## الحل: import java.util.Arrays;

## public class ArrayBinaryTree<E> {

## private Object[] tree;

## private int size;

## private static final int DEFAULT\_CAPACITY = 10;

## public ArrayBinaryTree() {

## tree = new Object[DEFAULT\_CAPACITY];

## size = 0;

## }

## private void ensureCapacity(int index) {

## if (index >= tree.length) {

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

## }

## }

## public void addRoot(E e) {

## if (size != 0) throw new IllegalStateException("Root already exists");

## tree[0] = e;

## size = 1;

## }

## public void addLeft(int parentIndex, E e) {

## int leftChild = 2 \* parentIndex + 1;

## ensureCapacity(leftChild);

## if (tree[parentIndex] == null) throw new IllegalStateException("Parent does not exist");

## tree[leftChild] = e;

## size++;

## }

## public void addRight(int parentIndex, E e) {

## int rightChild = 2 \* parentIndex + 2;

## ensureCapacity(rightChild);

## if (tree[parentIndex] == null) throw new IllegalStateException("Parent does not exist");

## tree[rightChild] = e;

## size++;

## }

## public E get(int index) {

## if (index >= tree.length || tree[index] == null) return null;

## return (E) tree[index];

## }

## public void set(int index, E e) {

## if (index >= tree.length || tree[index] == null) throw new IllegalStateException("No element at index");

## tree[index] = e;

## }

## public void remove(int index) {

## if (index >= tree.length || tree[index] == null) throw new IllegalStateException("No element at index");

## tree[index] = null;

## removeChildren(index);

## }

## private void removeChildren(int index) {

## int leftChild = 2 \* index + 1;

## int rightChild = 2 \* index + 2;

## if (leftChild < tree.length) tree[leftChild] = null;

## if (rightChild < tree.length) tree[rightChild] = null;

## }

## public void printTree() {

## System.out.println(Arrays.toString(tree));

## }

## public static void main(String[] args) {

## ArrayBinaryTree<Integer> bt = new ArrayBinaryTree<>();

## bt.addRoot(10);

## bt.addLeft(0, 20);

## bt.addRight(0, 30);

## bt.addLeft(1, 40);

## bt.addRight(1, 50);

## bt.printTree();

## }

## }

1. **addRoot(E e)** – Adds a root element.
2. **addLeft(int parentIndex, E e)** – Adds a left child to the given parent index.
3. **addRight(int parentIndex, E e)** – Adds a right child to the given parent index.
4. **get(int index)** – Retrieves an element at a specific index.
5. **set(int index, E e)** – Updates an element at a given index.
6. **remove(int index)** – Removes an element and its children.
7. **printTree()** – Displays the tree in array form.

## 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.