**Scenario 4**

import java.util.LinkedList;

import java.util.Scanner;

import java.time.LocalDateTime;

import java.time.format.DateTimeFormatter;

class Comment {

    private String username;

    private String content;

    private LocalDateTime createdAt;

    private LocalDateTime editedAt;

    private static DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss");

    public Comment(String username, String content, LocalDateTime createdAt) {

        this.username = username;

        this.content = content;

        this.createdAt = createdAt;

        this.editedAt = null;

    }

    public String getUsername() {

        return username;

    }

    public String getContent() {

        return content;

    }

    public void setContent(String content) {

        this.content = content;

    }

    public void setEditedAt(LocalDateTime editedAt) {

        this.editedAt = editedAt;

    }

    @Override

    public String toString() {

        String timestamp = createdAt.format(formatter);

        String editInfo = editedAt != null ? " (edited at " + editedAt.format(formatter) + ")" : "";

        return username + ": " + content + " - " + timestamp + editInfo;

    }

}

class CommentSection {

    private LinkedList<Comment> comments;

    private Scanner scanner;

    private DateTimeFormatter formatter;

    public CommentSection() {

        comments = new LinkedList<>();

        scanner = new Scanner(System.in);

        formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss");

        addDefaultComments();

    }

    private void addDefaultComments() {

        addComment(new Comment("Ahmed", "This post is really insightful!", LocalDateTime.now().minusDays(5)));

        addComment(new Comment("Fatima", "I completely agree with the points raised.", LocalDateTime.now().minusDays(3)));

        addComment(new Comment("Bilal", "Could you elaborate more on the third point?", LocalDateTime.now().minusDays(2)));

        addComment(new Comment("Ayesha", "Great discussion happening here!", LocalDateTime.now().minusDays(1)));

        addComment(new Comment("Omar", "I have a different perspective on this topic.", LocalDateTime.now().minusHours(12)));

    }

    public void run() {

        int choice;

        do {

            displayMenu();

            choice = getIntInput("Enter your choice: ");

            switch (choice) {

                case 1:

                    addNewComment();

                    break;

                case 2:

                    editComment();

                    break;

                case 3:

                    deleteComment();

                    break;

                case 4:

                    searchCommentsByKeyword();

                    break;

                case 5:

                    displayCommentsChronological();

                    break;

                case 6:

                    displayCommentsReverse();

                    break;

                case 7:

                    System.out.println("Exiting program. Goodbye!");

                    break;

                default:

                    System.out.println("Invalid choice. Please try again.");

            }

            System.out.println();

        } while (choice != 7);

        scanner.close();

    }

    private void displayMenu() {

        System.out.println("\t\t\t\t\t\t\t=======================================");

        System.out.println("\t\t\t\t\t\t\t\tSOCIAL MEDIA COMMENT SECTION");

        System.out.println("\t\t\t\t\t\t\t=======================================");

        System.out.println("1. Add New Comment");

        System.out.println("2. Edit Comment");

        System.out.println("3. Delete Comment");

        System.out.println("4. Search Comments by Keyword");

        System.out.println("5. Display Comments (Chronological)");

        System.out.println("6. Display Comments (Reverse/Recent First)");

        System.out.println("7. Exit");

        System.out.println("=======================================");

    }

    private void addNewComment() {

        String username = getStringInput("Enter your username: ");

        String content = getStringInput("Enter your comment: ");

        Comment comment = new Comment(username, content, LocalDateTime.now());

        addComment(comment);

        System.out.println("Comment added successfully!");

    }

    private void addComment(Comment comment) {

        comments.add(comment);

    }

    private void editComment() {

        if (comments.isEmpty()) {

            System.out.println("No comments available to edit!");

            return;

        }

        displayCommentsChronological();

        int index = getIntInput("Enter index of comment to edit: ");

        if (index >= 0 && index < comments.size()) {

            String newContent = getStringInput("Enter new comment content: ");

            Comment comment = comments.get(index);

            comment.setContent(newContent);

            comment.setEditedAt(LocalDateTime.now());

            System.out.println("Comment edited successfully!");

        } else {

            System.out.println("Invalid index!");

        }

    }

    private void deleteComment() {

        if (comments.isEmpty()) {

            System.out.println("No comments available to delete!");

            return;

        }

        displayCommentsChronological();

        int index = getIntInput("Enter index of comment to delete: ");

        if (index >= 0 && index < comments.size()) {

            comments.remove(index);

            System.out.println("Comment deleted successfully!");

        } else {

            System.out.println("Invalid index!");

        }

    }

    private void searchCommentsByKeyword() {

        if (comments.isEmpty()) {

            System.out.println("No comments available to search!");

            return;

        }

        String keyword = getStringInput("Enter keyword to search: ");

        boolean found = false;

        System.out.println("\nSearch Results for '" + keyword + "':");

        System.out.println("---------------------------------------");

        for (int i = 0; i < comments.size(); i++) {

            Comment comment = comments.get(i);

            if (comment.getContent().toLowerCase().contains(keyword.toLowerCase()) ||

                comment.getUsername().toLowerCase().contains(keyword.toLowerCase())) {

                System.out.println(i + ": " + comment);

                found = true;

            }

        }

        if (!found) {

            System.out.println("No comments found containing '" + keyword + "'");

        }

    }

    private void displayCommentsChronological() {

        if (comments.isEmpty()) {

            System.out.println("No comments available!");

            return;

        }

        System.out.println("\nComments (Oldest First):");

        System.out.println("---------------------------------------");

        for (int i = 0; i < comments.size(); i++) {

            System.out.println(i + ": " + comments.get(i));

        }

    }

    private void displayCommentsReverse() {

        if (comments.isEmpty()) {

            System.out.println("No comments available!");

            return;

        }

        System.out.println("\nComments (Newest First):");

        System.out.println("---------------------------------------");

        for (int i = comments.size() - 1; i >= 0; i--) {

            System.out.println(i + ": " + comments.get(i));

        }

    }

    private int getIntInput(String prompt) {

        System.out.print(prompt);

        while (!scanner.hasNextInt()) {

            System.out.println("Please enter a valid number.");

            scanner.next();

            System.out.print(prompt);

        }

        int value = scanner.nextInt();

        scanner.nextLine();

        return value;

    }

    private String getStringInput(String prompt) {

        System.out.print(prompt);

        return scanner.nextLine();

    }

}

public class scenario4 {

    public static void main(String[] args) {

        CommentSection commentSection = new CommentSection();

        commentSection.run();

    }

}

**Time Complexity Analysis and Data Structure Comparison**

Time Complexity Analysis (LinkedList)

**Insertion**

• At the End (comments.add(comment)):  
 Time Complexity: O(1)  
 Reason: LinkedList maintains a reference to the tail, allowing constant-time appending.

• At the Beginning (comments.addFirst(comment)):  
 Time Complexity: O(1)  
 Reason: Only the head pointer is updated.

• At a Specific Index (comments.add(index, comment)):  
 Time Complexity: O(n)  
 Reason: Must traverse the list to reach the index.

**Deletion**

• First or Last Element (removeFirst(), removeLast()):  
 Time Complexity: O(1)

• From a Specific Index (comments.remove(index)):  
 Time Complexity: O(n)  
 Reason: Must traverse to the index first.

**Searching**

• Searching by Keyword or Username:  
 Time Complexity: O(n)  
 Reason: Must iterate through each comment.

Time Complexities (LinkedList):

|  |  |
| --- | --- |
| Operation | Time Complexity |
| Insert at end | O(1) |
| Insert at beginning | O(1) |
| Insert in middle/index | O(n) |
| Delete at beginning | O(1) |
| Delete at end | O(1) |
| Delete in middle/index | O(n) |
| Search | O(n) |
| Access by index | O(n) |

**Would ArrayList be a Better Choice?**

1. In Favor of ArrayList:

* • Fast random access (O(1)).
* • Efficient iteration due to memory locality.
* • Similar or faster search performance in practice.
* • Ideal for operations that involve reading and displaying data.

1. Against ArrayList:

* • Insertion/deletion at the beginning or middle is slower (O(n)).
* • Requires resizing when capacity is exceeded.

**Conclusion:**ArrayList is better for your scenario where comments are mostly appended at the end and frequently displayed or searched.