CODE:

import java.util.ArrayList;

import java.util.List;

class BPlusTreeNode {

boolean isLeaf;

List<Integer> keys;

List<BPlusTreeNode> children;

// Constructor for a B+ Tree node

BPlusTreeNode(boolean isLeaf) {

this.isLeaf = isLeaf;

this.keys = new ArrayList<>();

this.children = new ArrayList<>();

}

}

class BPlusTree {

private BPlusTreeNode root;

private int degree; // Minimum degree (defines the range of children per node)

// Constructor for B+ Tree

public BPlusTree(int degree) {

this.root = new BPlusTreeNode(true);

this.degree = degree;

}

// Public method to insert key

public void insert(int key) {

BPlusTreeNode currentNode = this.root;

if (currentNode.keys.size() == (2 \* degree) - 1) {

// Split the root if it's full

BPlusTreeNode newRoot = new BPlusTreeNode(false);

this.root = newRoot;

newRoot.children.add(currentNode);

splitChild(newRoot, 0);

insertNonFull(newRoot, key);

} else {

insertNonFull(currentNode, key);

}

}

// Split a child node

private void splitChild(BPlusTreeNode parent, int index) {

BPlusTreeNode nodeToSplit = parent.children.get(index);

BPlusTreeNode newNode = new BPlusTreeNode(nodeToSplit.isLeaf);

int median = degree - 1;

parent.keys.add(index, nodeToSplit.keys.get(median)); // Move middle key up

parent.children.add(index + 1, newNode);

// Split keys and children

newNode.keys.addAll(nodeToSplit.keys.subList(median + 1, nodeToSplit.keys.size()));

nodeToSplit.keys.subList(median, nodeToSplit.keys.size()).clear();

if (!nodeToSplit.isLeaf) {

newNode.children.addAll(nodeToSplit.children.subList(degree, nodeToSplit.children.size()));

nodeToSplit.children.subList(degree, nodeToSplit.children.size()).clear();

}

}

// Insert into a non-full node

private void insertNonFull(BPlusTreeNode node, int key) {

if (node.isLeaf) {

// Find correct position for key

int i = node.keys.size() - 1;

node.keys.add(0); // Temporarily add a space for the new key

while (i >= 0 && key < node.keys.get(i)) {

node.keys.set(i + 1, node.keys.get(i));

i--;

}

node.keys.set(i + 1, key);

} else {

// Move down to the correct child

int i = node.keys.size() - 1;

while (i >= 0 && key < node.keys.get(i)) {

i--;

}

i++;

if (node.children.get(i).keys.size() == (2 \* degree) - 1) {

splitChild(node, i);

if (key > node.keys.get(i)) {

i++;

}

}

insertNonFull(node.children.get(i), key);

}

}

// Print the B+ tree structure

public void printTree(BPlusTreeNode node, int level) {

System.out.println("Level " + level + " " + node.keys);

if (!node.isLeaf) {

for (BPlusTreeNode child : node.children) {

printTree(child, level + 1);

}

}

}

// Public method to print the tree from the root

public void printTree() {

printTree(this.root, 0);

}

}

public class BPlusTreeExample {

public static void main(String[] args) {

BPlusTree bPlusTree = new BPlusTree(3); // B+ tree of degree 3 (can be changed)

// Insert some keys

for (int i = 0; i < 10; i++) {

bPlusTree.insert(i);

}

// Print the tree structure

bPlusTree.printTree();

// You can add more operations like search, etc.

}

}

OUTPUT:

