Exercise 5: Task Management System

1. Understand Linked Lists

Types of Linked Lists:

Singly Linked List

Each node points to the next node. One-directional.

Doubly Linked List

Each node points to both previous and next nodes. Two-directional.

Key Differences:

Singly Linked List:

Less memory (one pointer per node)

Simpler implementation

Traversal is only forward

Doubly Linked List:

Can traverse both directions

Slightly more memory overhead

Easier to delete a node from the middle

For this task management system, a singly linked list is sufficient and more efficient.

2. Setup

Task.java:

public class Task {

int taskId;

String taskName;

String status;

Task next;

public Task(int taskId, String taskName, String status) {

this.taskId = taskId;

this.taskName = taskName;

this.status = status;

this.next = null;

}

@Override

public String toString() {

return "[" + taskId + ", " + taskName + ", " + status + "]";

}

}

3. Implementation

TaskLinkedList.java

public class TaskLinkedList {

private Task head;

public void addTask(Task task) {

task.next = head;

head = task;

System.out.println("Task added: " + task.taskName);

}

public Task searchTask(int id) {

Task current = head;

while (current != null) {

if (current.taskId == id)

return current;

current = current.next;

}

return null;

}

public void displayTasks() {

if (head == null) {

System.out.println("No tasks available.");

return;

}

Task current = head;

while (current != null) {

System.out.println(current);

current = current.next;

}

}

public void deleteTask(int id) {

Task current = head, prev = null;

while (current != null && current.taskId != id) {

prev = current;

current = current.next;

}

if (current == null) {

System.out.println("Task not found.");

return;

}

if (prev == null) {

head = current.next; // deleting head

} else {

prev.next = current.next;

}

System.out.println("Task deleted: " + current.taskName);

}

}

Main.java:

public class Main {

public static void main(String[] args) {

TaskLinkedList taskList = new TaskLinkedList();

Task t1 = new Task(1, "Design UI", "Pending");

Task t2 = new Task(2, "Write Backend", "In Progress");

Task t3 = new Task(3, "Test System", "Pending");

taskList.addTask(t1);

taskList.addTask(t2);

taskList.addTask(t3);

System.out.println("\nAll Tasks:");

taskList.displayTasks();

System.out.println("\nSearching for Task ID 2:");

Task found = taskList.searchTask(2);

System.out.println(found != null ? found : "Task not found.");

System.out.println("\nDeleting Task ID 1:");

taskList.deleteTask(1);

System.out.println("\nRemaining Tasks:");

taskList.displayTasks();

}

}

4. Analysis

Time Complexity of Singly Linked List Operations:

Operation Time Complexity

Add O(1)

Search O(n)

Traverse O(n)

Delete O(n)