**Exercise 5: Task Management System**

**1. Understand Linked Lists**

**Types of linked lists:**

* **Singly Linked List:** Each node points to the next node.
* **Doubly Linked List:** Each node points to both the next and previous nodes.

**2. Setup**

**Create a Task class:**

public class Task {

private int taskId;

private String taskName;

private String status;

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

this.taskId = taskId;

this.taskName = taskName;

this.status = status;

}

// Getters and Setters

public int getTaskId() {

return taskId;

}

public void setTaskId(int taskId) {

this.taskId = taskId;

}

public String getTaskName() {

return taskName;

}

public void setTaskName(String taskName) {

this.taskName = taskName;

}

public String getStatus() {

return status;

}

public void setStatus(String status) {

this.status = status;

}

}

**3. Implementation**

**Implement a singly linked list to manage tasks:**

public class TaskManagement {

private class Node {

Task task;

Node next;

Node(Task task) {

this.task = task;

this.next = null;

}

}

private Node head;

public TaskManagement() {

head = null;

}

public void addTask(Task task) {

Node newNode = new Node(task);

if (head == null) {

head = newNode;

} else {

Node temp = head;

while (temp.next != null) {

temp = temp.next;

}

temp.next = newNode;

}

}

public Task searchTask(int taskId) {

Node temp = head;

while (temp != null) {

if (temp.task.getTaskId() == taskId) {

return temp.task;

}

temp = temp.next;

}

return null;

}

public void deleteTask(int taskId) {

if (head == null) {

return;

}

if (head.task.getTaskId() == taskId) {

head = head.next;

return;

}

Node temp = head;

while (temp.next != null && temp.next.task.getTaskId() != taskId) {

temp = temp.next;

}

if (temp.next != null) {

temp.next = temp.next.next;

}

}

public void traverseTasks() {

Node temp = head;

while (temp != null) {

System.out.println(temp.task.getTaskName() + ": " + temp.task.getStatus());

temp = temp.next;

}

}

public static void main(String[] args) {

TaskManagement management = new TaskManagement();

management.addTask(new Task(1, "Task 1", "Pending"));

management.addTask(new Task(2, "Task 2", "Completed"));

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

management.traverseTasks();

System.out.println("Search Task:");

System.out.println(management.searchTask(2).getTaskName());

management.deleteTask(1);

System.out.println("After Deletion:");

management.traverseTasks();

}

}

**4. Analysis**

**Time complexity:**

* **Add:** O(n) - requires traversal to the end
* **Search:** O(n)
* **Traverse:** O(n)
* **Delete:** O(n)

**Advantages:**

Linked lists allow dynamic resizing and efficient insertions/deletions compared to arrays, especially when the exact number of elements is unknown or frequently changing.