**DCIT 308: DATA STRUCTURES AND ALGORITHMS**

**ISAAC NII OKAI OKAI**

**10909577**

**CLASS EXERCISES**

**IMPLEMENTATION OF STACK USING AN ARRAY**

import java.util.Arrays;

public class StackArray {

private int[] array;

private int top;

public StackArray(int size) {

array = new int[size];

top = -1;

}

public void push(int value) {

if (top == array.length - 1) {

throw new RuntimeException("Stack is full");

}

array[++top] = value;

}

public int pop() {

if (top == -1) {

throw new RuntimeException("Stack is empty");

}

return array[top--];

}

public boolean isEmpty() {

return top == -1;

}

public static void main(String[] args) {

StackArray stack = new StackArray(5);

stack.push(1);

stack.push(2);

stack.push(3);

System.out.println(stack.pop()); // 3

System.out.println(stack.pop()); // 2

System.out.println(stack.pop()); // 1

}

}

**IMPLEMETATION OF STACK USING A LINKED LIST**

import java.util.LinkedList;

public class StackLinkedList {

private LinkedList<Integer> linkedList;

public StackLinkedList() {

linkedList = new LinkedList<>();

}

public void push(int value) {

linkedList.addFirst(value);

}

public int pop() {

return linkedList.removeFirst();

}

public boolean isEmpty() {

return linkedList.isEmpty();

}

public static void main(String[] args) {

StackLinkedList stack = new StackLinkedList();

stack.push(1);

stack.push(2);

stack.push(3);

System.out.println(stack.pop()); // 3

System.out.println(stack.pop()); // 2

System.out.println(stack.pop()); // 1

}

}

**IMPLEMENTING A QUEUE USING AN ARRAY**

import java.util.Arrays;

public class QueueArray2 {

private int[] array;

private int front;

private int rear;

public QueueArray2(int size) {

array = new int[size];

front = -1;

rear = -1;

}

public void enqueue(int value) {

if (rear == array.length - 1) {

int[] tempArray = new int[array.length \* 2];

System.arraycopy(array, 0, tempArray, 0, array.length);

array = tempArray;

}

rear++;

array[rear] = value;

}

public int dequeue() {

if (front == rear) {

throw new RuntimeException("Queue is empty");

}

front++;

return array[front];

}

public boolean isEmpty() {

return front == rear;

}

public static void main(String[] args) {

QueueArray2 queue = new QueueArray2(5);

queue.enqueue(1);

queue.enqueue(2);

queue.enqueue(3);

System.out.println(queue.dequeue()); // 1

System.out.println(queue.dequeue()); // 2

System.out.println(queue.dequeue()); // 3

}

}

**IMPLEMENTING A QUEUE USING A LINKED LIST**

import java.util.LinkedList;

public class QueueLinkedList {

private LinkedList<Integer> linkedList;

public QueueLinkedList() {

linkedList = new LinkedList<>();

}

public void enqueue(int value) {

linkedList.add(value);

}

public int dequeue() {

return linkedList.removeFirst();

}

public boolean isEmpty() {

return linkedList.isEmpty();

}

public static void main(String[] args) {

QueueLinkedList queue = new QueueLinkedList();

queue.enqueue(1);

queue.enqueue(2);

queue.enqueue(3);

System.out.println(queue.dequeue()); // 1

System.out.println(queue.dequeue()); // 2

System.out.println(queue.dequeue()); // 3

}

}

**IMPLEMENTING A BINARY TREE**

public class BinaryTree {

public Node root;

public BinaryTree() {

root = null;

}

public void add(int value) {

Node newNode = new Node(value);

if (root == null) {

root = newNode;

} else {

Node currentNode = root;

while (true) {

if (value < currentNode.value) {

if (currentNode.left == null) {

currentNode.left = newNode;

break;

} else {

currentNode = currentNode.left;

}

} else {

if (currentNode.right == null) {

currentNode.right = newNode;

break;

} else {

currentNode = currentNode.right;

}

}

}

}

}

public void printInOrder() {

printInOrder(root);

}

private void printInOrder(Node node) {

if (node == null) {

return;

}

printInOrder(node.left);

System.out.println(node.value);

printInOrder(node.right);

}

public static void main(String[] args) {

BinaryTree tree = new BinaryTree();

tree.add(10);

tree.add(5);

tree.add(15);

tree.add(2);

tree.add(7);

tree.add(12);

tree.add(17);

tree.printInOrder();

}

}

class Node {

public int value;

public Node left;

public Node right;

public Node(int value) {

this.value = value;

this.left = null;

this.right = null;

}

}