**Exercise 5: Task Management System**

**Code:**

**TaskManagementSystem.java**

import java.util.Scanner;

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;

}

public String toString() {

return "ID: " + taskId + ", Name: " + taskName + ", Status: " + status;

}

}

class TaskList {

Task head;

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

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

if (head == null) {

head = newTask;

} else {

Task current = head;

while (current.next != null) {

current = current.next;

}

current.next = newTask;

}

System.out.println("Task added successfully.");

}

public void searchTask(int id) {

Task current = head;

while (current != null) {

if (current.taskId == id) {

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

return;

}

current = current.next;

}

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

}

public void traverseTasks() {

if (head == null) {

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

return;

}

System.out.println("\n--- Task List ---");

Task current = head;

while (current != null) {

System.out.println(current);

current = current.next;

}

}

public void deleteTask(int id) {

if (head == null) {

System.out.println("No tasks to delete.");

return;

}

if (head.taskId == id) {

head = head.next;

System.out.println("Task deleted successfully.");

return;

}

Task current = head;

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

current = current.next;

}

if (current.next == null) {

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

} else {

current.next = current.next.next;

System.out.println("Task deleted successfully.");

}

}

public void printAnalysis() {

System.out.println("\n--- Time Complexity Analysis ---");

System.out.println("Add Task -> O(n) [adding at end]");

System.out.println("Search Task -> O(n)");

System.out.println("Traverse -> O(n)");

System.out.println("Delete Task -> O(n)");

System.out.println("\n--- Advantages of Linked Lists Over Arrays ---");

System.out.println("1. Dynamic size: no need to define capacity in advance.");

System.out.println("2. Efficient insertions and deletions (no shifting required).");

System.out.println("3. Better memory utilization for varying data sizes.");

System.out.println("4. Suitable for applications with frequent insert/delete operations.");

}

}

public class TaskManagementSystem {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

TaskList taskList = new TaskList();

while (true) {

System.out.println("\n--- Task Management Menu ---");

System.out.println("1. Add Task");

System.out.println("2. Search Task by ID");

System.out.println("3. Display All Tasks");

System.out.println("4. Delete Task by ID");

System.out.println("5. View Time Complexity Analysis");

System.out.println("6. Exit");

System.out.print("Choose an option: ");

int choice = sc.nextInt();

switch (choice) {

case 1 -> {

System.out.print("Enter Task ID: ");

int id = sc.nextInt();

sc.nextLine();

System.out.print("Enter Task Name: ");

String name = sc.nextLine();

System.out.print("Enter Status: ");

String status = sc.nextLine();

taskList.addTask(id, name, status);

}

case 2 -> {

System.out.print("Enter Task ID to search: ");

int id = sc.nextInt();

taskList.searchTask(id);

}

case 3 -> taskList.traverseTasks();

case 4 -> {

System.out.print("Enter Task ID to delete: ");

int id = sc.nextInt();

taskList.deleteTask(id);

}

case 5 -> taskList.printAnalysis();

case 6 -> {

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

return;

}

default -> System.out.println("Invalid option!");

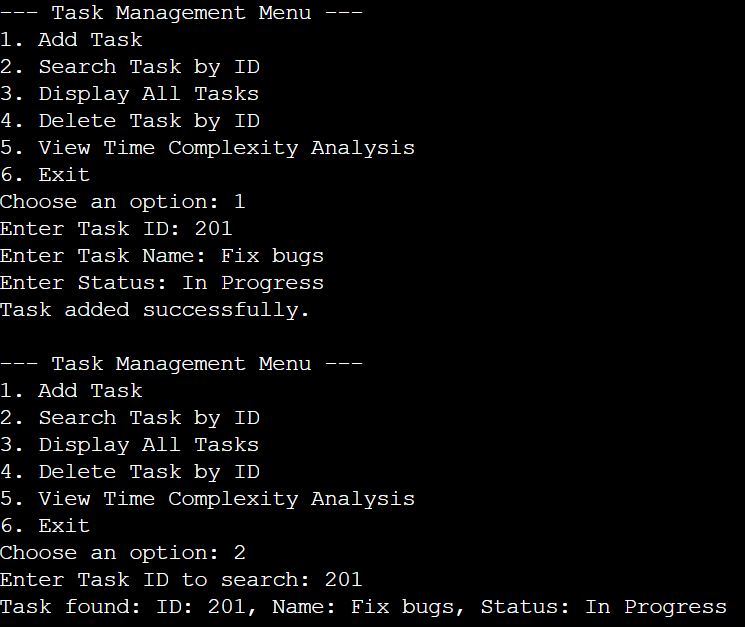
}

}

}

}

**Output:**

****

