**Algorithms & Data Structures**

**(CODES WITH OUTPUT)**

**Exercise 1: Inventory Management System**

**Product.java**

public 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 and setters

    public String getProductId() {

        return productId;

    }

    public String getProductName() {

        return productName;

    }

    public int getQuantity() {

        return quantity;}

    public double getPrice() {

        return price;

    }

    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

                + "]";

    }

}

**InventoryManager.java**

import java.util.HashMap;

public class InventoryManager {

    private HashMap<String, Product> inventory;

    public InventoryManager() {

        inventory = new HashMap<>();

    }

    // Add product

    public void addProduct(Product product) {

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

        System.out.println("Product added: " + product);

    }

    // Update product

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

        Product product = inventory.get(productId);

        if (product != null) {

            product.setQuantity(quantity);

            product.setPrice(price);

            System.out.println("Product updated: " + product);

        } else {

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

        }

    }

    // Delete product

    public void deleteProduct(String productId) {

        Product removed = inventory.remove(productId);

        if (removed != null) {

            System.out.println("Product deleted: " + removed);

        } else {

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

        }

    }

    // Show all products

    public void displayInventory() {

        System.out.println("\nCurrent Inventory:");

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

            System.out.println(product);

        }

    }

}

**Main.java**

public class Main {

    public static void main(String[] args) {

        InventoryManager manager = new InventoryManager();

        Product p1 = new Product("P001", "Keyboard", 10, 799.0);

        Product p2 = new Product("P002", "Mouse", 20, 499.0);

        manager.addProduct(p1);

        manager.addProduct(p2);

        manager.displayInventory();

        manager.updateProduct("P001", 15, 749.0);

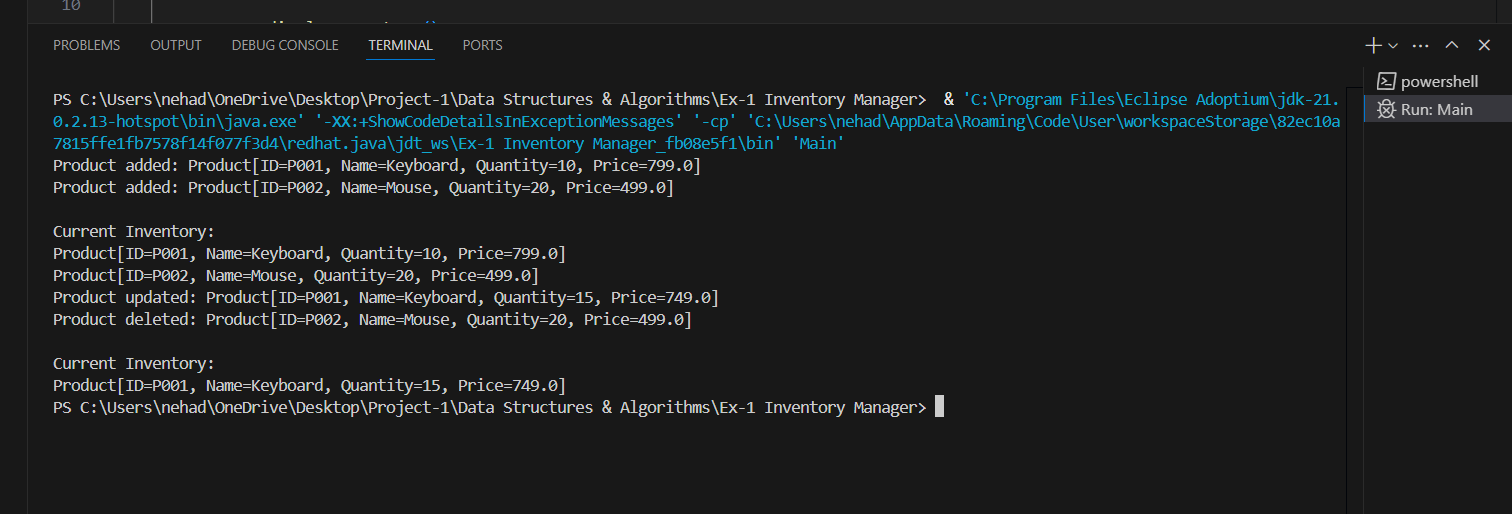
        manager.deleteProduct("P002");

        manager.displayInventory();

    }

}

**Output**:



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

**Product.java**

public class Product {

    int productId;

    String productName;

    String category;

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

        this.productId = productId;

        this.productName = productName;

        this.category = category;

    }

    public String toString() {

        return productId + ": " + productName + " (" + category + ")";

    }

}

**LinearSearch.java**

public class LinearSearch {

    public static Product search(Product[] products, String name) {

        for (Product p : products) {

            if (p.productName.equalsIgnoreCase(name)) {

                return p;

            }

        }

        return null;

    }

}

**BinarySearch.java**

import java.util.Arrays;

import java.util.Comparator;

public class BinarySearch {

    public static Product search(Product[] products, String name) {

        Arrays.sort(products, Comparator.comparing(p -> p.productName.toLowerCase()));

        int low = 0, high = products.length - 1;

        while (low <= high) {

            int mid = (low + high) / 2;

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

            if (cmp == 0)

                return products[mid];

            if (cmp < 0)

                low = mid + 1;

            else

                high = mid - 1;

        }

        return null;

    }

}

**Main.java**

public class Main {

    public static void main(String[] args) {

        Product[] products = {

            new Product(1, "Shoes", "Footwear"),

            new Product(2, "Laptop", "Electronics"),

            new Product(3, "T-shirt", "Apparel"),

            new Product(4, "Smartphone", "Electronics")

        };

        System.out.println("Linear Search for 'Laptop':");

        Product p1 = LinearSearch.search(products, "Laptop");

        System.out.println(p1);

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

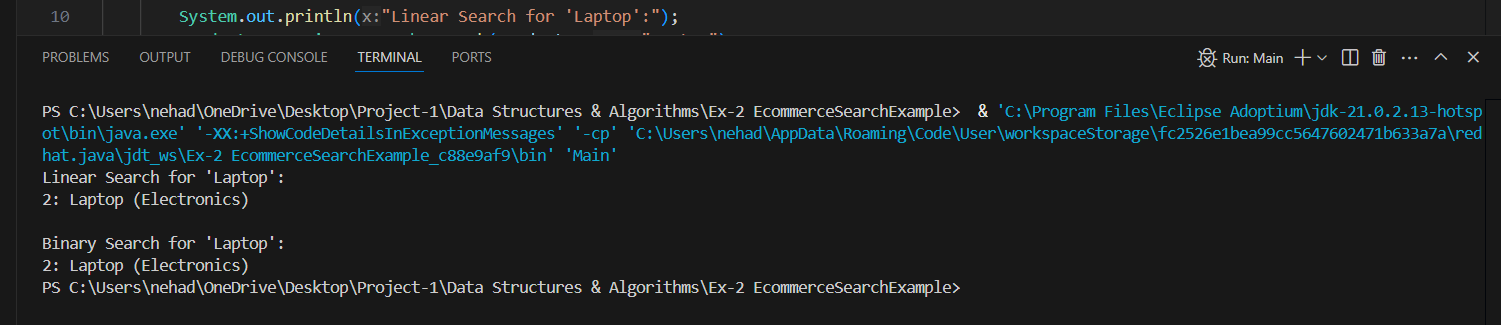
        Product p2 = BinarySearch.search(products, "Laptop");

        System.out.println(p2);

    }

}

**Output:**



**Exercise 3: Sorting Customer Orders**

**OrderSorter.java**

public class OrderSorter {

    public static void bubbleSort(Order[] orders) {

        int n = orders.length;

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

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

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

                    Order temp = orders[j];

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

                    orders[j + 1] = temp;

                }

            }

        }

    }

    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].getTotalPrice();

        int i = low - 1;

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

            if (orders[j].getTotalPrice() < 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 order : orders) {

            System.out.println(order);

        }

    }

}

**Order,java**

public class Order {

    private String orderId;

    private String customerName;

    private double totalPrice;

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

        this.orderId = orderId;

        this.customerName = customerName;

        this.totalPrice = totalPrice;

    }

    public String getOrderId() { return orderId; }

    public String getCustomerName() { return customerName; }

    public double getTotalPrice() { return totalPrice; }

    @Override

    public String toString() {

        return "Order[ID=" + orderId + ", Customer=" + customerName + ", Total=₹" + totalPrice + "]";

    }

}

**Main.java**

public class Main {

    public static void main(String[] args) {

        Order[] orders = {

                new Order("O001", "Alice", 3200.0),

                new Order("O002", "Bob", 1500.0),

                new Order("O003", "Charlie", 5400.0),

                new Order("O004", "David", 2500.0)

        };

        System.out.println("Original Orders:");

        OrderSorter.printOrders(orders);

        // Bubble Sort

        Order[] bubbleSorted = orders.clone();

        OrderSorter.bubbleSort(bubbleSorted);

        System.out.println("\nOrders Sorted by Bubble Sort:");

        OrderSorter.printOrders(bubbleSorted);

        // Quick Sort

        Order[] quickSorted = orders.clone();

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

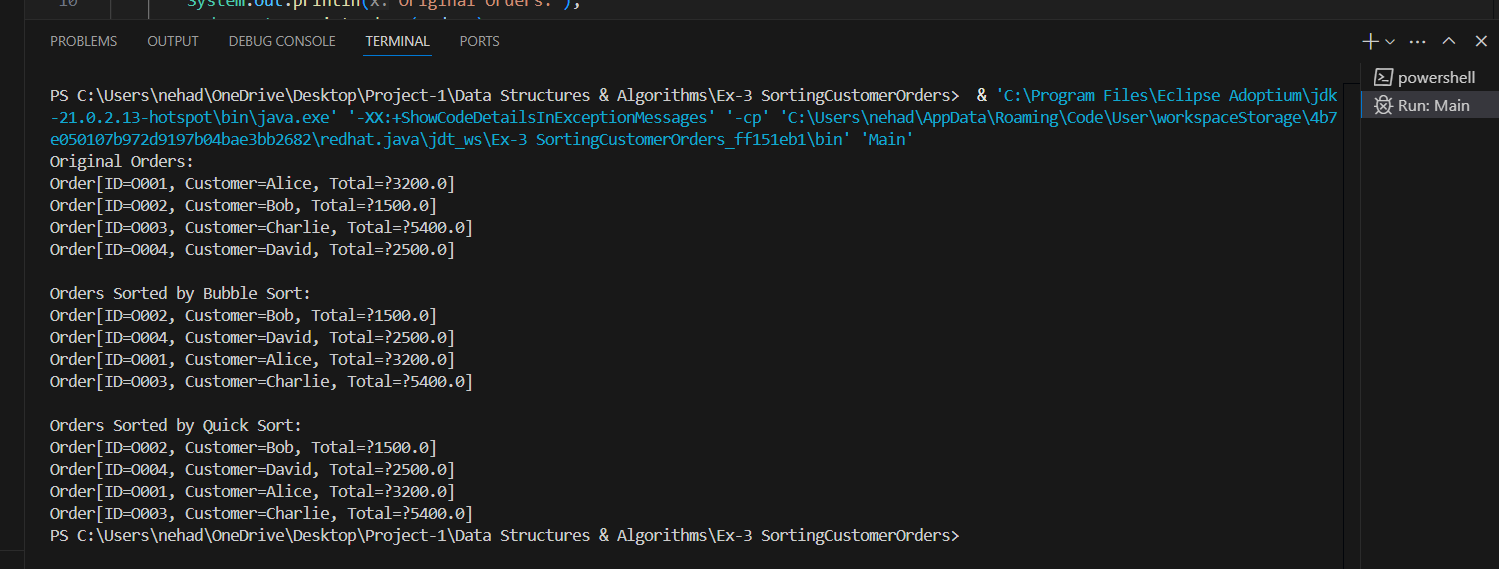
        System.out.println("\nOrders Sorted by Quick Sort:");

        OrderSorter.printOrders(quickSorted);

    }

}

**Output:**



**Exercise 4: Employee Management System**

**Employee.java**

public class Employee {

private String employeeId;

private String name;

private String position;

private 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 getEmployeeId() {

return employeeId;

}

public String toString() {

return "Employee[ID=" + employeeId + ", Name=" + name + ", Position=" + position + ", Salary=Rs." + salary + "]";

}

}

**EmployeeManager.java**

public class EmployeeManager {

    private Employee[] employees;

    private int count;

    public EmployeeManager(int size) {

        employees = new Employee[size];

        count = 0;

    }

    public void addEmployee(Employee emp) {

        if (count < employees.length) {

            employees[count++] = emp;

            System.out.println("Added: " + emp);

        } else {

            System.out.println("Employee array is full!");

        }

    }

    public void searchEmployee(String id) {

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

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

                System.out.println("Found: " + employees[i]);

                return;

            }

        }

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

    }

    public void displayAllEmployees() {

        System.out.println("\nAll Employees:");

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

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

        }

    }

    public void deleteEmployee(String id) {

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

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

                System.out.println("Deleted: " + employees[i]);

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

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

                }

                employees[--count] = null;

                return;

            }

        }

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

    }

}

**Main.java**

public class Main {

    public static void main(String[] args) {

        EmployeeManager manager = new EmployeeManager(5);

        manager.addEmployee(new Employee("E001", "Neha", "Developer", 65000));

        manager.addEmployee(new Employee("E002", "Ravi", "Manager", 75000));

        manager.addEmployee(new Employee("E003", "Kiran", "Designer", 55000));

        manager.displayAllEmployees();

        manager.searchEmployee("E002");

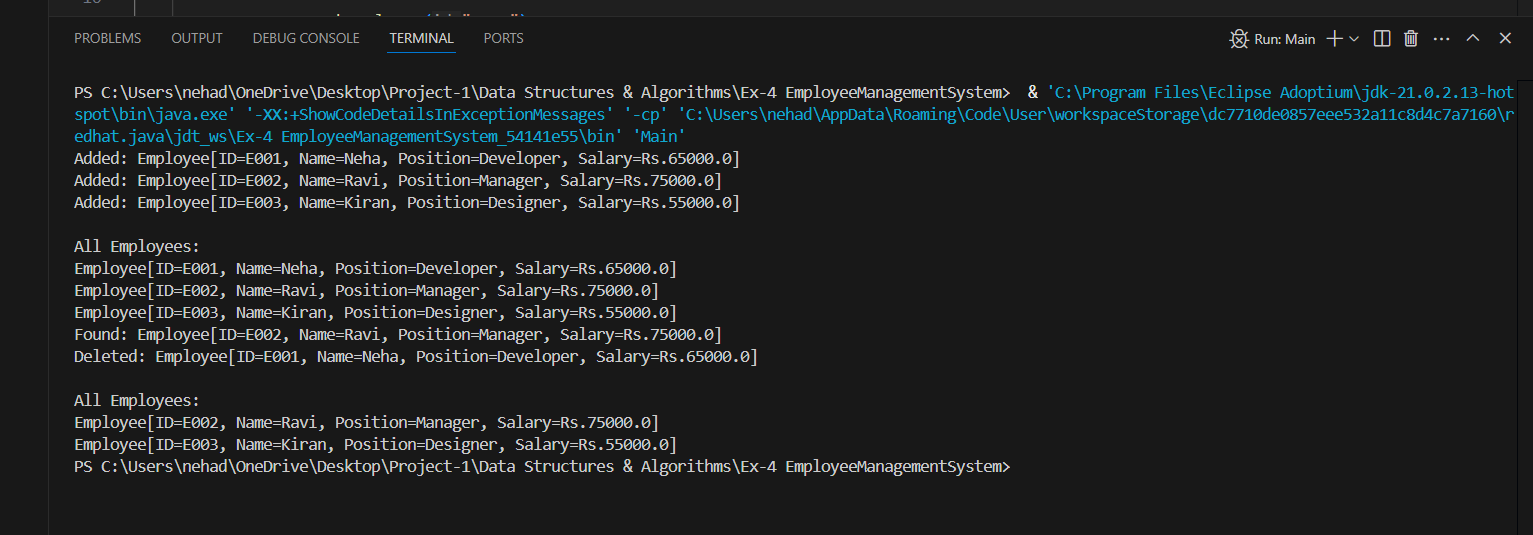
        manager.deleteEmployee("E001");

        manager.displayAllEmployees();

    }

}

**Output:**



**Exercise 5: Task Management System**

**TaskManager.java**

public class TaskManager {

    private Task head;

    // Add task to the end of the list

    public void addTask(String taskId, String taskName, String status) {

        Task newTask = new Task(taskId, taskName, status);

        if (head == null) {

            head = newTask;

        } else {

            Task current = head;

            while (current.next != null) {

                current = current.next;

            }

            current.next = newTask;

        }

        System.out.println("Added: " + newTask);

    }

    // Search task by ID

    public void searchTask(String taskId) {

        Task current = head;

        while (current != null) {

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

                System.out.println("Found: " + current);

                return;

            }

            current = current.next;

        }

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

    }

    // Traverse tasks

    public void displayTasks() {

        System.out.println("\nAll Tasks:");

        Task current = head;

        while (current != null) {

            System.out.println(current);

            current = current.next;

        }

    }

    // Delete task by ID

    public void deleteTask(String taskId) {

        if (head == null)

            return;

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

            System.out.println("Deleted: " + head);

            head = head.next;

            return;

        }

        Task current = head;

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

            current = current.next;

        }

        if (current.next != null) {

            System.out.println("Deleted: " + current.next);

            current.next = current.next.next;

        } else {

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

        }

    }

}

**Task.java**

public class Task {

    String taskId;

    String taskName;

    String status;

    Task next;

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

        this.taskId = taskId;

        this.taskName = taskName;

        this.status = status;

        this.next = null;

    }

    public String toString() {

        return "Task[ID=" + taskId + ", Name=" + taskName + ", Status=" + status + "]";

    }

}

**Main.java**

public class Main {

    public static void main(String[] args) {

        TaskManager manager = new TaskManager();

        manager.addTask("T001", "Design UI", "Pending");

        manager.addTask("T002", "Develop backend", "In Progress");

        manager.addTask("T003", "Testing", "Pending");

        manager.displayTasks();

        manager.searchTask("T002");

        manager.deleteTask("T001");

        manager.displayTasks();

    }

}

**Output:**

A screenshot of a computer program

AI-generated content may be incorrect.

**Exercise 6: Library Management System**

**LibraryManager.java**

import java.util.Arrays;

import java.util.Comparator;

public class LibraryManager {

    private Book[] books;

    private int count;

    public LibraryManager(int size) {

        books = new Book[size];

        count = 0;

    }

    public void addBook(Book book) {

        if (count < books.length) {

            books[count++] = book;

            System.out.println("Added: " + book);

        } else {

            System.out.println("Library is full!");

        }

    }

    public void linearSearch(String title) {

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

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

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

                System.out.println("Found: " + books[i]);

                return;

            }

        }

        System.out.println("Book not found!");

    }

    public void binarySearch(String title) {

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

        Arrays.sort(books, 0, count, Comparator.comparing(Book::getTitle));

        int left = 0, right = count - 1;

        while (left <= right) {

            int mid = (left + right) / 2;

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

            if (cmp == 0) {

                System.out.println("Found: " + books[mid]);

                return;

            } else if (cmp < 0) {

                left = mid + 1;

            } else {

                right = mid - 1;

            }

        }

        System.out.println("Book not found!");

    }

    public void displayBooks() {

        System.out.println("\nAll Books:");

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

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

        }

    }

}

**Book.java**

public 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 getTitle() {

        return title;

    }

    public String toString() {

        return "Book[ID=" + bookId + ", Title=" + title + ", Author=" + author + "]";

    }

}

**Main.java**

public class Main {

    public static void main(String[] args) {

        LibraryManager library = new LibraryManager(10);

        library.addBook(new Book("B001", "Java Programming", "Neha Dasari"));

        library.addBook(new Book("B002", "Data Structures", "Ravi Teja"));

        library.addBook(new Book("B003", "Operating Systems", "Kiran Kumar"));

        library.addBook(new Book("B004", "Design Patterns", "Sneha Roy"));

        library.displayBooks();

        library.linearSearch("Operating Systems");

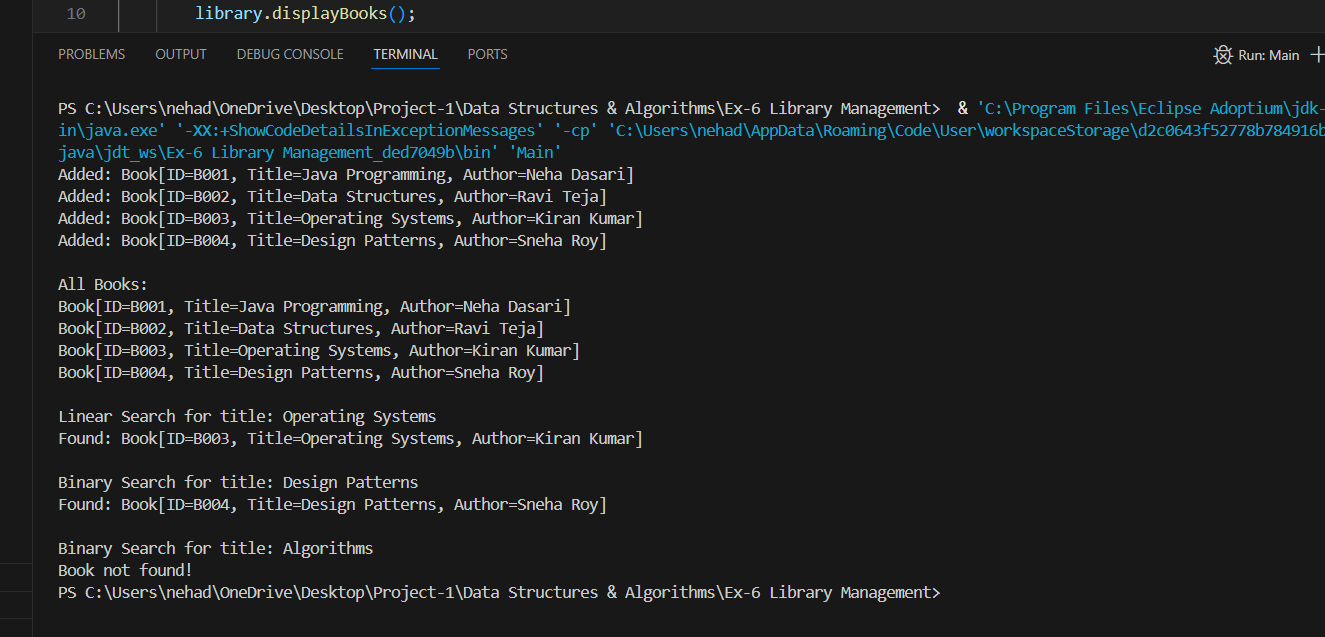
        library.binarySearch("Design Patterns");

        library.binarySearch("Algorithms");

    }

}

**OUTPUT:**



**Exercise 7: Financial Forecasting**

**FinancialForecast.java**

public class FinancialForecast {

    public static double forecastValue(double initialValue, double growthRate, int years) {

        if (years == 0) {

            return initialValue;

        }

        return forecastValue(initialValue, growthRate, years - 1) \* (1 + growthRate);

    }

    public static double forecastIterative(double initialValue, double growthRate, int years) {

        double result = initialValue;

        for (int i = 1; i <= years; i++) {

            result \*= (1 + growthRate);

        }

        return result;

    }

}

**Main.java**

public class Main {

    public static void main(String[] args) {

        double initial = 1000.0;

        double rate = 0.10;

        int years = 5;

        double futureRecursive = FinancialForecast.forecastValue(initial, rate, years);

        double futureIterative = FinancialForecast.forecastIterative(initial, rate, years);

        System.out.printf("Recursive Future Value after %d years: Rs.%.2f\n", years, futureRecursive);

        System.out.printf("Iterative Future Value after %d years: Rs.%.2f\n", years, futureIterative);

    }

}

**Output:**

