**Exercise 1: Inventory Management System**

**CODE:**

**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 void setQuantity(int quantity) {

        this.quantity = quantity;

    }

    public void setPrice(double price) {

        this.price = price;

    }

    public String toString() {

        return "[" + productId + "] " + productName + " - Qty: " + quantity + ", Price: $" + price;

    }

}

**Inventory.java**

import java.util.HashMap;

import java.util.Map;

public class Inventory {

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

    public void addProduct(Product product) {

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

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

        } else {

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

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

        }

    }

    public void updateProduct(int productId, int newQty, double newPrice) {

        Product p = products.get(productId);

        if (p != null) {

            p.setQuantity(newQty);

            p.setPrice(newPrice);

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

        } else {

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

        }

    }

    public void deleteProduct(int productId) {

        Product removed = products.remove(productId);

        if (removed != null) {

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

        } else {

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

        }

    }

    public void displayAll() {

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

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

            System.out.println(p);

        }

    }

}

**Main.java**

public class Main {

    public static void main(String[] args) {

        Inventory inventory = new Inventory();

        Product p1 = new Product(101, "Laptop", 10, 750.00);

        Product p2 = new Product(102, "Smartphone", 25, 300.00);

        Product p3 = new Product(103, "Monitor", 15, 150.00);

        inventory.addProduct(p1);

        inventory.addProduct(p2);

        inventory.addProduct(p3);

        inventory.displayAll();

        inventory.updateProduct(102, 30, 280.00);

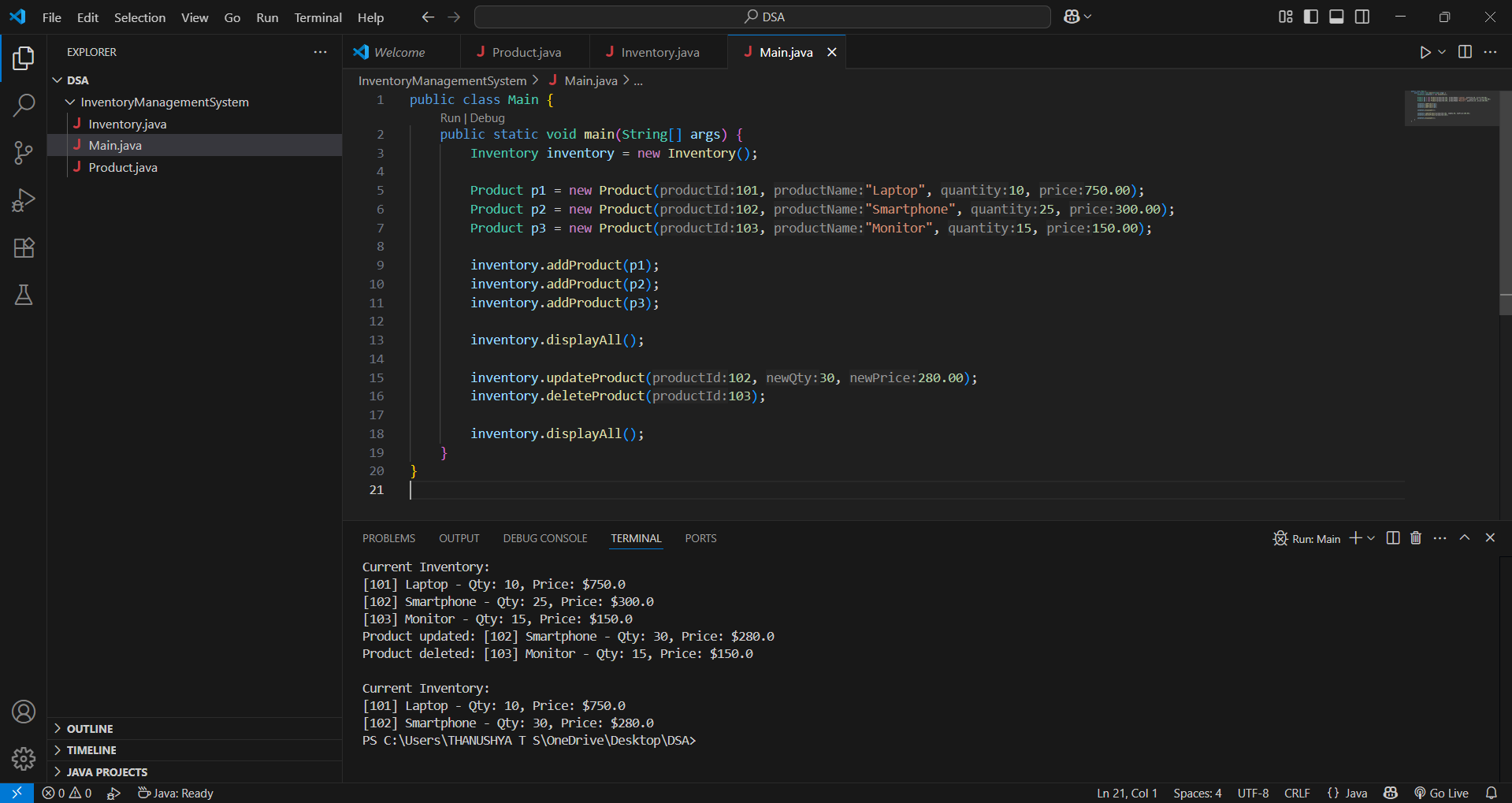
        inventory.deleteProduct(103);

        inventory.displayAll();

    }

}

**OUTPUT:**



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

**CODE:**

**Product.java**

package ECommerce;

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;

    }

    public String toString() {

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

    }

}

**SearchUtils.java**

package ECommerce;

public class SearchUtils {

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

        for (Product p : products) {

            if (p.getProductName().equalsIgnoreCase(targetName)) {

                return p;

            }

        }

        return null;

    }

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

        int left = 0;

        int right = products.length - 1;

        while (left <= right) {

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

            int compare = products[mid].getProductName().compareToIgnoreCase(targetName);

            if (compare == 0) return products[mid];

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

            else right = mid - 1;

        }

        return null;

    }

    public static void sortByName(Product[] products) {

        java.util.Arrays.sort(products, (a, b) -> a.getProductName().compareToIgnoreCase(b.getProductName()));

    }

}

**Main.java**

package ECommerce;

public class Main {

    public static void main(String[] args) {

        Product[] products = {

            new Product(1, "Laptop", "Electronics"),

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

            new Product(3, "Notebook", "Stationery"),

            new Product(4, "Pen", "Stationery"),

            new Product(5, "Smartphone", "Electronics")

        };

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

        Product foundLinear = SearchUtils.linearSearch(products, "Pen");

        System.out.println(foundLinear != null ? foundLinear : "Not found");

        SearchUtils.sortByName(products);

        System.out.println("\n Binary Search for 'Pen':");

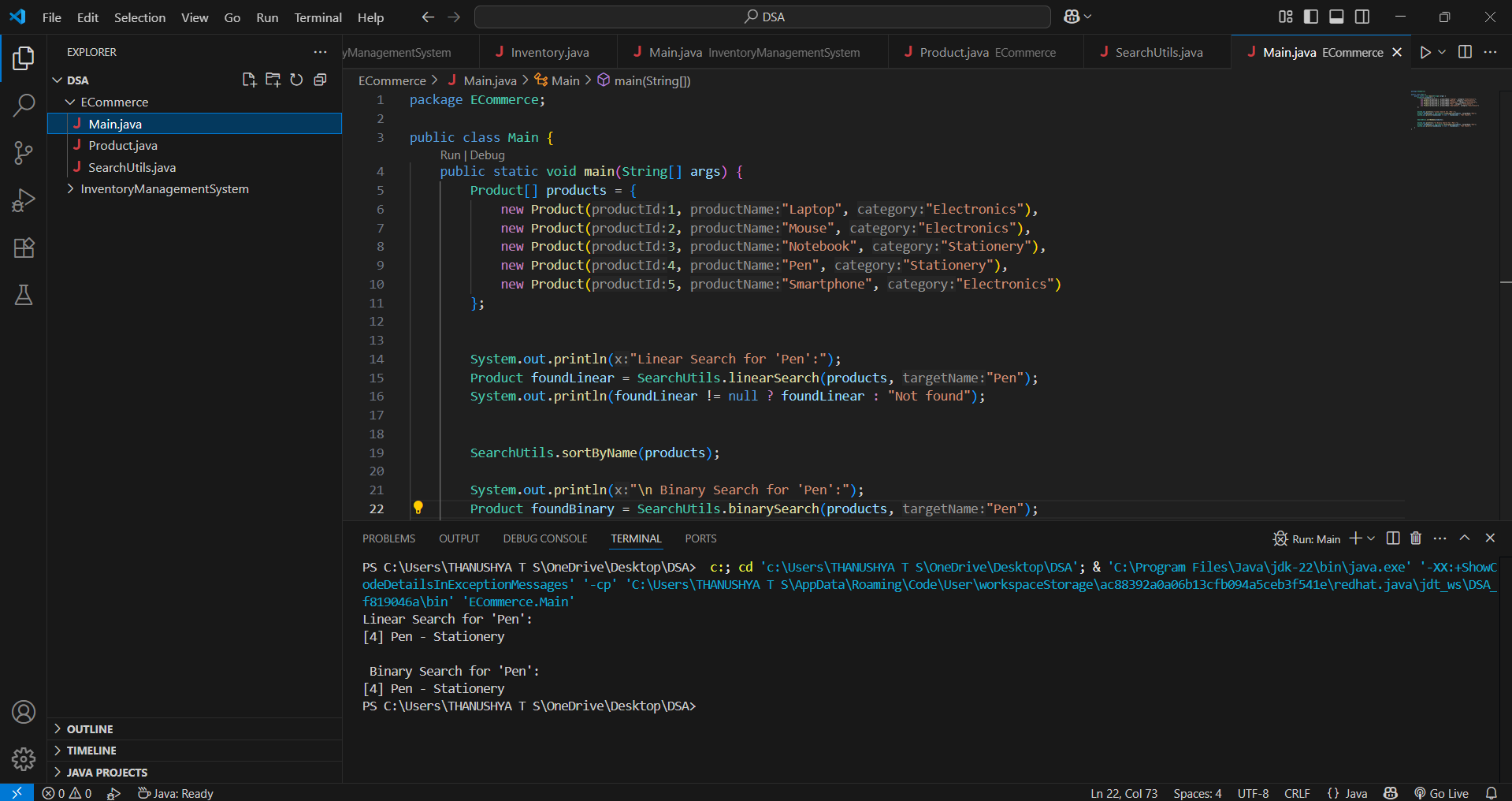
        Product foundBinary = SearchUtils.binarySearch(products, "Pen");

        System.out.println(foundBinary != null ? foundBinary : "Not found");

    }

}

**OUTPUT:**

****

**Exercise 4: Employee Management System**

**CODE:**

**Employee.java**

package EmployeeManagementSystem;

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 toString() {

        return "[" + employeeId + "] " + name + " - " + position + " " + salary;

    }

}

**EmployeeManager.java**

package EmployeeManagementSystem;

public class EmployeeManager {

    private Employee[] employees;

    private int size;

    public EmployeeManager(int capacity) {

        employees = new Employee[capacity];

        size = 0;

    }

    public void addEmployee(Employee e) {

        if (size >= employees.length) {

            System.out.println("Employee list full. Cannot add more.");

            return;

        }

        employees[size++] = e;

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

    }

    public Employee searchEmployee(int empId) {

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

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

                return employees[i];

            }

        }

        return null;

    }

    public void deleteEmployee(int empId) {

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

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

                // Shift remaining elements

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

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

                }

                employees[--size] = null;

                System.out.println("Deleted employee with ID: " + empId);

                return;

            }

        }

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

    }

    public void displayAll() {

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

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

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

        }

    }

}

**Main.java**

package EmployeeManagementSystem;

public class Main {

    public static void main(String[] args) {

        EmployeeManager manager = new EmployeeManager(5);

        manager.addEmployee(new Employee(101, "Alice", "Developer", 60000));

        manager.addEmployee(new Employee(102, "Bob", "Designer", 50000));

        manager.addEmployee(new Employee(103, "Charlie", "Manager", 80000));

        manager.displayAll();

        System.out.println("\n Searching for Employee ID 102:");

        Employee found = manager.searchEmployee(102);

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

        System.out.println("\n Deleting Employee ID 102");

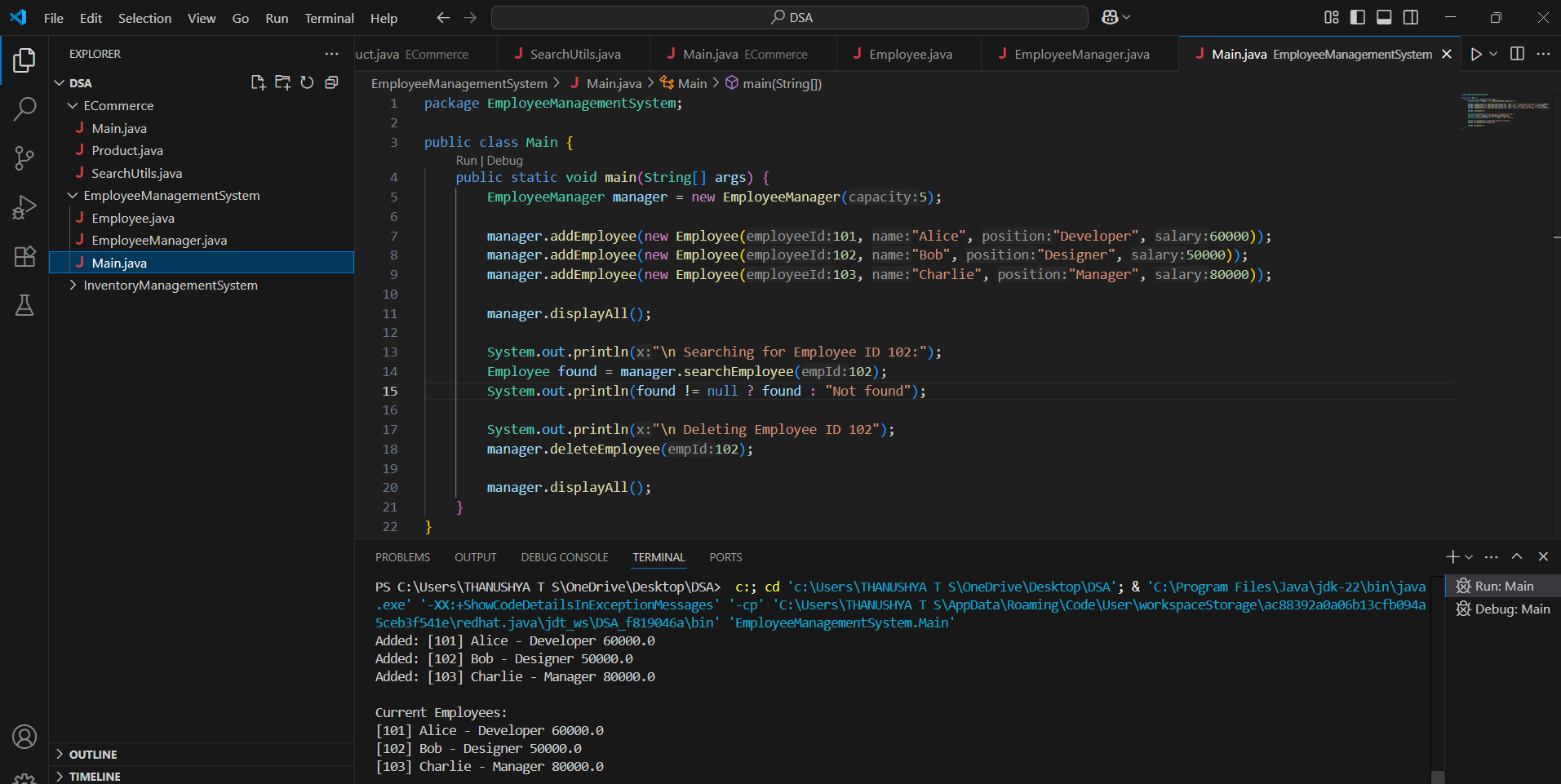
        manager.deleteEmployee(102);

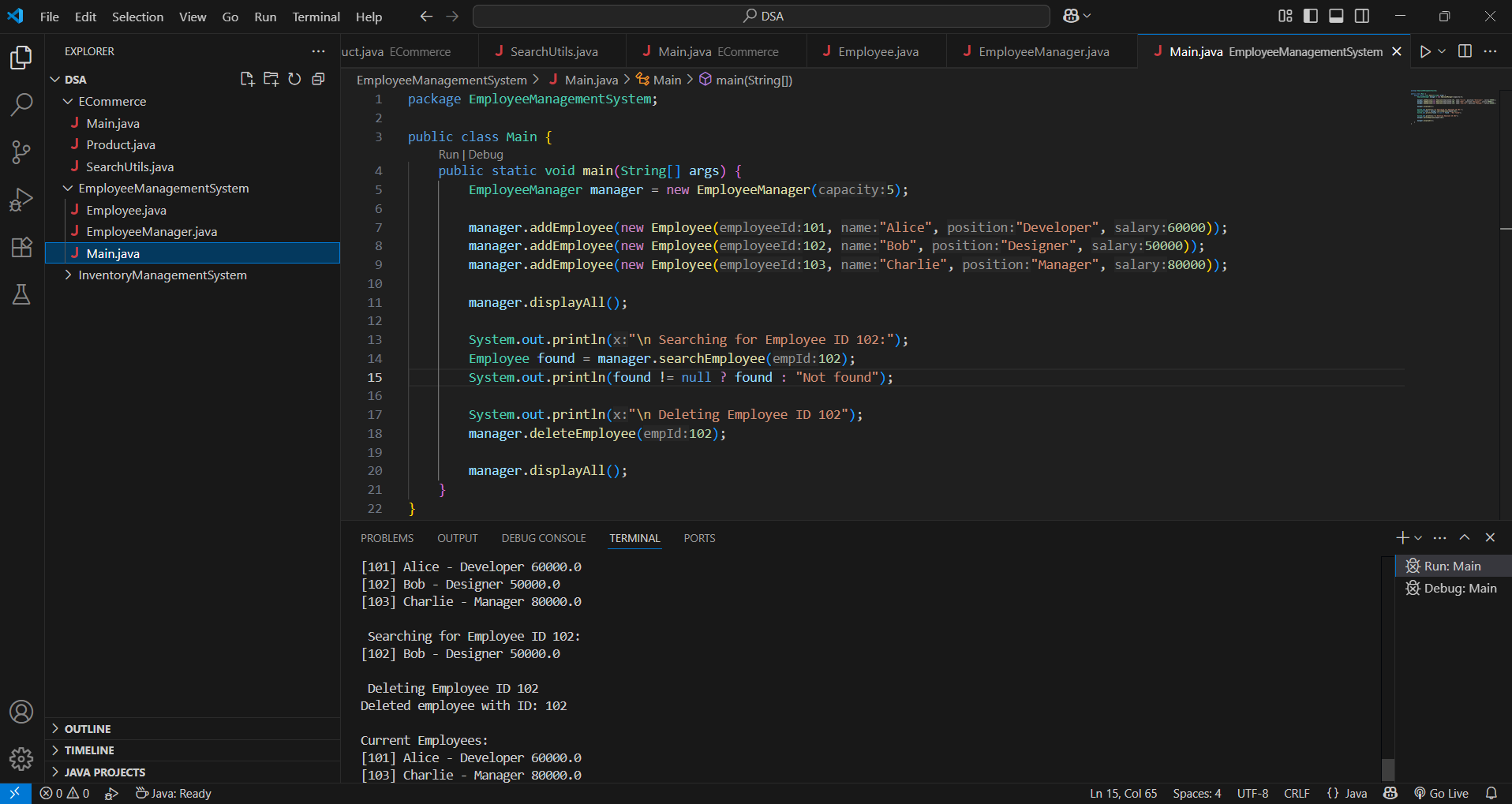
        manager.displayAll();

    }

}

**OUTPUT:**

****



**Exercise 5: Task Management System**

**CODE:**

**Task.java**

package TaskManagement;

public class Task {

    private int taskId;

    private String taskName;

    private String status;

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

        this.taskId = taskId;

        this.taskName = taskName;

        this.status = status;

    }

    public int getTaskId() {

        return taskId;

    }

    public String toString() {

        return "[" + taskId + "] " + taskName + " - Status: " + status;

    }

}

**Node.java**

package TaskManagement;

public class Node {

     Task task;

    Node next;

    public Node(Task task) {

        this.task = task;

        this.next = null;

    }

}

**TaskManager.java**

package TaskManagement;

public class TaskManager {

    private Node head;

    public void addTask(Task task) {

        Node newNode = new Node(task);

        if (head == null) {

            head = newNode;

        } else {

            Node curr = head;

            while (curr.next != null) {

                curr = curr.next;

            }

            curr.next = newNode;

        }

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

    }

    public Task searchTask(int taskId) {

        Node curr = head;

        while (curr != null) {

            if (curr.task.getTaskId() == taskId) {

                return curr.task;

            }

            curr = curr.next;

        }

        return null;

    }

    public void deleteTask(int taskId) {

        if (head == null) return;

        if (head.task.getTaskId() == taskId) {

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

            head = head.next;

            return;

        }

        Node prev = head;

        Node curr = head.next;

        while (curr != null) {

            if (curr.task.getTaskId() == taskId) {

                System.out.println("Deleted: " + curr.task);

                prev.next = curr.next;

                return;

            }

            prev = curr;

            curr = curr.next;

        }

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

    }

    public void displayTasks() {

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

        Node curr = head;

        while (curr != null) {

            System.out.println(curr.task);

            curr = curr.next;

        }

    }

}

**Main.java**

package TaskManagement;

public class Main {

    public static void main(String[] args) {

        TaskManager manager = new TaskManager();

        manager.addTask(new Task(1, "Design Database", "Pending"));

        manager.addTask(new Task(2, "Implement API", "In Progress"));

        manager.addTask(new Task(3, "Test Features", "Pending"));

        manager.displayTasks();

        System.out.println("\n Searching Task ID 2:");

        Task result = manager.searchTask(2);

        System.out.println(result != null ? result : "Not found");

        System.out.println("\n Deleting Task ID 2:");

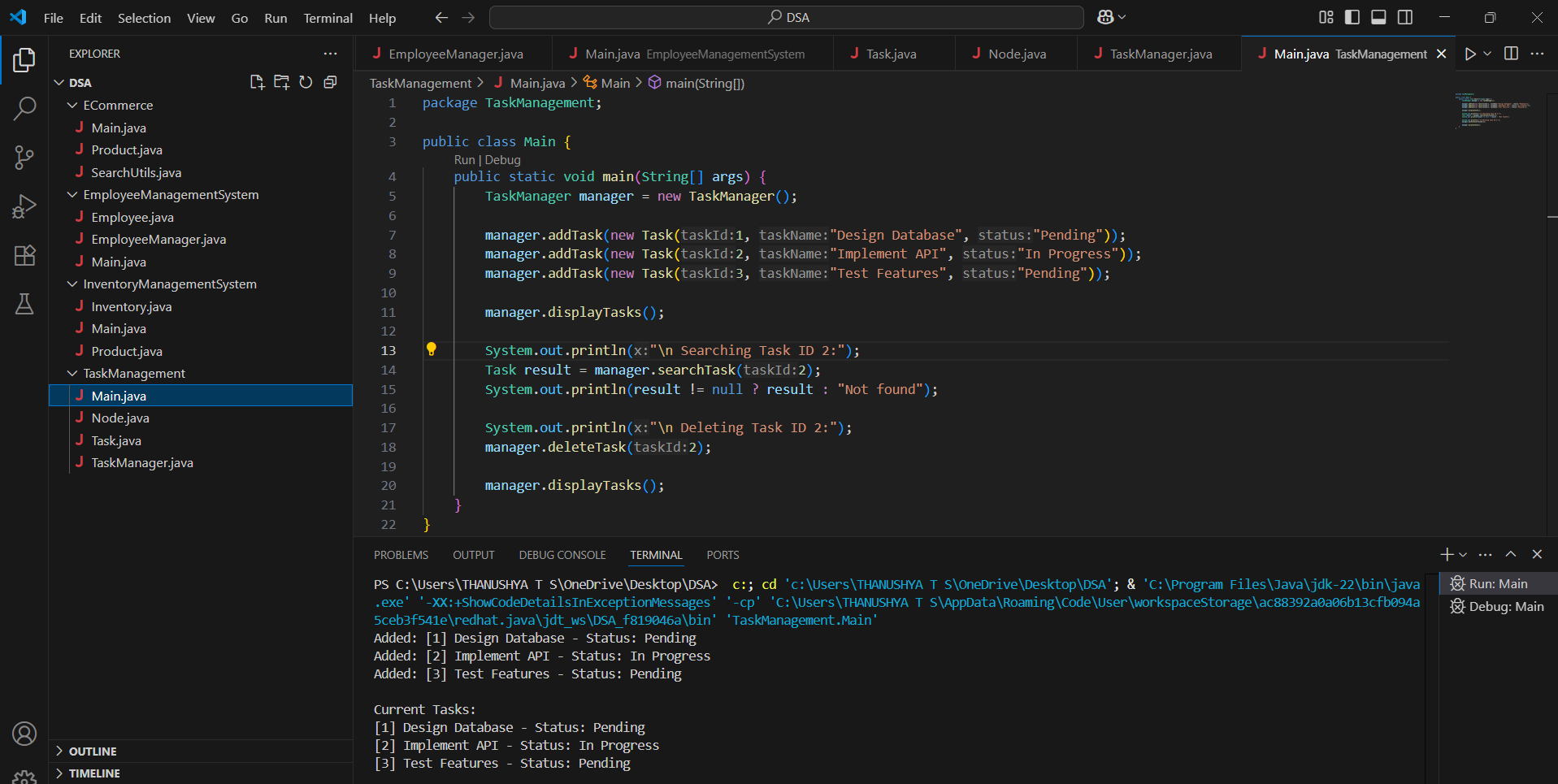
        manager.deleteTask(2);

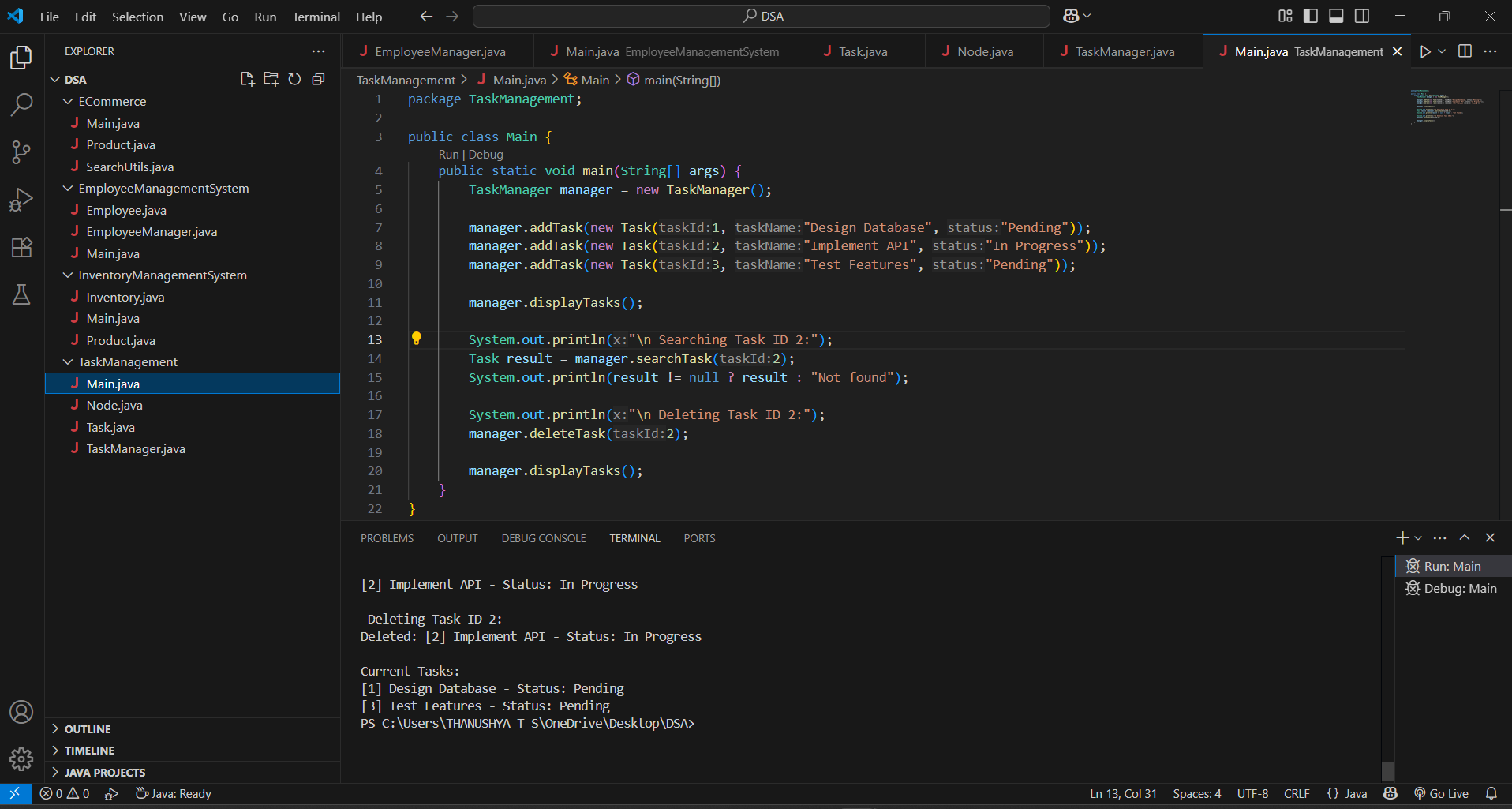
        manager.displayTasks();

    }

}

**OUTPUT:**





**Exercise 6: Library Management System**

**CODE:**

**Book.java**

package Library;

public class Book {

    private int bookId;

    private String title;

    private String author;

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

        this.bookId = bookId;

        this.title = title;

        this.author = author;

    }

    public String getTitle() {

        return title;

    }

    public String toString() {

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

    }

}

**SearchUtils.java**

package Library;

import java.util.Arrays;

import java.util.Comparator;

public class SearchUtils {

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

        for (Book book : books) {

            if (book.getTitle().equalsIgnoreCase(targetTitle)) {

                return book;

            }

        }

        return null;

    }

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

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

        while (left <= right) {

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

            int compare = books[mid].getTitle().compareToIgnoreCase(targetTitle);

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

            else if (compare < 0) left = mid + 1;

            else right = mid - 1;

        }

        return null;

    }

    public static void sortBooksByTitle(Book[] books) {

        Arrays.sort(books, Comparator.comparing(Book::getTitle, String.CASE\_INSENSITIVE\_ORDER));

    }

}

**Main.java**

package Library;

public class Main {

    public static void main(String[] args) {

        Book[] books = {

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

            new Book(102, "Clean Code", "Robert C. Martin"),

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

            new Book(104, "The Pragmatic Programmer", "Andrew Hunt")

        };

        System.out.println("Linear Search for '1984':");

        Book found1 = SearchUtils.linearSearch(books, "1984");

        System.out.println(found1 != null ? found1 : "Book not found");

        SearchUtils.sortBooksByTitle(books);

        System.out.println("\n Binary Search for '1984':");

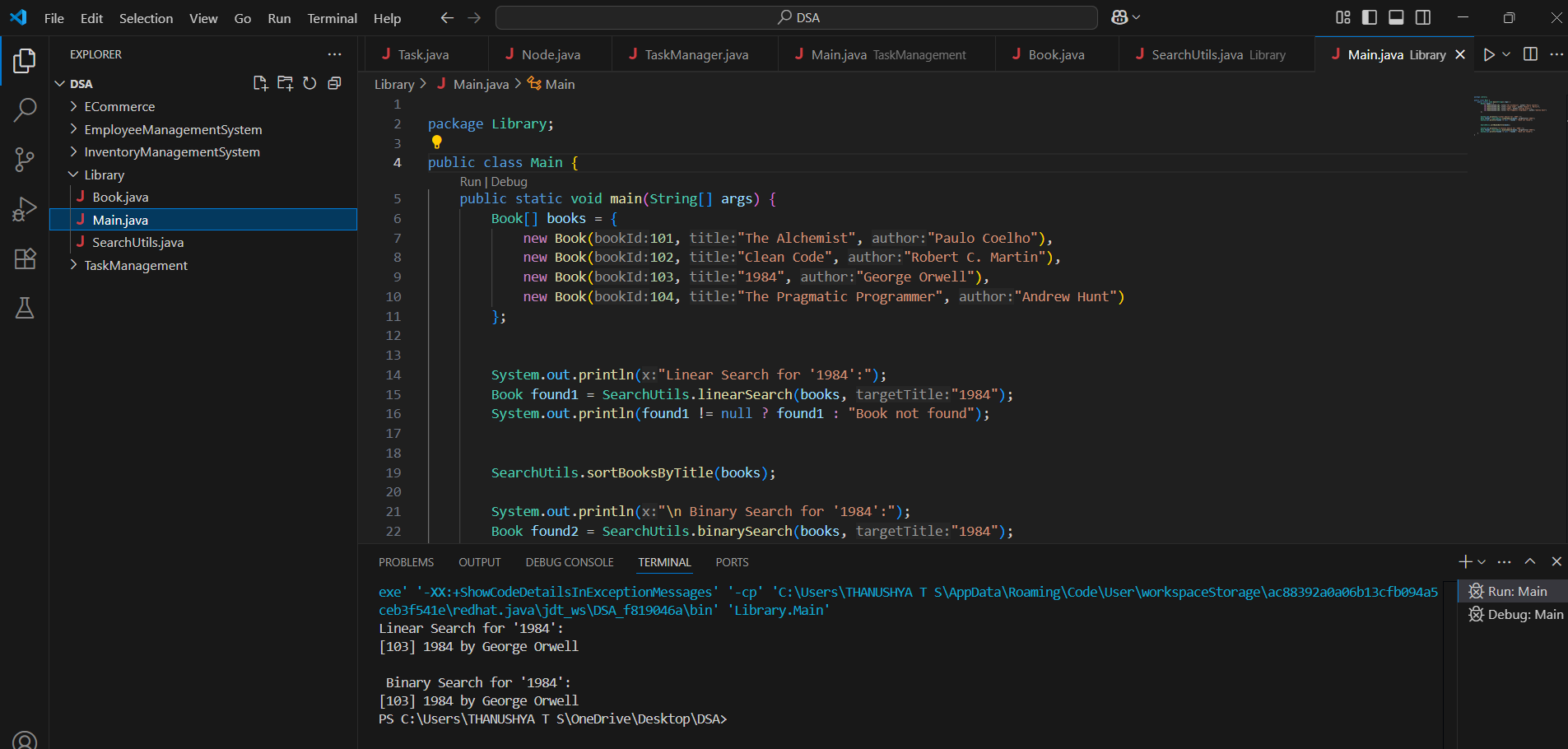
        Book found2 = SearchUtils.binarySearch(books, "1984");

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

    }

}

**OUTPUT:**



**Exercise 7: Financial Forecasting**

**CODE:**

**RecursiveForecast.java**

package Forecast;

public class RecursiveForecast {

     public static double futureValue(double initialValue, double rate, int years) {

        if (years == 0) return initialValue;

        return futureValue(initialValue, rate, years - 1) \* (1 + rate);

    }

    public static double futureValueMemo(double initialValue, double rate, int years, Double[] memo) {

        if (years == 0) return initialValue;

        if (memo[years] != null) return memo[years];

        memo[years] = futureValueMemo(initialValue, rate, years - 1, memo) \* (1 + rate);

        return memo[years];

    }

}

**Main.java**

package Forecast;

public class Main {

     public static void main(String[] args) {

        double initial = 1000.0;

        double rate = 0.10; // 10% annual growth

        int years = 5;

        System.out.println("Recursive Forecast:");

        double value = RecursiveForecast.futureValue(initial, rate, years);

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

        System.out.println("\n Optimized with Memoization:");

        Double[] memo = new Double[years + 1];

        double valueMemo = RecursiveForecast.futureValueMemo(initial, rate, years, memo);

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

    }

}

**OUTPUT:**

