## 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.

class ArrayBinaryTree {

private int[] tree;

private int capacity;

public ArrayBinaryTree(int capacity) {

this.capacity = capacity;

this.tree = new int[capacity];

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

tree[i] = -1;

}

}

public void setRoot(int value) {

tree[0] = value;

}

public void setLeft(int parentIndex, int value) {

int leftIndex = 2 \* parentIndex + 1;

if (leftIndex < capacity) {

tree[leftIndex] = value;

} else {

System.out.println("");

}

}

public void setRight(int parentIndex, int value) {

int rightIndex = 2 \* parentIndex + 2;

if (rightIndex < capacity) {

tree[rightIndex] = value;

} else {

System.out.println("خ.");

}

}

public int get(int index) {

if (index < capacity) {

return tree[index];

}

return -1;

}

public void display() {

System.out.print("ت: ");

for (int value : tree) {

System.out.print((value != -1 ? value : "-") + " ");

}

System.out.println();

}

public static void main(String[] args) {

ArrayBinaryTree bt = new ArrayBinaryTree(15);

bt.setRoot(1);

bt.setLeft(0, 2);

bt.setRight(0, 3);

bt.setLeft(1, 4);

bt.setRight(1, 5);

bt.setLeft(2, 6);

bt.setRight(2, 7);

bt.display();ة

}

}