**Exercise 1: Inventory Management System**

import java.util.HashMap;

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;

    }

    public String getProductId() { return productId; }

    public String getProductName() { return productName; }

    public int getQuantity() { return quantity; }

    public double getPrice() { return price; }

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

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

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

    @Override

    public String toString() {

        return "Product{" +

                "ID='" + productId + '\'' +

                ", Name='" + productName + '\'' +

                ", Quantity=" + quantity +

                ", Price=" + price +

                '}';

    }

}

class Inventory {

    private HashMap<String, Product> products;

    public Inventory() {

        products = new HashMap<>();

    }

    public void addProduct(Product product) {

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

    }

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

        Product product = products.get(productId);

        if (product != null) {

            product.setProductName(name);

            product.setQuantity(quantity);

            product.setPrice(price);

        } else {

            System.out.println("Product not found: " + productId);

        }

    }

    public void deleteProduct(String productId) {

        if (products.remove(productId) == null) {

            System.out.println("Product not found: " + productId);

        }

    }

    public void displayInventory() {

        if (products.isEmpty()) {

            System.out.println("Inventory is empty.");

        } else {

            for (Product product : products.values()) {

                System.out.println(product);

            }

        }

    }

}

public class InventoryManagementSystem {

    public static void main(String[] args) {

        Inventory inventory = new Inventory();

        inventory.addProduct(new Product("P001", "Laptop", 10, 999.99));

        inventory.addProduct(new Product("P002", "Monitor", 20, 199.99));

        inventory.addProduct(new Product("P003", "Keyboard", 50, 49.99));

        System.out.println("Inventory after adding products:");

        inventory.displayInventory();

        inventory.updateProduct("P001", "Gaming Laptop", 8, 1099.99);

        inventory.deleteProduct("P002");

        System.out.println("\nInventory after update and delete:");

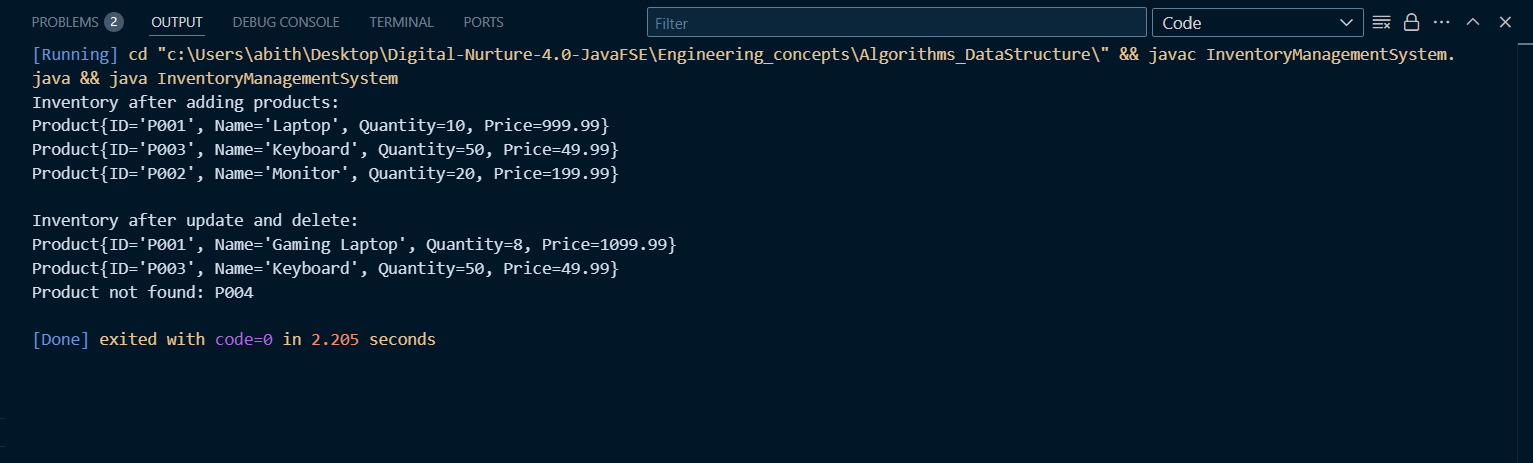
        inventory.displayInventory();

        inventory.deleteProduct("P004");

    }

}

**OUTPUT**



**Exercise 2: E-commerce Platform Search Function**

import java.util.Arrays;

class Product {

    String productId;

    String productName;

    String category;

    public Product(String productId, String productName, String category) {

        this.productId = productId;

        this.productName = productName;

        this.category = category;

    }

    public String toString() {

        return "[" + productId + ", " + productName + ", " + category + "]";

    }

}

public class EcommerceSearch {

    // Linear Search

    public static int linearSearch(Product[] products, String targetName) {

        for (int i = 0; i < products.length; i++) {

            if (products[i].productName.equalsIgnoreCase(targetName)) {

                return i;

            }

        }

        return -1;

    }

    // Binary Search (assumes array sorted by productName)

    public static int binarySearch(Product[] products, String targetName) {

        int left = 0, right = products.length - 1;

        while (left <= right) {

            int mid = left + (right - left) / 2;

            int cmp = products[mid].productName.compareToIgnoreCase(targetName);

            if (cmp == 0) return mid;

            if (cmp < 0) left = mid + 1;

            else right = mid - 1;

        }

        return -1;

    }

    public static void main(String[] args) {

        Product[] products = {

            new Product("P001", "Laptop", "Electronics"),

            new Product("P002", "Phone", "Electronics"),

            new Product("P003", "Shoes", "Fashion"),

            new Product("P004", "Book", "Books")

        };

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

        int idx1 = linearSearch(products, "Phone");

        System.out.println(idx1 >= 0 ? products[idx1] : "Product not found");

        // Sort products by productName for binary search

        Arrays.sort(products, (a, b) -> a.productName.compareToIgnoreCase(b.productName));

        System.out.println("\nBinary Search:");

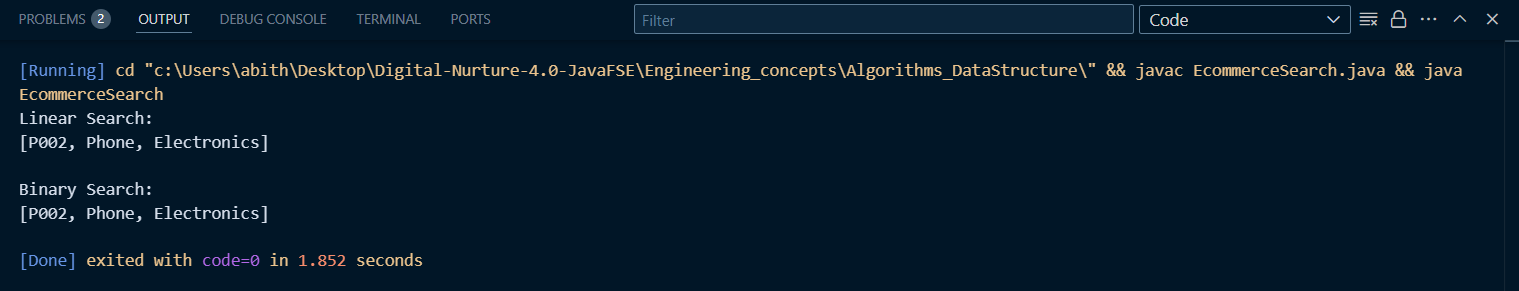
        int idx2 = binarySearch(products, "Phone");

        System.out.println(idx2 >= 0 ? products[idx2] : "Product not found");

    }

}

**OUTPUT:**



**Exercise 3: Sorting Customer Orders**

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 String toString() {

        return "[" + orderId + ", " + customerName + ", $" + totalPrice + "]";

    }

}

public class OrderSorting {

    // Bubble Sort

    public static void bubbleSort(Order[] orders) {

        for (int i = 0; i < orders.length - 1; i++) {

            for (int j = 0; j < orders.length - 1 - i; j++) {

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

                    Order temp = orders[j];

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

                    orders[j + 1] = temp;

                }

            }

        }

    }

    // 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;

    }

    public static void printOrders(Order[] orders) {

        for (Order o : orders) {

            System.out.println(o);

        }

    }

    public static void main(String[] args) {

        Order[] orders = {

            new Order("O001", "Alice", 500),

            new Order("O002", "Bob", 1500),

            new Order("O003", "Charlie", 800),

            new Order("O004", "Daisy", 1200)

        };

        System.out.println("Bubble Sort:");

        bubbleSort(orders);

        printOrders(orders);

        System.out.println("\nQuick Sort:");

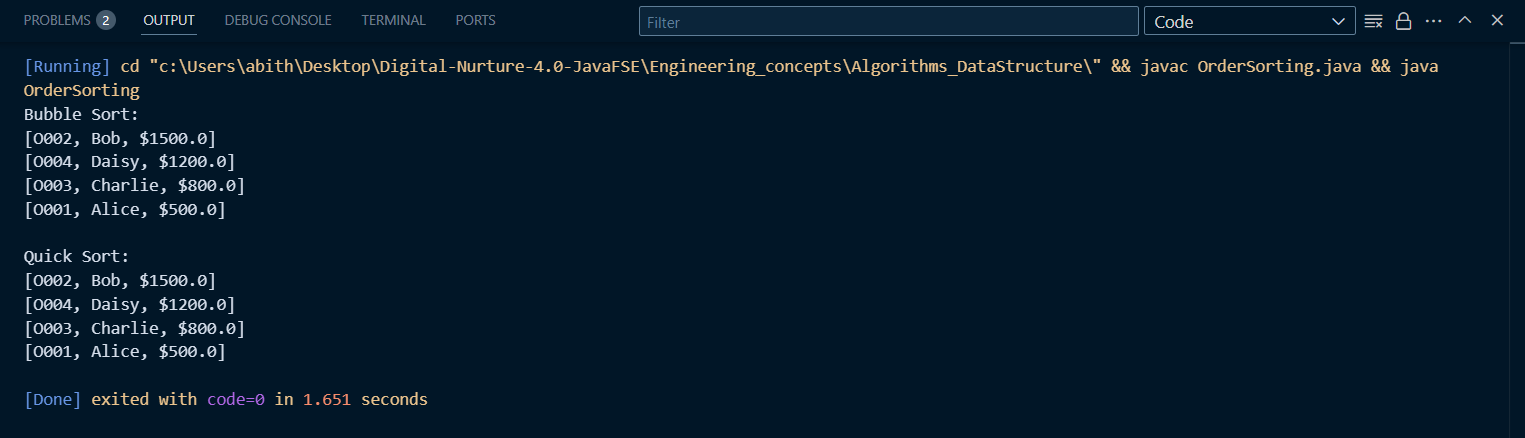
        quickSort(orders, 0, orders.length - 1);

        printOrders(orders);

    }

}

OUTPUT:



**Exercise 4: Employee Management System**

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 String toString() {

        return "[" + employeeId + ", " + name + ", " + position + ", $" + salary + "]";

    }

}

public class EmployeeManagement {

    static final int MAX = 100;

    Employee[] employees = new Employee[MAX];

    int count = 0;

    public void addEmployee(Employee e) {

        if (count < MAX) employees[count++] = e;

        else System.out.println("Employee limit reached.");

    }

    public Employee searchEmployee(String id) {

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

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

                return employees[i];

        }

        return null;

    }

    public void deleteEmployee(String id) {

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

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

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

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

                employees[--count] = null;

                return;

            }

        }

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

    }

    public void traverse() {

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

            System.out.println(employees[i]);

        }

    }

    public static void main(String[] args) {

        EmployeeManagement em = new EmployeeManagement();

        em.addEmployee(new Employee("E001", "John", "Manager", 5000));

        em.addEmployee(new Employee("E002", "Jane", "Developer", 4000));

        em.traverse();

        System.out.println("\nSearching:");

        System.out.println(em.searchEmployee("E002"));

        System.out.println("\nDeleting E001:");

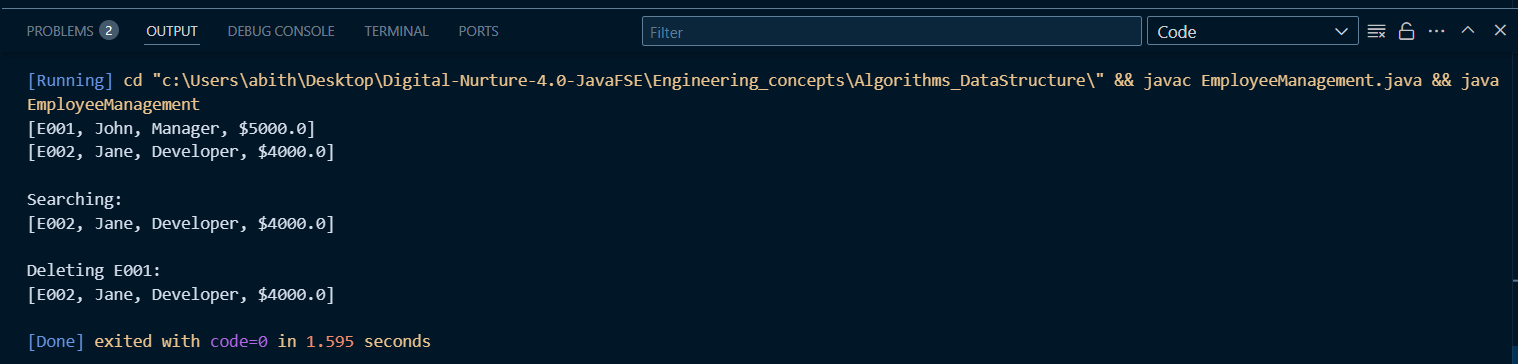
        em.deleteEmployee("E001");

        em.traverse();

    }

}

OUTPUT:



**Exercise 5: Task Management System**

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 String toString() {

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

    }

}

class Node {

    Task task;

    Node next;

    public Node(Task task) {

        this.task = task;

        this.next = null;

    }

}

public class TaskManagement {

    Node head = null;

    public void addTask(Task task) {

        Node newNode = new Node(task);

        newNode.next = head;

        head = newNode;

    }

    public Task searchTask(String id) {

        Node current = head;

        while (current != null) {

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

                return current.task;

            current = current.next;

        }

        return null;

    }

    public void deleteTask(String id) {

        Node current = head, prev = null;

        while (current != null) {

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

                if (prev == null) head = current.next;

                else prev.next = current.next;

                return;

            }

            prev = current;

            current = current.next;

        }

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

    }

    public void traverse() {

        Node current = head;

        while (current != null) {

            System.out.println(current.task);

            current = current.next;

        }

    }

    public static void main(String[] args) {

        TaskManagement tm = new TaskManagement();

        tm.addTask(new Task("T001", "Design UI", "Pending"));

        tm.addTask(new Task("T002", "Write Code", "In Progress"));

        tm.addTask(new Task("T003", "Test", "Completed"));

        tm.traverse();

        System.out.println("\nSearching for T002:");

        System.out.println(tm.searchTask("T002"));

        System.out.println("\nDeleting T002:");

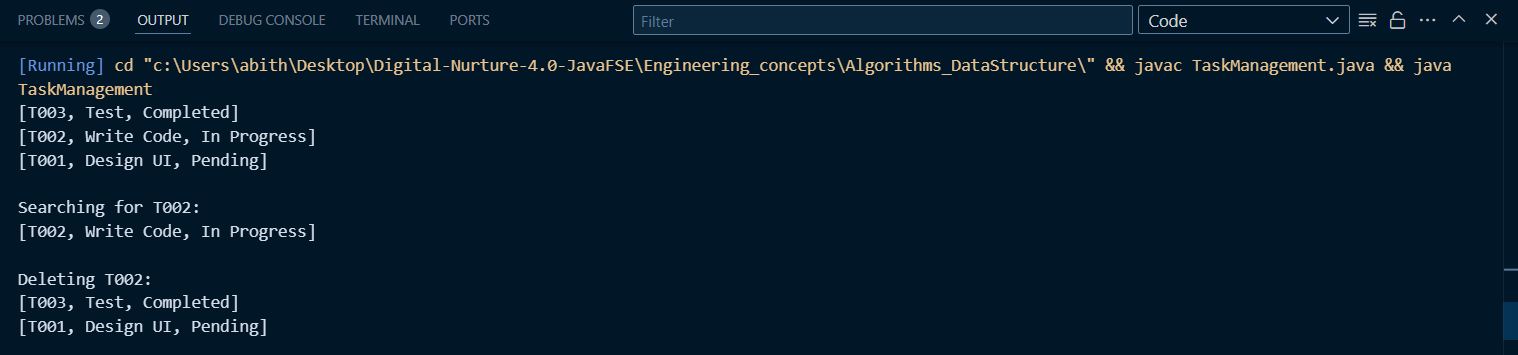
        tm.deleteTask("T002");

        tm.traverse();

    }

}

OUTPUT:



**Exercise 6: Library Management System**

import java.util.Arrays;

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 String toString() {

        return "[" + bookId + ", " + title + ", " + author + "]";

    }

}

public class LibrarySearch {

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

        for (int i = 0; i < books.length; i++) {

            if (books[i].title.equalsIgnoreCase(targetTitle)) {

                return i;

            }

        }

        return -1;

    }

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

        int left = 0, right = books.length - 1;

        while (left <= right) {

            int mid = left + (right - left) / 2;

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

            if (cmp == 0) return mid;

            if (cmp < 0) left = mid + 1;

            else right = mid - 1;

        }

        return -1;

    }

    public static void main(String[] args) {

        Book[] books = {

            new Book("B001", "Java Programming", "John Doe"),

            new Book("B002", "Data Structures", "Jane Smith"),

            new Book("B003", "Algorithms", "Alice Johnson")

        };

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

        int idx1 = linearSearch(books, "Algorithms");

        System.out.println(idx1 >= 0 ? books[idx1] : "Book not found");

        Arrays.sort(books, (a, b) -> a.title.compareToIgnoreCase(b.title));

        System.out.println("\nBinary Search:");

        int idx2 = binarySearch(books, "Algorithms");

        System.out.println(idx2 >= 0 ? books[idx2] : "Book not found");

    }

}

OUTPUT:

