**Exercise 1: Inventory Management System**

**code:**

1. Product.java:

public class Product {

private int productId;

private String productName;

private int quantity;

private double price;

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

this.productId = productId;

this.productName = productName;

this.quantity = quantity;

this.price = price;

}

public int 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 "ID: " + productId + ", Name: " + productName +

", Quantity: " + quantity + ", Price: Rs." + price;

}

}

2.InventoryManager.java:

import java.util.HashMap;

public class InventoryManager {

private HashMap<Integer, Product> inventory = new HashMap<>();

public void addProduct(Product product) {

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

System.out.println("Product added successfully!");

}

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

Product p = inventory.get(productId);

if (p != null) {

p.setProductName(name);

p.setQuantity(qty);

p.setPrice(price);

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

} else {

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

}

}

public void deleteProduct(int productId) {

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

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

} else {

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

}

}

public void displayInventory() {

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

System.out.println(p);

}

}

public static void main(String[] args) {

InventoryManager manager = new InventoryManager();

// Sample data

manager.addProduct(new Product(101, "Laptop", 5, 60000));

manager.addProduct(new Product(102, "Mouse", 15, 500));

manager.displayInventory();

manager.updateProduct(101, "Gaming Laptop", 3, 75000);

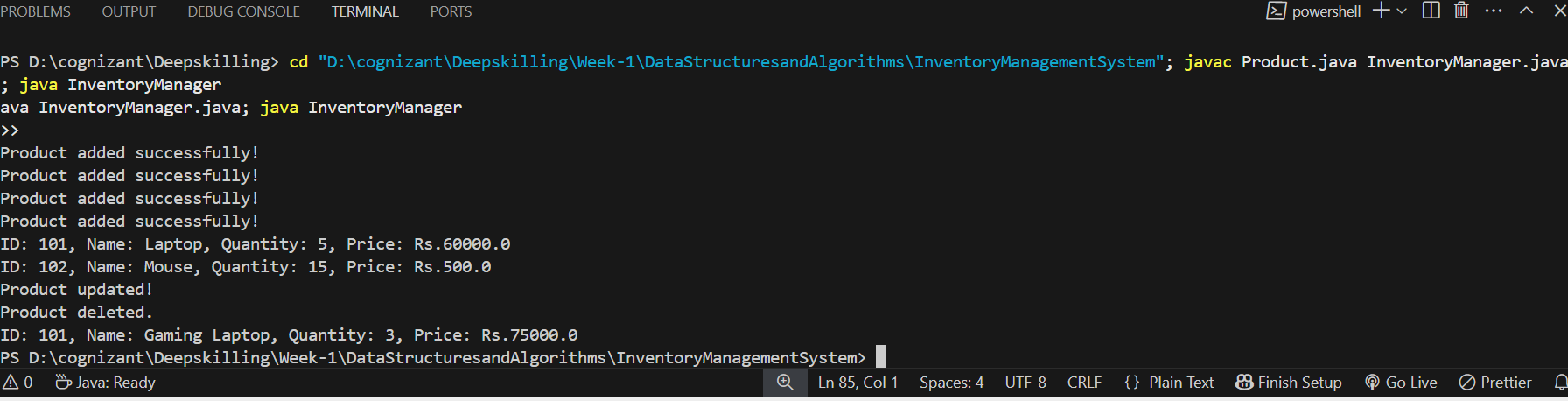
manager.deleteProduct(102);

manager.displayInventory();

}

}

OUTPUT:



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

CODE:

1.Product.java:

public class Product {

private int productId;

    private String productName;

    private String category;

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

        this.productId = productId;

        this.productName = productName;

        this.category = category;

    }

public int getProductId() {

        return productId;

    }

public String getProductName() {

        return productName;

    }

public String getCategory() {

        return category;

    }

    @Override

    public String toString() {

        return "ID: " + productId + ", Name: " + productName + ", Category: " + category;

    }

}

2. EcommerceSearch.java:

import java.util.Arrays;

import java.util.Comparator;

import java.util.Scanner;

public class ECommerceSearch {

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

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

        while (left <= right) {

            int mid = (left + right) / 2;

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

            if (cmp == 0)

                return products[mid];

            else if (cmp < 0)

                left = mid + 1;

            else

                right = mid - 1;

        }

        return null;

    }

    public static void main(String[] args) {

        Product[] products = {

            new Product(101, "Laptop", "Electronics"),

            new Product(102, "Mouse", "Accessories"),

            new Product(103, "Shoes", "Footwear"),

            new Product(104, "Bag", "Bags"),

            new Product(105, "Keyboard", "Accessories")

        };

        Arrays.sort(products, Comparator.comparing(Product::getProductName, String.CASE\_INSENSITIVE\_ORDER));

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter product name to search: ");

        String searchName = sc.nextLine();

        Product result = binarySearch(products, searchName);

        if (result != null)

            System.out.println("Product found: " + result);

        else

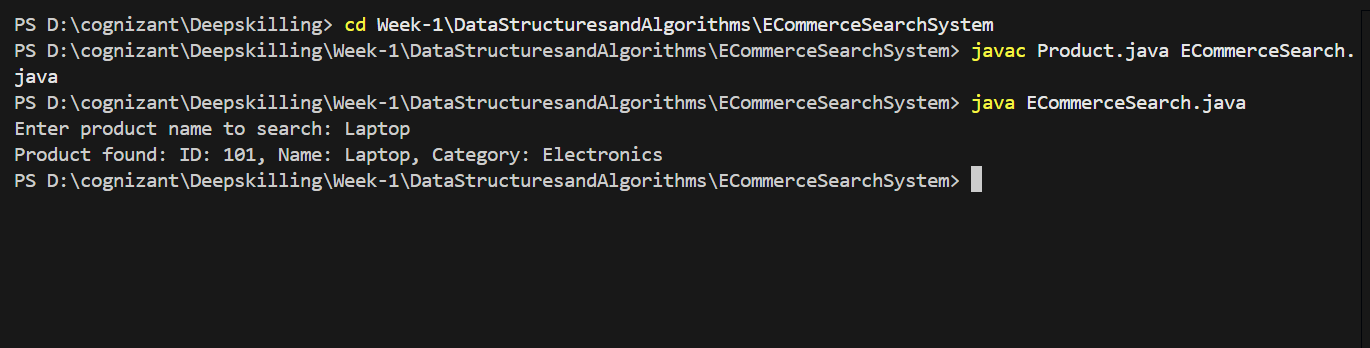
            System.out.println("Product not found.");

        sc.close();

    }

}

OUTPUT :



**Exercise 3: Sorting Customer Orders**

CODE :

1.Order.java:

public class Order {

    private int orderId;

    private String customerName;

    private double totalPrice;

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

        this.orderId = orderId;

        this.customerName = customerName;

        this.totalPrice = totalPrice;

    }

    public int getOrderId() { return orderId; }

    public String getCustomerName() { return customerName; }

    public double getTotalPrice() { return totalPrice; }

    @Override

    public String toString() {

        return "OrderID: " + orderId + ", Customer: " + customerName + ", Total Price: " + totalPrice;

    }

}

**2. OrderSorter.java:**

public class OrderSorter {

    // Bubble Sort

    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()) {

                    // Swap

                    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].getTotalPrice();

        int i = low - 1;

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

            if (orders[j].getTotalPrice() > pivot) { // Descending order

                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 displayOrders(Order[] orders) {

        for (Order order : orders) {

            System.out.println(order);

        }

    }

    public static void main(String[] args) {

        Order[] orders = {

            new Order(1, "Ammu", 4500),

            new Order(2, "Raj", 2200),

            new Order(3, "Meera", 10000),

            new Order(4, "Kiran", 7800),

            new Order(5, "Anu", 3100)

        };

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

        displayOrders(orders);

        System.out.println("\nSorted using Bubble Sort (High to Low):");

        bubbleSort(orders);

        displayOrders(orders);

        // Reset orders

        orders = new Order[] {

            new Order(1, "Ammu", 4500),

            new Order(2, "Raj", 2200),

            new Order(3, "Meera", 10000),

            new Order(4, "Kiran", 7800),

            new Order(5, "Anu", 3100)

        };

        System.out.println("\nSorted using Quick Sort (High to Low):");

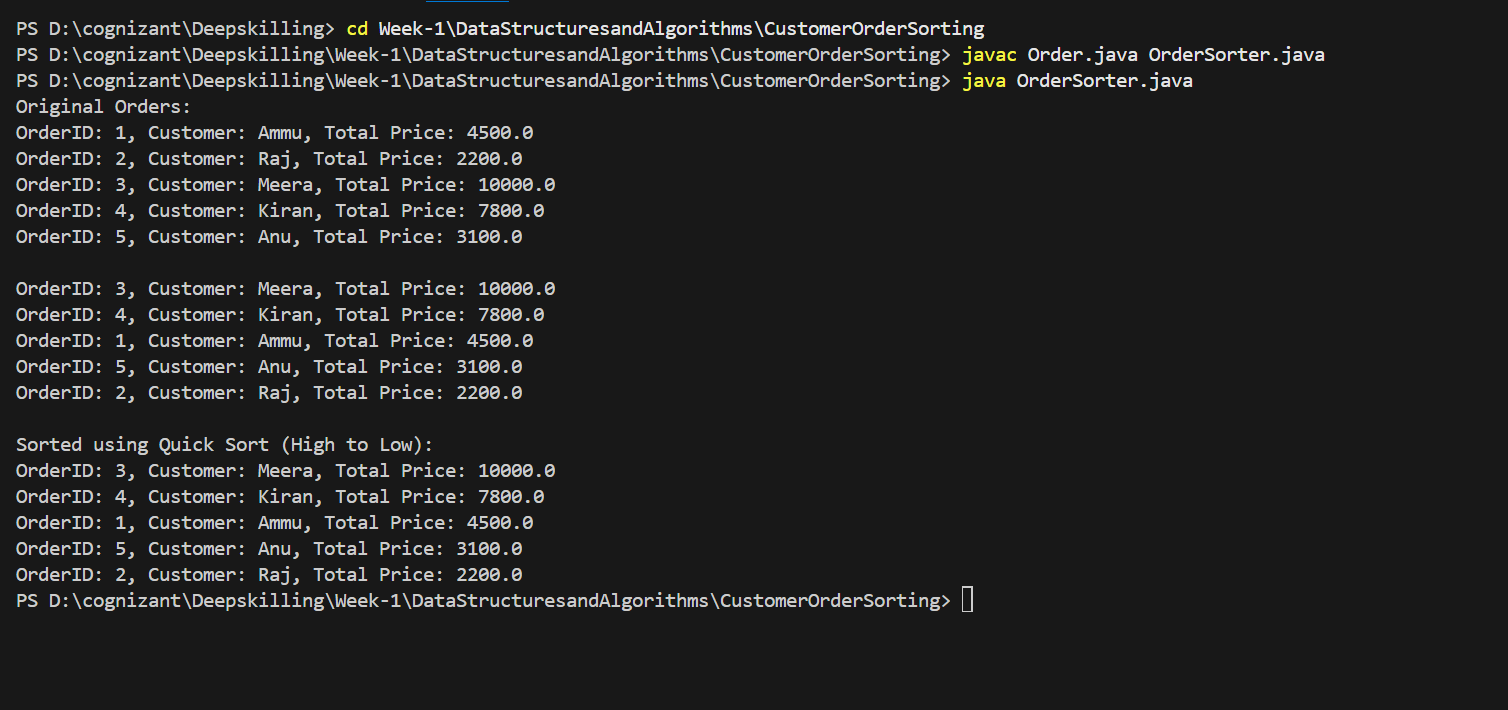
        quickSort(orders, 0, orders.length - 1);

        displayOrders(orders);

    }

}

**OUTPUT :**



**Exercise 4: Employee Management System**

**CODE:**

1. **Employee.java:**

public class Employee {

private int employeeId;

private String name;

private String position;

private double salary;

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

this.employeeId = employeeId;

this.name = name;

this.position = position;

this.salary = salary;

}

public int getEmployeeId() { return employeeId; }

public String getName() { return name; }

public String getPosition() { return position; }

public double getSalary() { return salary; }

@Override

public String toString() {

return "ID: " + employeeId + ", Name: " + name + ", Position: " + position + ", Salary: " + salary;

}

}

1. **EmployeeManagementSystem.java :**

public class EmployeeManagementSystem {

private Employee[] employees = new Employee[10];

private int count = 0;

public void addEmployee(Employee emp) {

if (count < employees.length) {

employees[count++] = emp;

} else {

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

}

}

public void traverseEmployees() {

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

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

}

}

public void searchEmployee(int empId) {

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

if (employees[i].getEmployeeId() == empId) {

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

return;

}

}

System.out.println("Employee Not Found!");

}

public void deleteEmployee(int empId) {

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

if (employees[i].getEmployeeId() == empId) {

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

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

}

employees[--count] = null;

System.out.println("Employee Deleted.");

return;

}

}

System.out.println("Employee Not Found.");

}

public static void main(String[] args) {

EmployeeManagementSystem ems = new EmployeeManagementSystem();

ems.addEmployee(new Employee(1, "Ammu", "Developer", 50000));

ems.addEmployee(new Employee(2, "Kiran", "Tester", 40000));

ems.addEmployee(new Employee(3, "Raj", "Manager", 75000));

System.out.println("All Employees:");

ems.traverseEmployees();

System.out.println("\nSearching Employee with ID 2:");

ems.searchEmployee(2);

System.out.println("\nDeleting Employee with ID 2:");

ems.deleteEmployee(2);

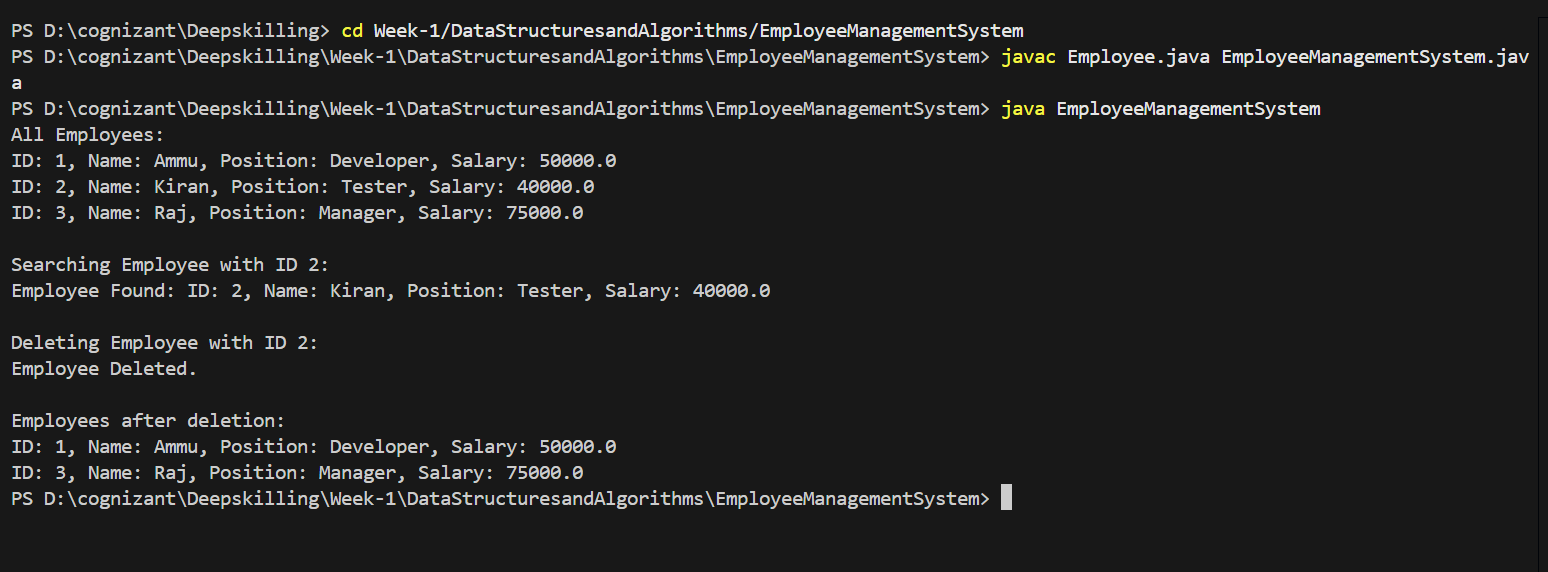
System.out.println("\nEmployees after deletion:");

ems.traverseEmployees();

}

}

**OUTPUT :**

****

**Exercise 5: Task Management System**

**CODE:**

**1. Task.java :**

public class Task {

int taskId;

String taskName;

String status;

public Task next;

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

this.taskId = taskId;

this.taskName = taskName;

this.status = status;

this.next = null;

}

@Override

public String toString() {

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

}

}

**2. TaskManager.java :**

public class TaskManager {

Task head = null;

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: " + newTask.taskName);

}

public void traverseTasks() {

if (head == null) {

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

return;

}

Task current = head;

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

while (current != null) {

System.out.println(current);

current = current.next;

}

}

public static void main(String[] args) {

TaskManager manager = new TaskManager();

manager.addTask(1, "Design UI", "Pending");

manager.addTask(2, "Develop API", "In Progress");

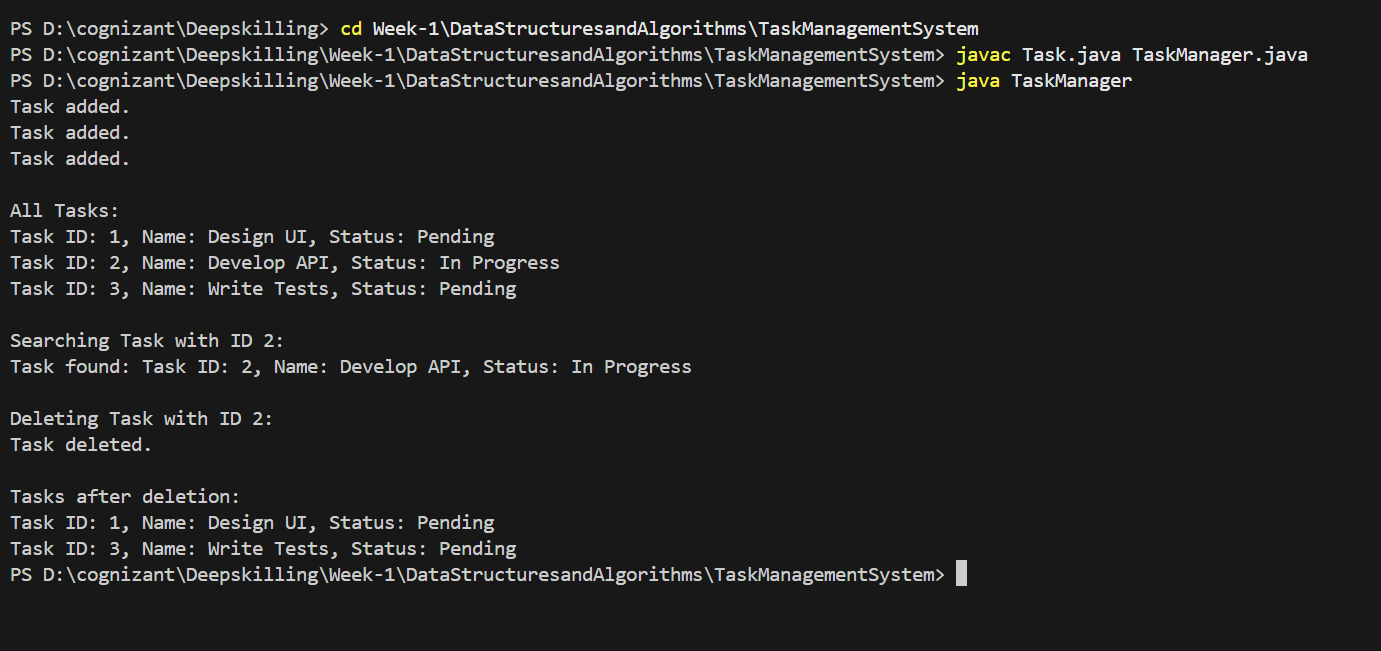
manager.addTask(3, "Write Tests", "Pending");

manager.traverseTasks();

}

}

OUTPUT :



**Exercise 6: Library Management System**

CODE :

**1.Book.java :**

public class Book {

int bookId;

String title;

String author;

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

this.bookId = bookId;

this.title = title;

this.author = author;

}

@Override

public String toString() {

return "Book ID: " + bookId + ", Title: " + title + ", Author: " + author;

}

}

**2. LibraryManager.java :**

import java.util.Arrays;

import java.util.Comparator;

public class LibraryManager {

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

for (Book book : books) {

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

return book;

}

}

return null;

}

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

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

while (low <= high) {

int mid = (low + high) / 2;

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

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

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

else high = mid - 1;

}

return null;

}

public static void main(String[] args) {

Book[] books = {

new Book(101, "The Alchemist", "Paulo Coelho"),

new Book(102, "To Kill a Mockingbird", "Harper Lee"),

new Book(103, "1984", "George Orwell"),

new Book(104, "Pride and Prejudice", "Jane Austen"),

new Book(105, "The Great Gatsby", "F. Scott Fitzgerald")

};

System.out.println("\n🔍 Linear Search Result:");

Book found = linearSearch(books, "1984");

System.out.println(found != null ? found : "Book not found.");

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

System.out.println("\n🔍 Binary Search Result:");

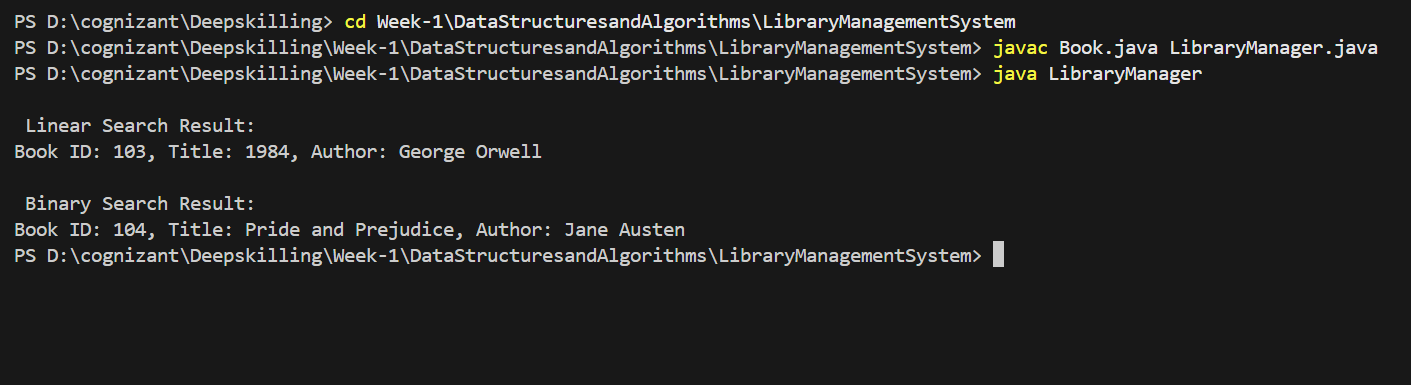
Book found2 = binarySearch(books, "Pride and Prejudice");

System.out.println(found2 != null ? found2 : "Book not found.");

}

}

**OUTPUT :**



**Exercise 7: Financial Forecasting**

**CODE :**

**1.Forecast.java :**

public class Forecast {

public static double predictFutureValue(double currentValue, double growthRate, int years) {

if (years == 0) {

return currentValue;

}

return predictFutureValue(currentValue \* (1 + growthRate), growthRate, years - 1);

}

public static void main(String[] args) {

double presentValue = 10000;

double growthRate = 0.1;

int years = 5;

double futureValue = predictFutureValue(presentValue, growthRate, years);

System.out.printf("Future value after %d years: ₹%.2f\n", years, futureValue);

}

}

**OUTPUT :**

