2nd Assignment

Data Structure and Algorithm (Java)

Deadline: 26th April 2025 **Marks: 20**

Important Instructions:

No late submissions allowed.

Submission through email is not permitted in any case.

Plagiarism will not be tolerated and will be graded as **ZERO.**

Make a Word document, add the executable against each problem statement, and submit the document file on the portal within the deadline.

Scenario 1:

Simulate the Undo and Redo operations of a text editor using **two stacks** – one for undo and one for redo operations.

Requirements:

- Each action should be either insert text or delete text.
- Maintain two stacks:
  - **Undo Stack**: Stores performed actions.
  - **Redo Stack**: Stores undone actions.

Functionalities to Implement:

1. performAction(String type, String text) – Push to Undo stack and clear Redo stack.
2. undo() – Pop from Undo stack and push to Redo stack.
3. redo() – Pop from Redo stack and push back to Undo stack.
4. showHistory() – Display Undo stack content (last actions performed).

Conditions to Handle:

- Don't allow undo if the Undo stack is empty.
- Don't allow redo if the Redo stack is empty.
- After performing a new action, clear the Redo stack.

Scenario 2:

Build a simple helpdesk queue system where customers are served in the order they arrive (FIFO).

Requirements:

- Each customer has:
  - name (String)
  - issueDescription (String)
  - priority (Optional extension – normal or urgent)

Functionalities to Implement:

1. enqueue(String name, String issue) – Add a customer to the queue.
2. dequeue() – Serve the next customer (remove from queue).
3. peek() – View the next customer without removing.
4. isEmpty() – Check if the queue is empty.
5. displayQueue() – Display the list of all waiting customers.

Conditions to Handle:

- Don't dequeue if the queue is empty.
- Validate input (name and issue must not be blank).
- (Optional extension) Handle urgent tickets by placing them at the front.

Scenario 3:

Problem Statement:

Evaluate a **postfix expression** using a Stack.

Requirements:

- Accept input like: "5 6 + 2 *" → Should evaluate to 22.

Functionalities to Implement:

1. Parse and process tokens.
2. Push operands to the stack.
3. On encountering an operator, pop two operands and apply operation.
4. Return result.

Conditions to Handle:

- Handle division by zero.
- Validate postfix expression before evaluation.
- Return error if insufficient operands.

Scenario 4:

Problem Statement:

Manage aircraft takeoff using a **queue-based runway system** where planes take off in order of arrival.

Requirements:

Each aircraft has:

- flightNumber (String)
- destination (String)
- priority (optional extension)

Functionalities to Implement:

1. addFlight(String flightNumber, String destination)
2. authorizeTakeoff() – Dequeue flight for takeoff.
3. peekNextFlight() – Show the next flight in line.
4. displayQueue() – Show all waiting flights.

Conditions to Handle:

- If queue is empty, notify "No flights waiting."
- Ensure flight numbers are unique.
- (Optional extension) Handle priority-based queue.

----------End of Assignment! ----------