## Topics

1. Create Position Interface
2. Create Tree Interface
3. Create AbstractTree Class.
4. Create BinaryTree Interface
5. Create AbstractBinaryTree Class
6. Create Linked Binary Tree Using Linked List structure (Nodes)
7. Implement Basic Methods of LinkedBinaryTree

* addRoot( E e)
* addLeft(Position<E> p ,E e)
* addRight(Position<E> p ,E e)
* set(Position<E> p ,E e)
* remove(Position<E> p)

## Homework

1. Implement the Array Based Binary Tree Data structure as it is described in chapter 8.

import java.util.Arrays;

class ArrayBinaryTree<T> {

private static final int DEFAULT\_CAPACITY = 10;

private T[] tree;

private int size;

@SuppressWarnings("unchecked")

public ArrayBinaryTree() {

tree = (T[]) new Object[DEFAULT\_CAPACITY];

size = 0;

}

public void add(T element) {

ensureCapacity();

tree[size++] = element;

}

public T get(int index) {

if (index < 0 || index >= size) {

throw new IndexOutOfBoundsException("Index out of range: " + index);

}

return tree[index];

}

public T getLeftChild(int index) {

int leftIndex = 2 \* index + 1;

return (leftIndex < size) ? tree[leftIndex] : null;

}

public T getRightChild(int index) {

int rightIndex = 2 \* index + 2;

return (rightIndex < size) ? tree[rightIndex] : null;

}

public T getParent(int index) {

if (index == 0) return null;

int parentIndex = (index - 1) / 2;

return tree[parentIndex];

}

private void ensureCapacity() {

if (size == tree.length) {

tree = Arrays.copyOf(tree, tree.length \* 2);

}

}

public void inorderTraversal(int index) {

if (index >= size || tree[index] == null) return;

inorderTraversal(2 \* index + 1);

System.out.print(tree[index] + " ");

inorderTraversal(2 \* index + 2);

}

public void preorderTraversal(int index) {

if (index >= size || tree[index] == null) return;

System.out.print(tree[index] + " ");

preorderTraversal(2 \* index + 1);

preorderTraversal(2 \* index + 2);

}

public void postorderTraversal(int index) {

if (index >= size || tree[index] == null) return;

postorderTraversal(2 \* index + 1);

postorderTraversal(2 \* index + 2);

System.out.print(tree[index] + " ");

}

}