Topics

* Create Position Interface
* Create Tree Interface
* Create AbstractTree Class.
* Create BinaryTree Interface
* Create AbstractBinaryTree Class
* Create Linked Binary Tree Using Linked List structure (Nodes)
* 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

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

class ArrayBinaryTree {

private Integer[] tree;

private int size;

public ArrayBinaryTree(int capacity) {

tree = new Integer[capacity];

size = 0; }

public void insert(int value) {

if (size < tree.length) {

tree[size] = value;

size++;

} else {

System.out.println("الشجرة ممتلئة، لا يمكن إدخال المزيد من العناصر.");

}

}

public int getLeftChildIndex(int index) {

return 2 \* index + 1;

}

public int getRightChildIndex(int index) {

return 2 \* index + 2;

}

public int getParentIndex(int index) {

if (index == 0) {

return -1;

}

return (index - 1) / 2;

}

public void display() {

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

System.out.println("Index " + i + ": " + tree[i]);

}

}

public static void main(String[] args) {

ArrayBinaryTree binaryTree = new ArrayBinaryTree(10);

binaryTree.insert(1);

binaryTree.insert(2);

binaryTree.insert(3);

binaryTree.insert(4);

binaryTree.insert(5);

binaryTree.display();

}

}