Homework

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

import java.util.Arrays;

public class ArrayBinaryTree<T> {

private T[] tree;

private int size;

private static final int DEFAULT\_CAPACITY = 10;

public ArrayBinaryTree() {

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

size = 0;

}

private void ensureCapacity() {

if (size >= tree.length) {

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

}

}

public void add(T element) {

ensureCapacity();

tree[size] = element;

size++;

}

public T getRoot() {

return size > 0 ? tree[0] : null;

}

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 || index >= size) return null;

int parentIndex = (index - 1) / 2;

return tree[parentIndex];

}

public void printTree() {

for (int i = 0; i < size; i++) {

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

}

System.out.println();

}

public static void main(String[] args) {

ArrayBinaryTree<String> tree = new ArrayBinaryTree<>();

tree.add("A");

tree.add("B");

tree.add("C");

tree.add("D");

tree.add("E");

tree.add("F");

tree.add("G");

tree.printTree();

System.out.println("Root: " + tree.getRoot());

System.out.println("Left Child of A: " + tree.getLeftChild(0));

System.out.println("Right Child of A: " + tree.getRightChild(0));

System.out.println("Parent of E: " + tree.getParent(4));

}

}