**Exercise 5: Task Management System**

**Scenario:**  
You are developing a task management system where tasks need to be added, deleted, and traversed efficiently.

**1. Understand Linked Lists:**

Types of Linked Lists:

* Singly Linked List: Each node contains data and a reference to the next node.
* Doubly Linked List: Each node has references to both the next and the previous nodes.

Why Linked Lists?

* They are dynamic, allowing efficient memory usage.
* Insertion and deletion are faster compared to arrays (especially in the middle of the list).

**2. Setup:**

Task Class:  
Attributes:

* taskId (int)
* taskName (String)
* status (String)

TaskManager Class:  
Manages a singly linked list of Task nodes.

**3. Implementation:**

**//TaskManagement.java**

public class TaskManagement {

public static void main(String[] args) {

TaskManager manager = new TaskManager();

manager.addTask(1, "Design UI", "Pending");

manager.addTask(2, "Build Backend", "In Progress");

manager.addTask(3, "Write Tests", "Pending");

System.out.println("All Tasks:");

manager.displayTasks();

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

Task found = manager.searchTask(2);

if (found != null) {

System.out.println("Found: " + found.taskName + ", Status: " + found.status);

} else {

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

}

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

manager.deleteTask(1);

System.out.println("\nUpdated Task List:");

manager.displayTasks();

}

}

//Task.java

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;

}

}

//TaskManager.java

class TaskManager {

Task head = null;

public void addTask(int id, String name, String status) {

Task newTask = new Task(id, name, status);

if (head == null) {

head = newTask;

} else {

Task temp = head;

while (temp.next != null) {

temp = temp.next;

}

temp.next = newTask;

}

}

public Task searchTask(int id) {

Task temp = head;

while (temp != null) {

if (temp.taskId == id) return temp;

temp = temp.next;

}

return null;

}

public void displayTasks() {

Task temp = head;

while (temp != null) {

System.out.println("ID: " + temp.taskId + ", Name: " + temp.taskName + ", Status: " + temp.status);

temp = temp.next;

}

}

public void deleteTask(int id) {

if (head == null) return;

if (head.taskId == id) {

head = head.next;

return;

}

Task current = head;

Task prev = null;

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

prev = current;

current = current.next;

}

if (current != null) {

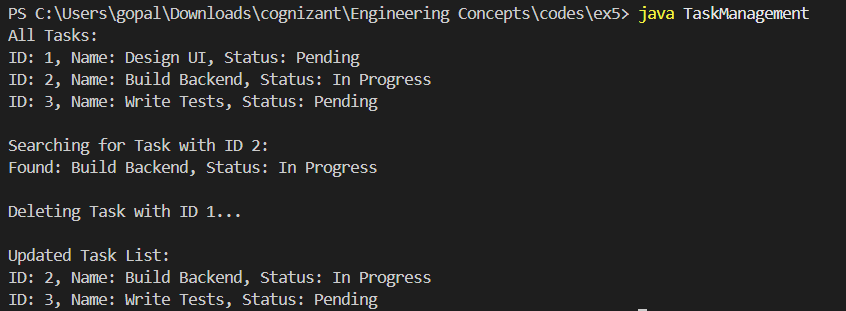
prev.next = current.next;

}

}

}

**Output:**



**4. Analysis:**

Time Complexity:

* Add Task: O(n) – traverses to the end.
* Search Task: O(n) – searches sequentially.
* Traverse: O(n) – visits all nodes.
* Delete Task: O(n) – needs to search then update links.

Advantages of Linked Lists:

* No memory wastage
* Dynamic in size
* Efficient insertions/deletions

Limitations:

* Slower search compared to arrays
* Extra memory for node pointers

**Conclusion:**  
Using a singly linked list offers flexibility and efficiency in dynamic task management scenarios. It’s ideal for systems where insertions and deletions are frequent.