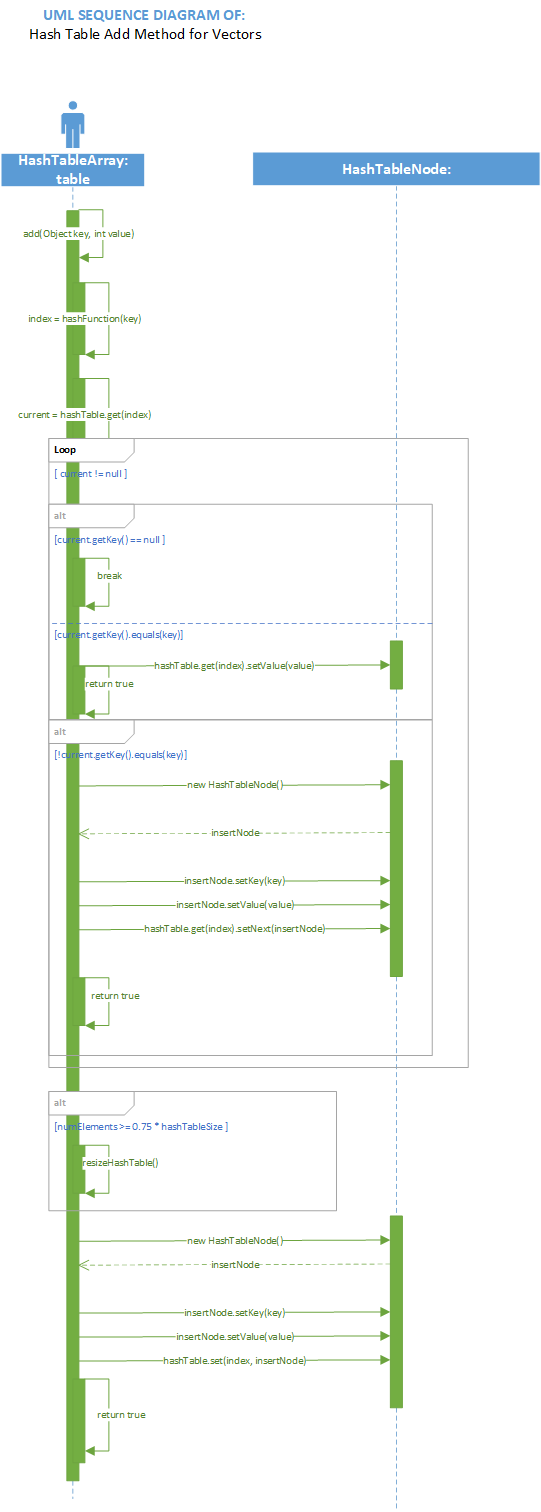


Figure Sequence Diagram of Add Method for Vector Hash Table Implementation

## Minimax Tic Tac Toe

Graphical user interface

Description automatically generated with medium confidence

Figure 44 MiniMax Tic Tac Toe Sequence Diagram



Figure Sequence Diagram of AVL Tree Addition

# UML Statechart Diagrams

## In-order Traversal for any Tree Data Structure

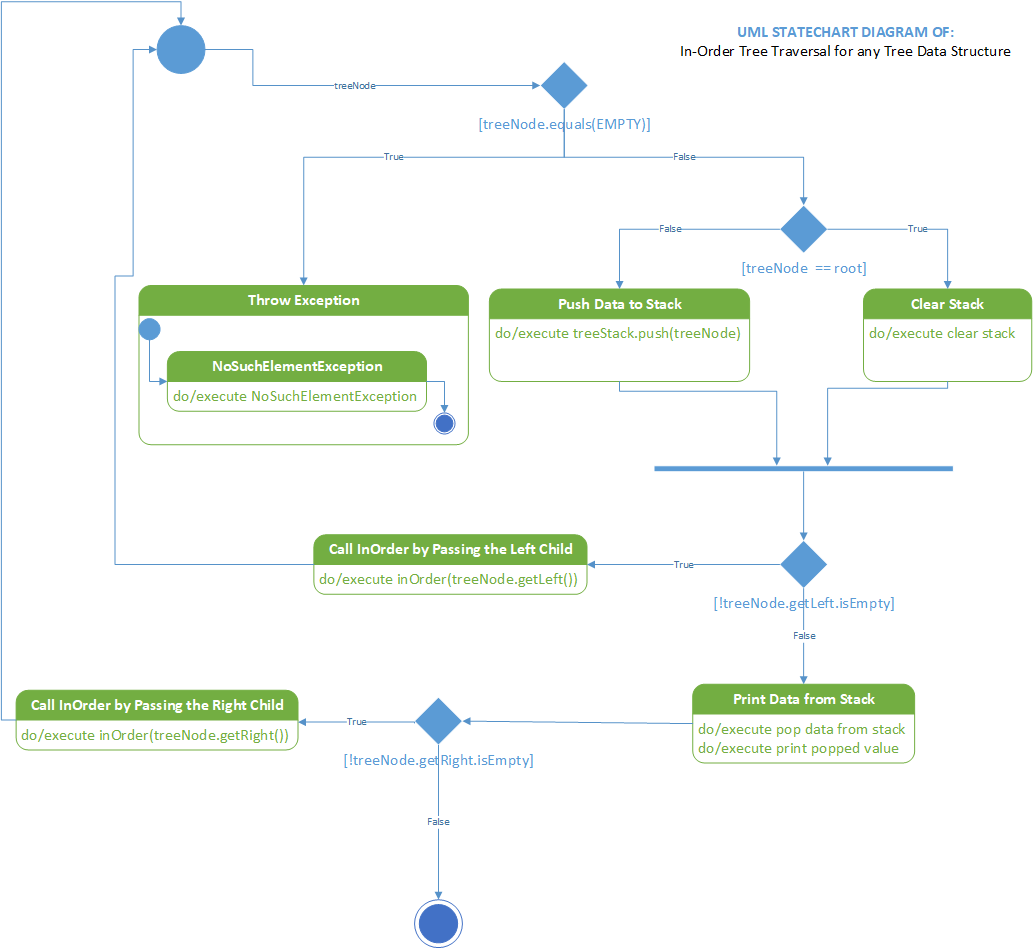


Figure Statechart diagram of In-Order traversal for any Tree Data Structure Implementation

## Pre-Order Traversal for any Tree Data Structure

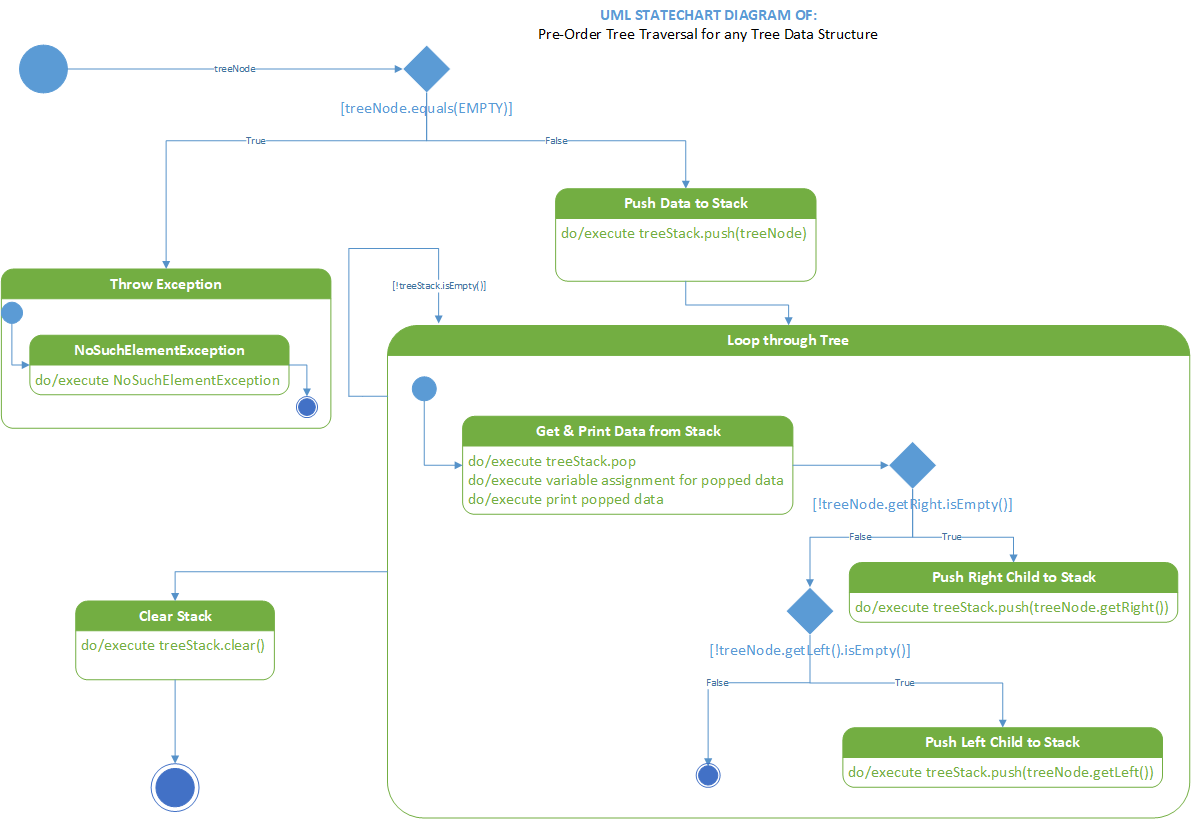


Figure State Chart Diagram of Pre-Order Traversal for any Tree Data Structure Implementation

## Post-Order Traversal for any Tree Data Structure

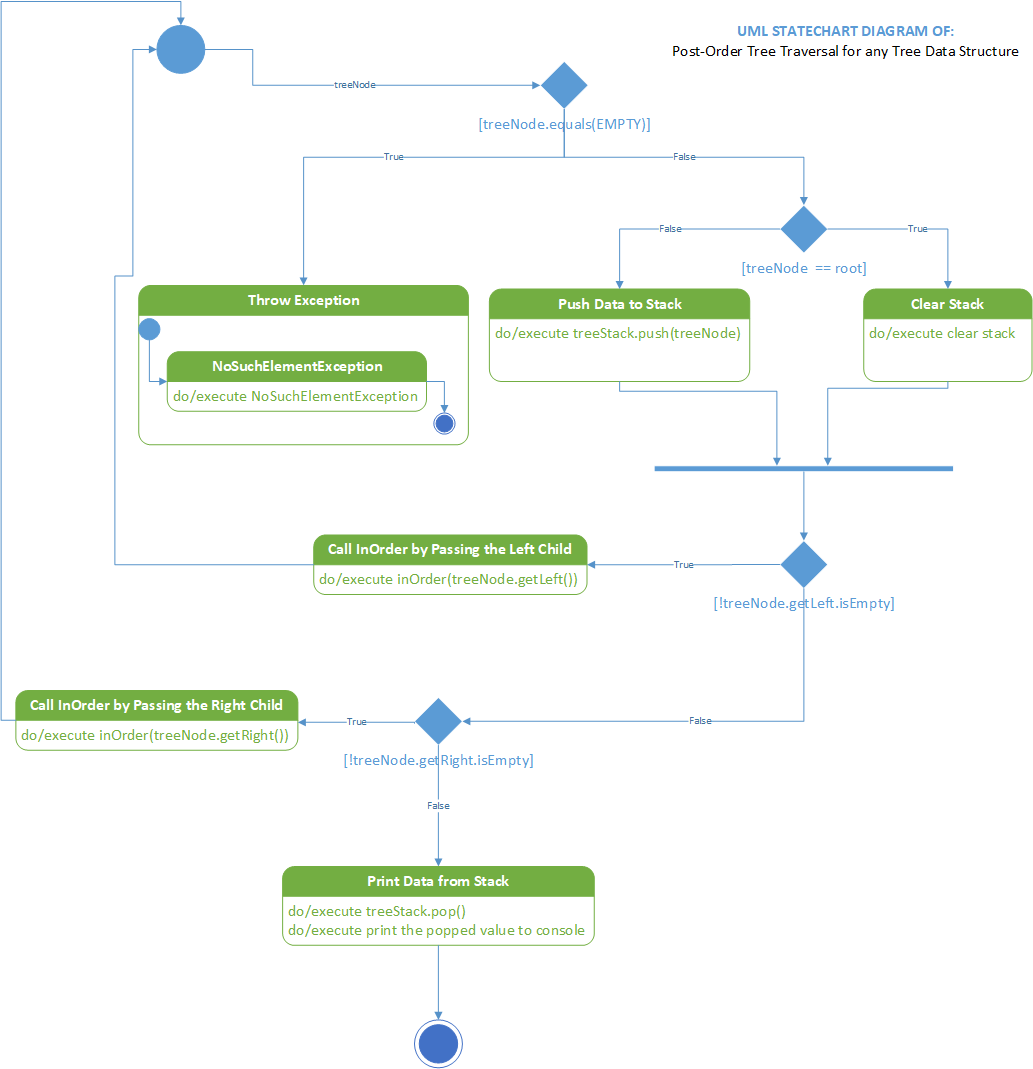


Figure Statechart diagram of Post-Order Traversal for any Tree Data Structure Implementation

## Level Order Traversal for any Tree Data Structure

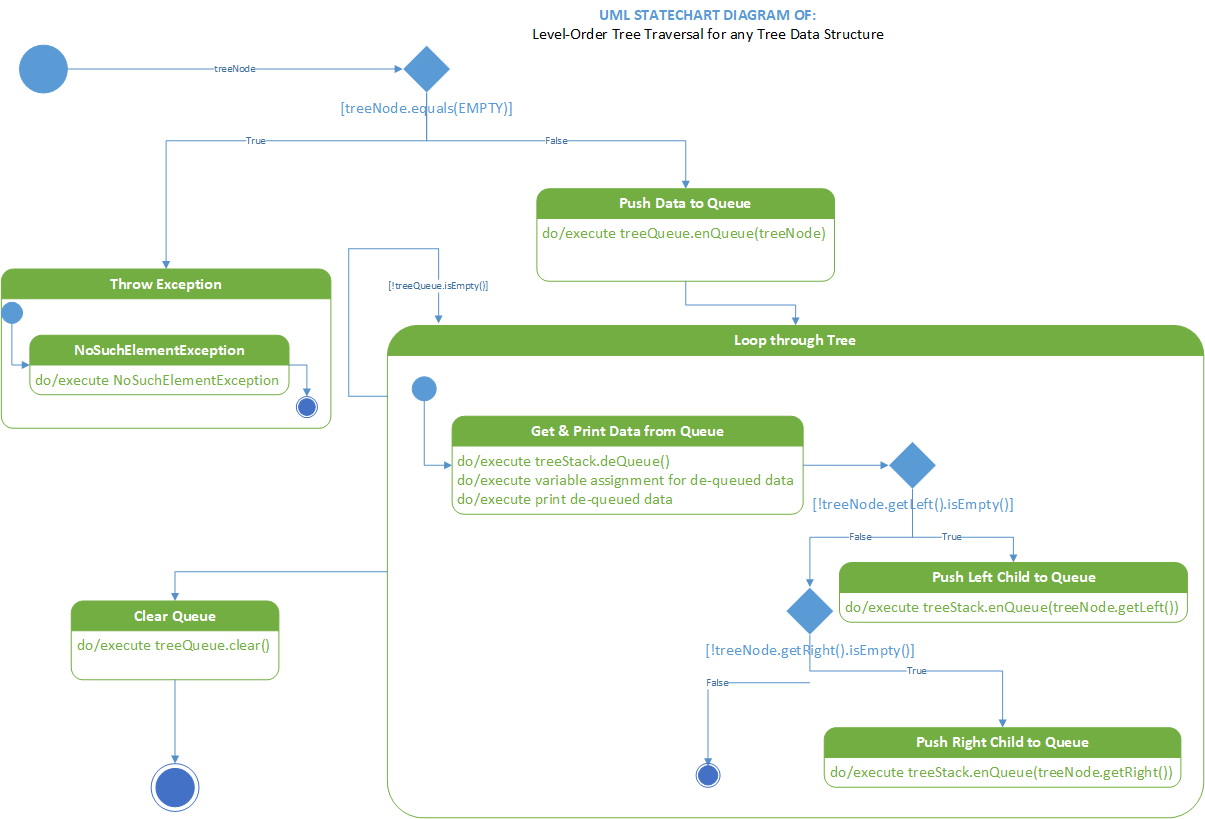


Figure Statechart diagram of Level Order Traversal for any Tree Data Structure Implementation

## Hash Table Add Method for any Container (Array, ArrayList, Vector)

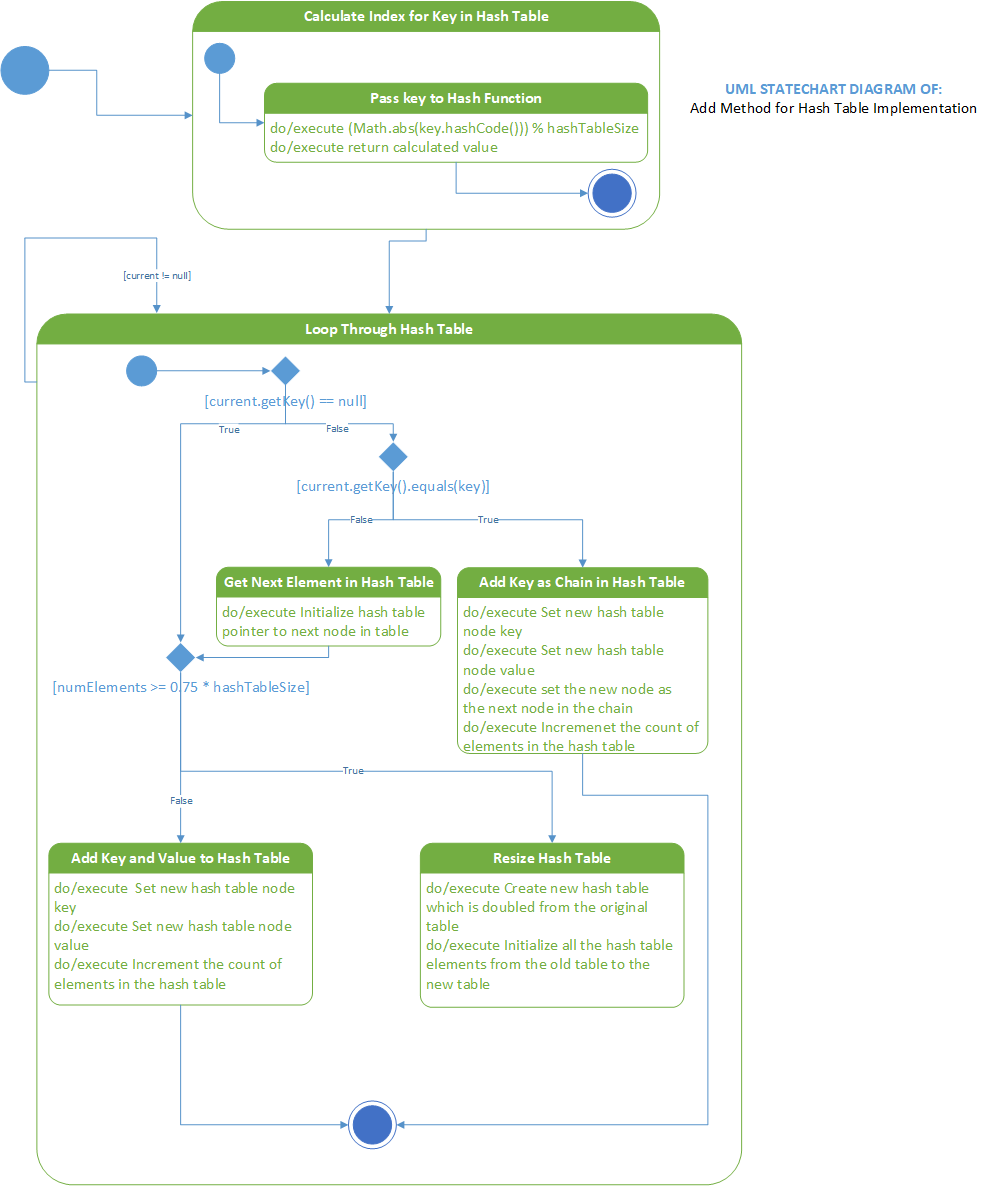


Figure Statechart Diagram of Add Method for any Hash Table Implementation (Array, ArrayList, Vector)

## Minimax Tic Tac Toe

Diagram

Description automatically generated

Figure 51 State Chart Diagram of Minimax Tic Tac Toe Game



Figure Statechart Diagram of AVL Tree Addition

# UML Activity Diagrams

## In-Order Traversal for any Tree Data Structure

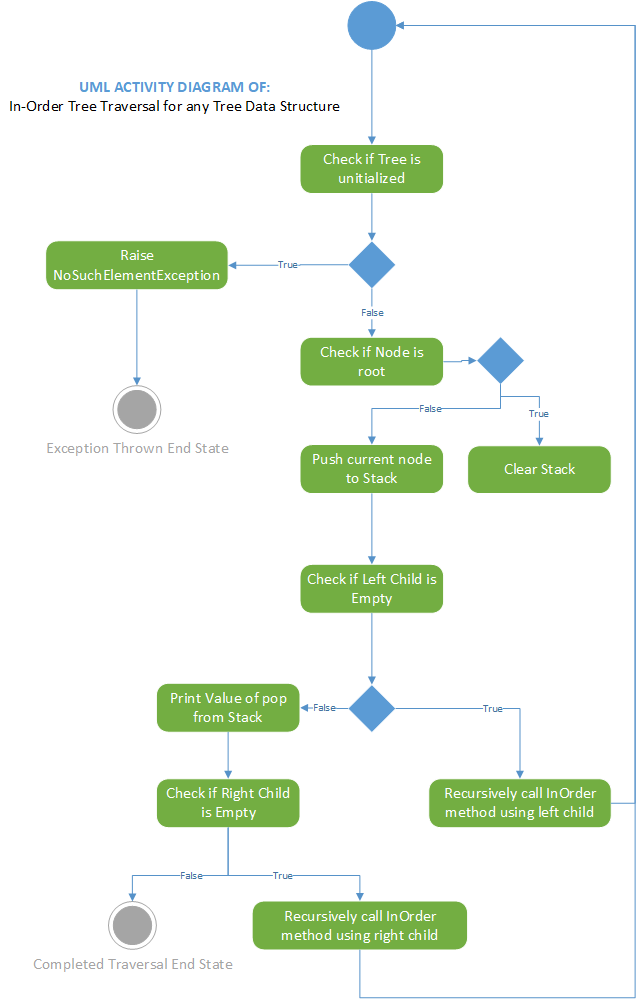


Figure Activity Diagram of In-Order Tree Traversal for any Tree Data Structure

## Pre-Order Traversal for any Tree Data Structure

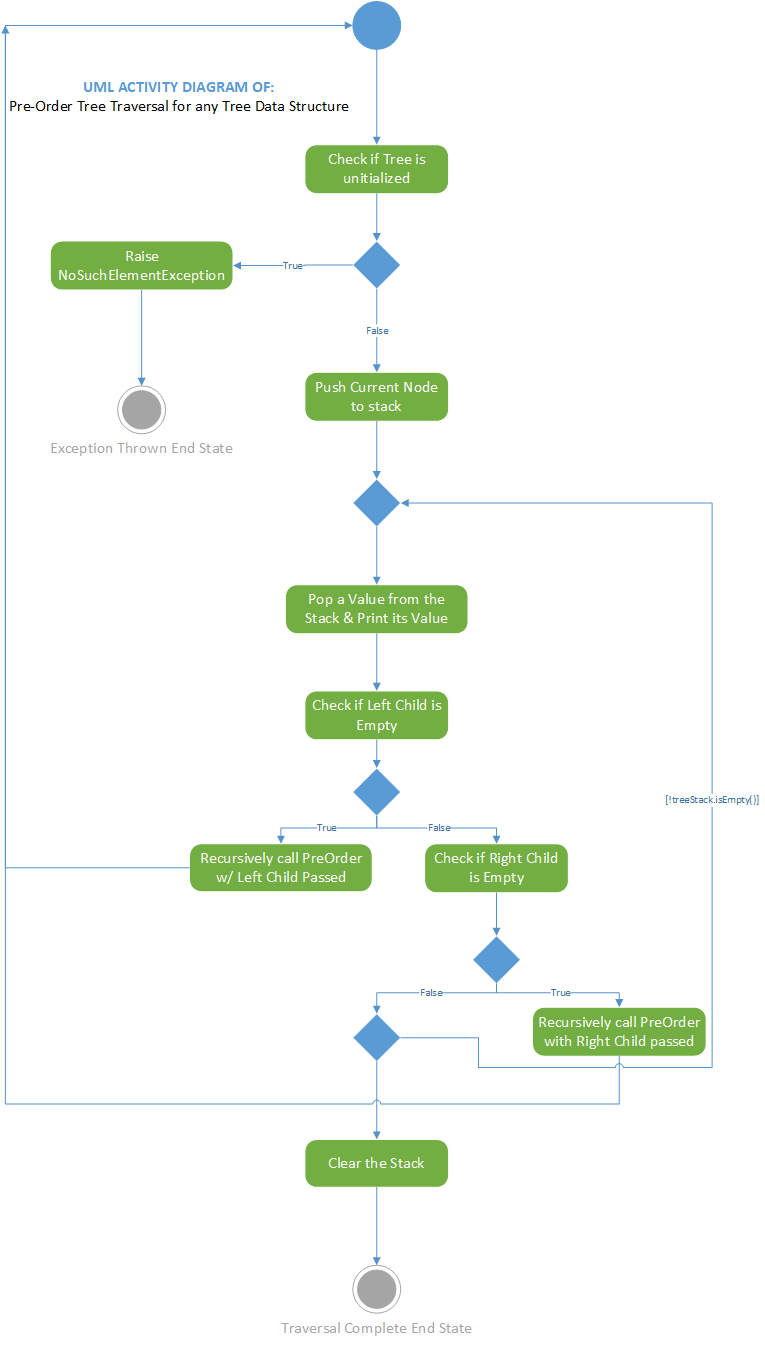


Figure Activity Diagram of Pre-Order Traversal for any Tree Data Structure Implementation

## Post-Order Traversal for any Tree Data Structure

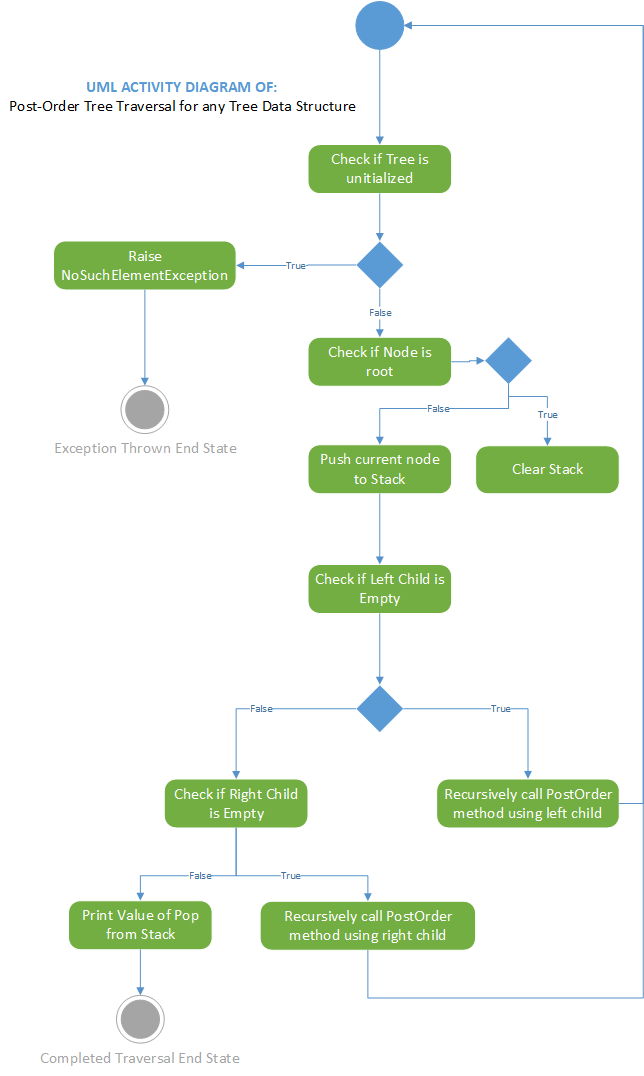


Figure Activity Diagram of Post-Order Traversal for any Tree Data Structure Implementation

## Level-Order Traversal for any Tree Data Structure

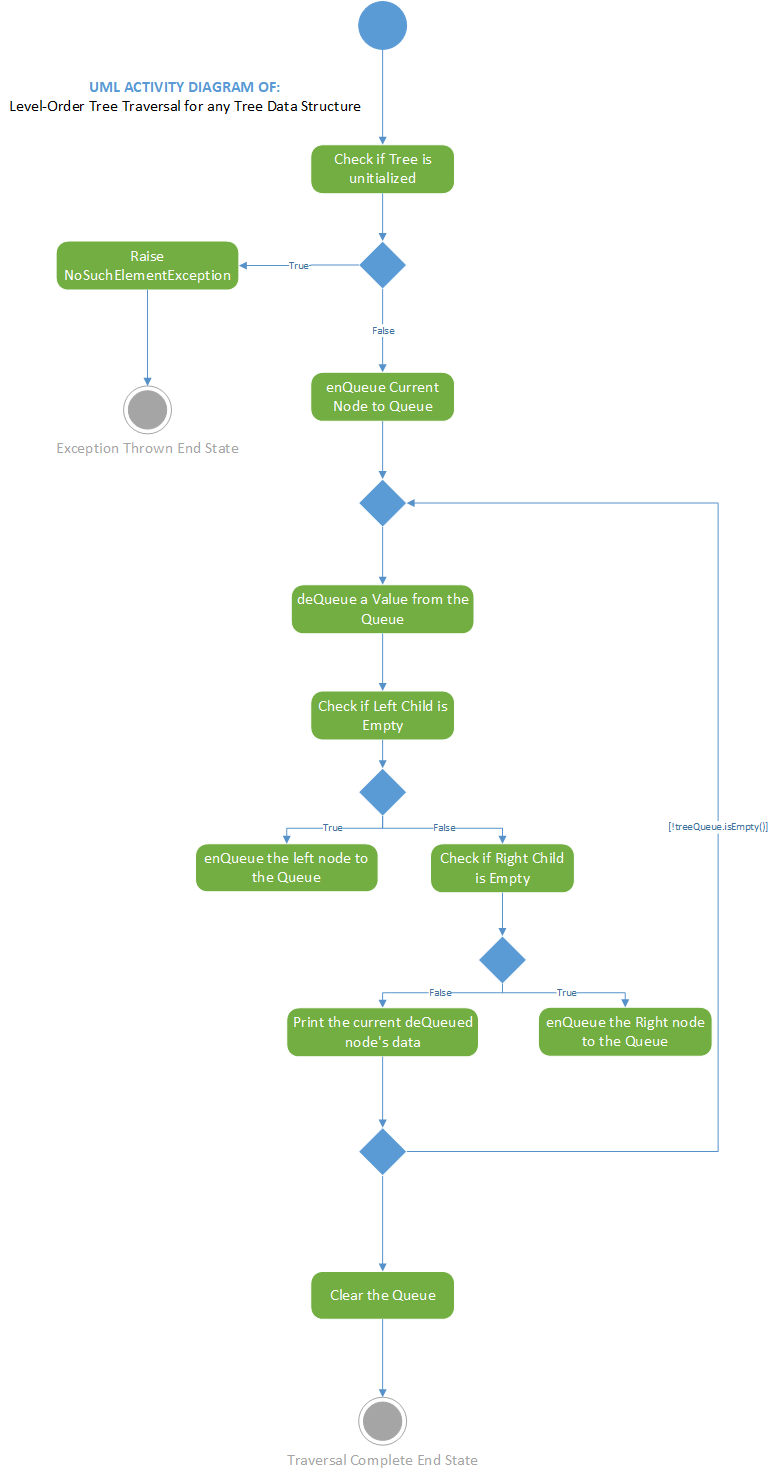


Figure Activity Diagram of Level-Order Traversal for any Tree Data Structure

## Hash Table Add Method for any Container (Array, ArrayList, Vector)

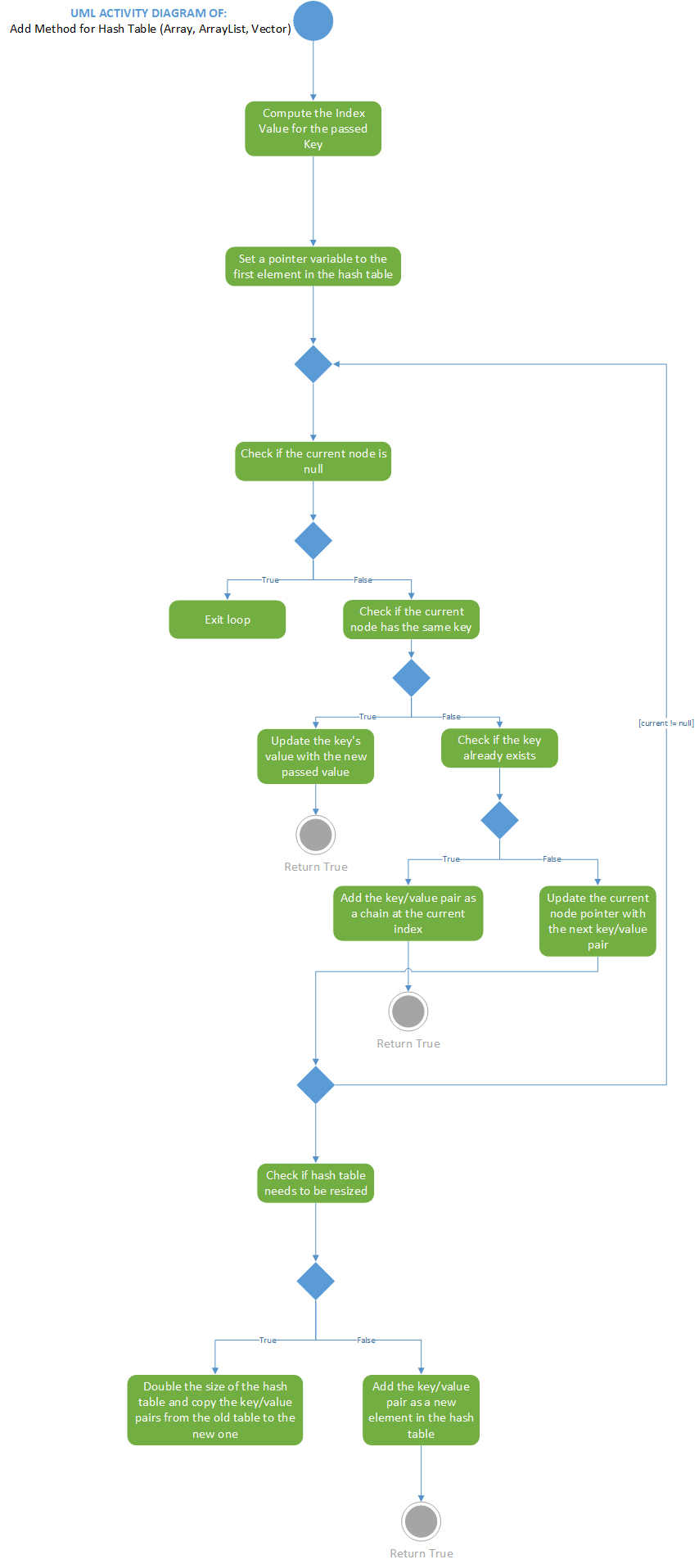


Figure Activity Diagram of Hash Table Add Method for any Container (Array, ArrayList, Vector)

## Minimax Tic Tac Toe Game

Diagram

Description automatically generated

Figure 58 Activity Diagram of Minimax Tic Tac Toe Game



Figure Activity Diagram of Addition in AVL Trees

Diagram

Description automatically generated

Figure 60 Activity Diagram of Minimax Tic Tac Toe Game