Group Assignment 2: Chat Message Manager System

Course	WIA1002 / WIB1002 Data Structure
Topics	Singly Linked List, Doubly Linked List, Stack
Submission Format	ONE java file named `ChatManager.java`
Group Size	5 students (If your group has fewer or more members, the group leader must distribute the work fairly to ensure equal contribution from each member.)
Time Limit	1 hour and 30 minutes (including submission, no extra times will be given)

Building a Lightweight Chat Manager for "QuickTalk" Messaging App

Your team has been hired as junior developers by **QuickTalk**, a startup launching a new lightweight messaging app designed for low-end mobile devices. Unlike traditional apps, **QuickTalk stores and manages chat messages entirely in memory**, using efficient data structures instead of heavy databases.

Your task is to **develop the in-memory message manager**, which will:

- Store messages efficiently
- Support inserting/removing messages at any position
- Let users move a **cursor** through messages (like scrolling)
- Provide **Undo** and **Redo** operations when users change their chat

The company is preparing for a **demo in 90 minutes**, and your team must build a working prototype quickly. To succeed, each group member will focus on a specific part so the system can be completed and tested within the deadline.

This simulation reflects a **real-world software sprint** where multiple developers contribute components that are tested in parallel.

Objective

Your group will build a **Chat Message Manager System** using:

- **Singly linked list** for message history
- **Doubly linked list with cursor** for navigation
- **Stack** for undo and redo operations

All work must be done in **one Java file**: `ChatManager.java`.

Tasks and Examples

Student 1 – Singly Linked List

Implement:

- `addFirst(String msg)`
- `addLast(String msg)`
- `removeFirst()`
- `print()`

Example Output:

```
list.addFirst("Hello");
list.addLast("How are you?");
list.removeFirst();
list.print();  // Output: How are you? -> null
```

Student 2 – Singly Linked List

Implement:

- `addAt(int index, String msg)`
- `removeAt(int index)`

Example Output:

```
list.addAt(1, "I'm fine");
list.removeAt(0);
list.print();  // Output: I'm fine -> null
```

Student 3 – Doubly Linked List with Cursor

Implement:

- `insertAtCursor(String msg)`
- `moveLeft()`, `moveRight()`
- `print()` showing cursor using brackets

Example Output:

```
history.insertAtCursor("Hi");
history.insertAtCursor("Bye");
history.moveLeft();
history.insertAtCursor("Wait");
history.print();  // Output: Hi <-> [Wait] <-> Bye <-> null
```

Student 4 – Undo/Redo with Stack

Implement:

- `perform(String action)`
- `undo()`
- `redo()`
- `printStacks()`

Example Output:

```
manager.perform("add:Hi");
manager.perform("remove:Bye");
System.out.println("Undo: " + manager.undo());
System.out.println("Redo: " + manager.redo());
manager.printStacks();
```

Student 5 – Testing and Output

Write the `main()` method to test the features of all other classes. Create one test block for each class:

- Call methods with sample data
- Print the results

Note:
You can start immediately by calling the methods, even if the other students haven't finished. The real methods will work once added later.

Example:

```
SinglyLinkedList list = new SinglyLinkedList();
list.addFirst("Hello");  // Safe to write now
list.print();
```

This allows testing to be written **in parallel** with the other tasks.

Time Allocation (Total: 1 Hour 30 minutes)

Student	Task	Estimated Time
1	Singly Linked List Basic	30 mins
2	Indexed List Ops	30 mins
3	Cursor-based Doubly Linked List	40 mins
4	Undo/Redo Stack	30 mins
5	Testing	40 mins

Rubric (10 marks)

Singly Linked List Basic	2
Indexed Insert/Remove	2
Doubly Linked List with Cursor	2
Undo/Redo Stack	2
Integration and Output (main method)	2

Important: Submission Instructions

- Submit only **ONE** Java file: `ChatManager.java`
- Add the following comment block at the top:

```
// Group Tutorial [Tutorial Number]
// Group Members:
// Student 1: [Name] - Singly Linked List
// Student 2: [Name] - Indexed List
// Student 3: [Name] - Doubly Linked List with Cursor
// Student 4: [Name] - Undo/Redo with Stack
// Student 5: [Name] - Testing
// Student X: [Name] - Absent
```

- Each student must label their section of code with their name
- Only the group leader submits the file `.java` file through Spectrum.
- ✕ No separate submissions for each member