Assignment 7: Mastering Stacks and Queues

**Duration: 30 minutes**

Assignment Description:

Dive into the fundamental data structures of stacks and queues. Understand their applications and implementations in Java.

**Questions**:

Stacks and Queues Introduction: Explain the fundamental concepts of stacks and queues. Provide a real-world scenario where a stack is more suitable than a queue and vice versa.

**Solution** 1:

java

// Stacks and Queues Introduction: Stacks and queues are fundamental data structures.

// - A stack follows the Last-In-First-Out (LIFO) principle, making it suitable for undo mechanisms.

// - A queue follows the First-In-First-Out (FIFO) principle, ideal for tasks like task scheduling.

// Real-world Scenario: Consider a print queue in an office setting. It follows the FIFO principle, ensuring documents are printed in the order they were submitted.

class PrintQueue {

Queue<Document> documents;

PrintQueue() {

documents = new LinkedList<>();

}

}

Stack Implementation in Java: Implement a stack in Java and demonstrate how to push elements onto the stack, pop elements from the stack, and check for its emptiness. Include a Java code example for this implementation.

**Solution** 2:

java

// Stack Implementation in Java:

import java.util.Stack;

public class StackExample {

public static void main(String[] args) {

// Create a stack

Stack<Integer> stack = new Stack<>();

// Push elements onto the stack

stack.push(10);

stack.push(20);

stack.push(30);

// Pop elements from the stack

int popped = stack.pop();

System.out.println("Popped: " + popped);

// Check if the stack is empty

boolean isEmpty = stack.isEmpty();

System.out.println("Is Empty: " + isEmpty);

}

}

Queue Implementation in Java: Implement a queue in Java and illustrate how to enqueue elements into the queue, dequeue elements from the queue, and check if the queue is empty. Provide a Java code example for this queue implementation.

**Solution** 3:

java

// Queue Implementation in Java:

import java.util.LinkedList;

import java.util.Queue;

public class QueueExample {

public static void main(String[] args) {

// Create a queue

Queue<String> queue = new LinkedList<>();

// Enqueue elements into the queue

queue.offer("Alice");

queue.offer("Bob");

queue.offer("Charlie");

// Dequeue elements from the queue

String dequeued = queue.poll();

System.out.println("Dequeued: " + dequeued);

// Check if the queue is empty

boolean isEmpty = queue.isEmpty();

System.out.println("Is Empty: " + isEmpty);

}

}