**Name:** Paarthivi D

**Superset ID:** 6410961

**DN 4.0 - Java FSE Additional Hands-on**

**WEEK – 1**

**Algorithms and Data Structures:**

**Exercise 1: Inventory Management System**

**Code:**

package inventoryManagementSystem;

import java.util.HashMap;

import java.util.Map;

// Product Class

class Product {

private String productId;

private String productName;

private int quantity;

private double price;

public Product(String productId, String productName, int quantity, double price) {

this.productId = productId;

this.productName = productName;

this.quantity = quantity;

this.price = price;

}

// Getters & Setters

public String getProductId() { return productId; }

public String getProductName() { return productName; }

public int getQuantity() { return quantity; }

public double getPrice() { return price; }

public void setProductName(String name) { this.productName = name; }

public void setQuantity(int quantity) { this.quantity = quantity; }

public void setPrice(double price) { this.price = price; }

public void displayProduct() {

System.*out*.println(" " + productId + " | " + productName + " | Qty: " + quantity + " | ₹" + price);

}

}

// Inventory System

class Inventory {

private Map<String, Product> productMap = new HashMap<>();

// Add Product

public void addProduct(Product product) {

if (productMap.containsKey(product.getProductId())) {

System.*out*.println("Product already exists.");

} else {

productMap.put(product.getProductId(), product);

System.*out*.println("Product added.");

}

}

// Update Product

public void updateProduct(String productId, String name, int quantity, double price) {

Product product = productMap.get(productId);

if (product != null) {

product.setProductName(name);

product.setQuantity(quantity);

product.setPrice(price);

System.*out*.println("Product updated.");

} else {

System.*out*.println("Product not found.");

}

}

// Delete Product

public void deleteProduct(String productId) {

if (productMap.remove(productId) != null) {

System.*out*.println("Product deleted.");

} else {

System.*out*.println("Product not found.");

}

}

// Display all products

public void showAllProducts() {

System.*out*.println("\nInventory List:");

for (Product p : productMap.values()) {

p.displayProduct();

}

}

}

// Main class to test

public class InventoryManagementSystem {

public static void main(String[] args) {

Inventory inventory = new Inventory();

// Add products

inventory.addProduct(new Product("P101", "Laptop", 10, 75000.0));

inventory.addProduct(new Product("P102", "Monitor", 20, 15000.0));

inventory.addProduct(new Product("P103", "Mouse", 50, 500.0));

// Show products

inventory.showAllProducts();

// Update a product

inventory.updateProduct("P102", "LED Monitor", 25, 15500.0);

// Delete a product

inventory.deleteProduct("P103");

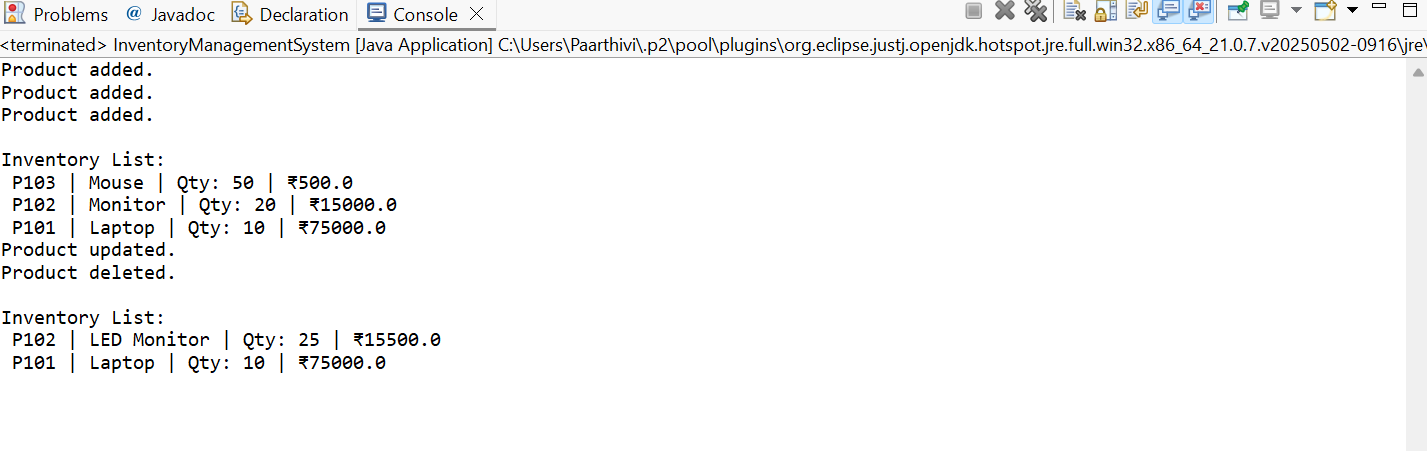
// Show updated inventory

inventory.showAllProducts();

}

}

**Output Screenshot:**

****

**Exercise 3: Sorting Customer Orders**

**Code:**

package customerOrderSorting;

public class CustomerOrderSorting {

// Order class

static class Order {

String orderId;

String customerName;

double totalPrice;

public Order(String orderId, String customerName, double totalPrice) {

this.orderId = orderId;

this.customerName = customerName;

this.totalPrice = totalPrice;

}

public void display() {

System.*out*.println("Order ID: " + orderId + ", Name: " + customerName + ", Total: ₹" + totalPrice);

}

}

// Bubble Sort

public static void bubbleSort(Order[] orders) {

int n = orders.length;

for (int i = 0; i < n - 1; i++) {

boolean swapped = false;

for (int j = 0; j < n - i - 1; j++) {

if (orders[j].totalPrice > orders[j + 1].totalPrice) {

Order temp = orders[j];

orders[j] = orders[j + 1];

orders[j + 1] = temp;

swapped = true;

}

}

if (!swapped) break; // Optimization: stop if already sorted

}

}

// Quick Sort

public static void quickSort(Order[] orders, int low, int high) {

if (low < high) {

int pi = *partition*(orders, low, high);

*quickSort*(orders, low, pi - 1);

*quickSort*(orders, pi + 1, high);

}

}

private static int partition(Order[] orders, int low, int high) {

double pivot = orders[high].totalPrice;

int i = low - 1;

for (int j = low; j < high; j++) {

if (orders[j].totalPrice < pivot) {

i++;

Order temp = orders[i];

orders[i] = orders[j];

orders[j] = temp;

}

}

Order temp = orders[i + 1];

orders[i + 1] = orders[high];

orders[high] = temp;

return i + 1;

}

// Helper: Display order list

public static void printOrders(String message, Order[] orders) {

System.*out*.println("\n" + message);

for (Order o : orders) o.display();

}

// Main

public static void main(String[] args) {

Order[] orders = {

new Order("O101", "Aarav", 3500.00),

new Order("O102", "Meera", 999.99),

new Order("O103", "Karan", 4250.75),

new Order("O104", "Paarthivi", 1999.00),

new Order("O105", "Dev", 850.00)

};

// Copy orders to use the same data for both sorts

Order[] bubbleSorted = orders.clone();

Order[] quickSorted = orders.clone();

// Perform Bubble Sort

*bubbleSort*(bubbleSorted);

*printOrders*("Orders sorted by Bubble Sort (Low to High):", bubbleSorted);

// Perform Quick Sort

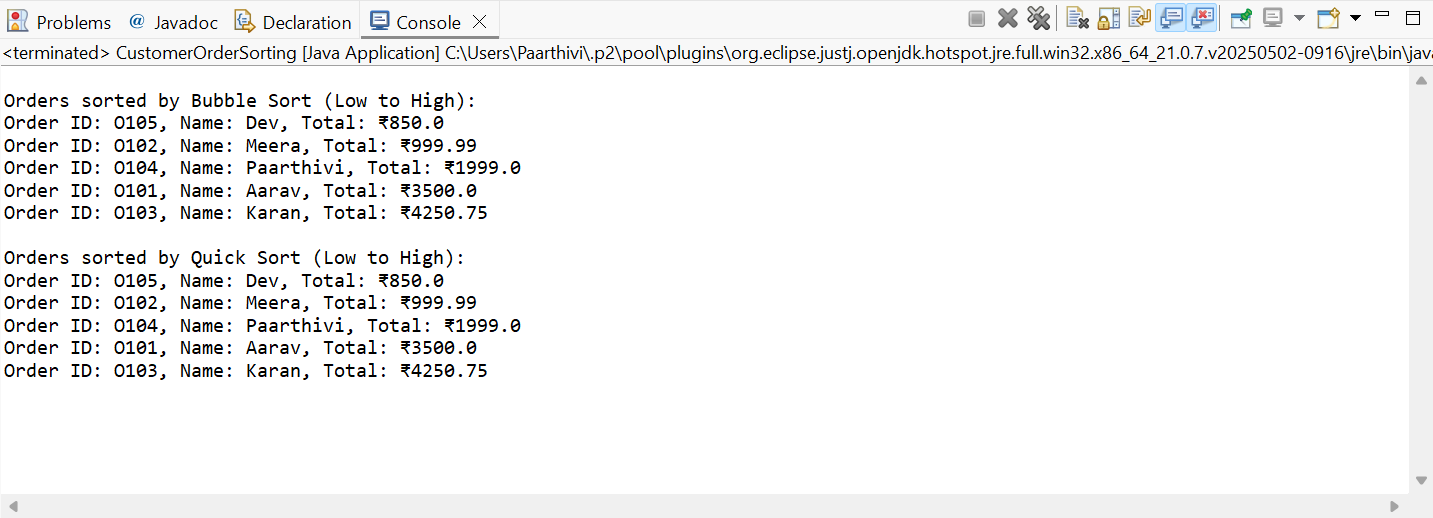
*quickSort*(quickSorted, 0, quickSorted.length - 1);

*printOrders*("Orders sorted by Quick Sort (Low to High):", quickSorted);

}

}

**Output Screenshot:**



**Exercise 4: Employee Management System**

**Code:**

package employeeManagementSystem;

import java.util.Scanner;

public class EmployeeManagementSystem {

// Employee Class

static class Employee {

String employeeId;

String name;

String position;

double salary;

public Employee(String employeeId, String name, String position, double salary) {

this.employeeId = employeeId;

this.name = name;

this.position = position;

this.salary = salary;

}

public void display() {

System.*out*.println("ID: " + employeeId + ", Name: " + name + ", Role: " + position + ", Salary: ₹" + salary);

}

}

// Array-based Implementation

static class EmployeeArrayManager {

private Employee[] employees;

private int size;

public EmployeeArrayManager(int capacity) {

employees = new Employee[capacity];

size = 0;

}

// Add Employee

public void addEmployee(Employee emp) {

if (size >= employees.length) {

System.*out*.println("Array full. Cannot add more employees.");

return;

}

employees[size++] = emp;

System.*out*.println("Employee added.");

}

// Search by ID

public Employee searchById(String id) {

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

if (employees[i].employeeId.equals(id)) {

return employees[i];

}

}

return null;

}

// Traverse/Display All

public void displayAll() {

System.*out*.println("\nEmployee List:");

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

employees[i].display();

}

}

// Delete by ID

public void deleteById(String id) {

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

if (employees[i].employeeId.equals(id)) {

// Shift remaining elements

for (int j = i; j < size - 1; j++) {

employees[j] = employees[j + 1];

}

employees[--size] = null;

System.*out*.println("Employee deleted.");

return;

}

}

System.*out*.println("Employee not found.");

}

}

// Test with sample data

public static void main(String[] args) {

EmployeeArrayManager manager = new EmployeeArrayManager(5);

manager.addEmployee(new Employee("E001", "Paarthivi", "Engineer", 65000));

manager.addEmployee(new Employee("E002", "Aarav", "Manager", 85000));

manager.addEmployee(new Employee("E003", "Meera", "Analyst", 55000));

manager.displayAll();

System.*out*.println("\nSearching for E002...");

Employee found = manager.searchById("E002");

if (found != null) found.display();

else System.*out*.println("Employee not found.");

System.*out*.println("\nDeleting E003...");

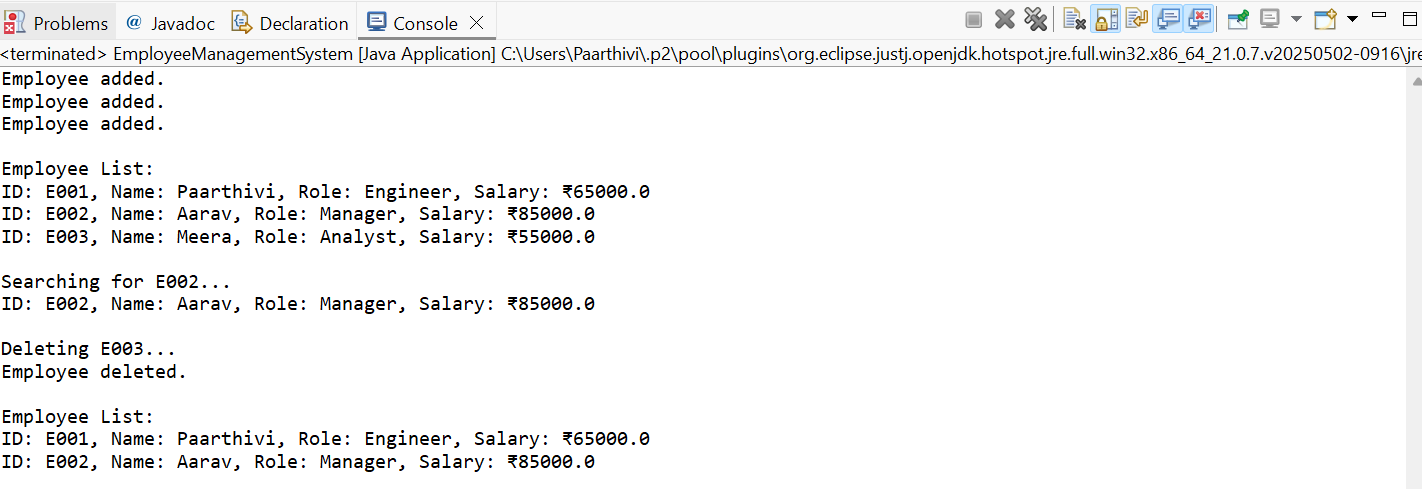
manager.deleteById("E003");

manager.displayAll();

}

}

**Output Screenshot:**



**Exercise 5: Task Management System**

**Code:**

package taskManagementSystem;

public class TaskManagementSystem {

// Task class

static class Task {

String taskId;

String taskName;

String status;

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

this.taskId = taskId;

this.taskName = taskName;

this.status = status;

}

public void display() {

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

}

}

// Singly Linked List Node

static class Node {

Task task;

Node next;

public Node(Task task) {

this.task = task;

this.next = null;

}

}

// Linked List implementation

static class TaskLinkedList {

private Node head;

// Add task to end

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;

}

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

}

// Search task by ID

public Task searchTask(String taskId) {

Node temp = head;

while (temp != null) {

if (temp.task.taskId.equals(taskId)) {

return temp.task;

}

temp = temp.next;

}

return null;

}

// Delete task by ID

public void deleteTask(String taskId) {

if (head == null) {

System.*out*.println("Task list is empty.");

return;

}

if (head.task.taskId.equals(taskId)) {

head = head.next;

System.*out*.println("Task deleted from head.");

return;

}

Node prev = head;

Node current = head.next;

while (current != null) {

if (current.task.taskId.equals(taskId)) {

prev.next = current.next;

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

return;

}

prev = current;

current = current.next;

}

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

}

// Traverse and display tasks

public void traverseTasks() {

System.*out*.println("\nTask List:");

Node temp = head;

while (temp != null) {

temp.task.display();

temp = temp.next;

}

}

}

// Main class

public static void main(String[] args) {

TaskLinkedList taskList = new TaskLinkedList();

// Add tasks

taskList.addTask(new Task("T101", "Design UI", "Pending"));

taskList.addTask(new Task("T102", "Develop Backend", "In Progress"));

taskList.addTask(new Task("T103", "Write Documentation", "Completed"));

// Display all

taskList.traverseTasks();

// Search

System.*out*.println("\nSearching for task T102...");

Task found = taskList.searchTask("T102");

if (found != null) {

found.display();

} else {

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

}

// Delete a task

System.*out*.println("\nDeleting task T101...");

taskList.deleteTask("T101");

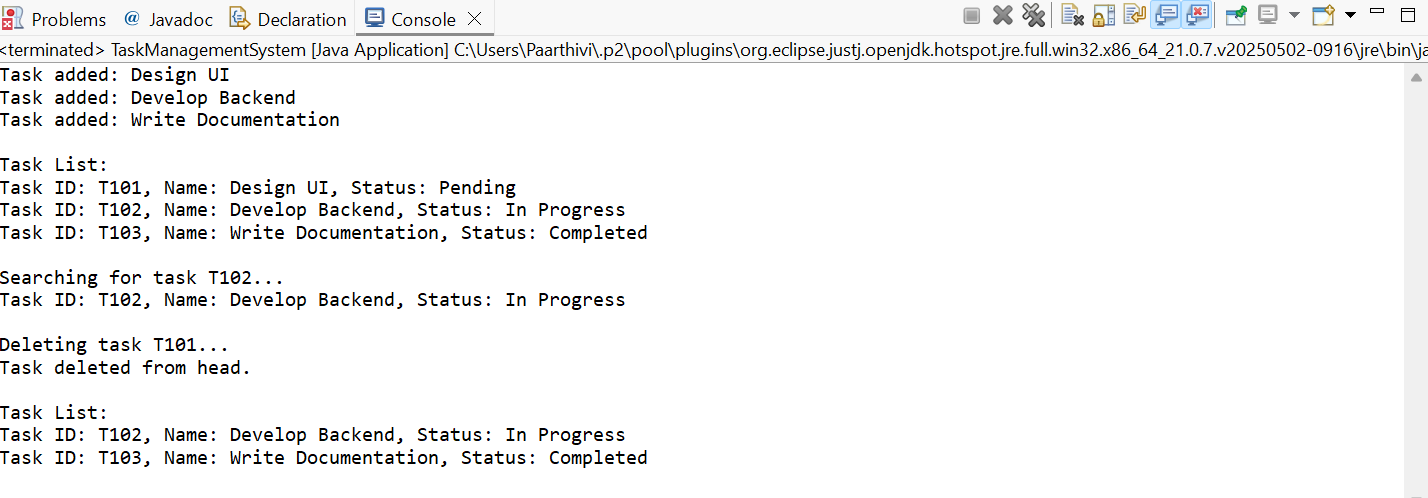
// Display updated list

taskList.traverseTasks();

}

}

**Output Screenshot:**



**Exercise 6: Library Management System**

**Code:**

package libraryManagementSystem;

import java.util.Arrays;

import java.util.Comparator;

public class LibraryManagementSystem {

// Book class

static class Book {

String bookId;

String title;

String author;

public Book(String bookId, String title, String author) {

this.bookId = bookId;

this.title = title;

this.author = author;

}

public void display() {

System.*out*.println("ID: " + bookId + ", Title: " + title + ", Author: " + author);

}

}

// Linear Search by title

public static Book linearSearch(Book[] books, String title) {

for (Book book : books) {

if (book.title.equalsIgnoreCase(title)) {

return book;

}

}

return null;

}

// Binary Search by title (array must be sorted)

public static Book binarySearch(Book[] books, String title) {

int low = 0;

int high = books.length - 1;

while (low <= high) {

int mid = (low + high) / 2;

int cmp = books[mid].title.compareToIgnoreCase(title);

if (cmp == 0) return books[mid];

else if (cmp < 0) low = mid + 1;

else high = mid - 1;

}

return null;

}

// Helper: Display all books

public static void displayAllBooks(Book[] books) {

System.*out*.println("\nBook List:");

for (Book b : books) {

b.display();

}

}

// Main method

public static void main(String[] args) {

Book[] books = {

new Book("B001", "The Alchemist", "Paulo Coelho"),

new Book("B002", "The Silent Patient", "Alex Michaelides"),

new Book("B003", "To Kill a Mockingbird", "Harper Lee"),

new Book("B004", "1984", "George Orwell"),

new Book("B005", "Sapiens", "Yuval Noah Harari")

};

// Display unsorted list

*displayAllBooks*(books);

// Linear Search

String searchTitle1 = "Sapiens";

System.*out*.println("\nLinear Search for: " + searchTitle1);

Book found1 = *linearSearch*(books, searchTitle1);

if (found1 != null) found1.display();

else System.*out*.println("Book not found.");

// Sort books by title for binary search

Arrays.*sort*(books, Comparator.*comparing*(b -> b.title.toLowerCase()));

// Display sorted list

*displayAllBooks*(books);

//Binary Search

String searchTitle2 = "1984";

System.*out*.println("\nBinary Search for: " + searchTitle2);

Book found2 = *binarySearch*(books, searchTitle2);

if (found2 != null) found2.display();

else System.*out*.println("Book not found.");

}

}

**Output Screenshot:**

