**Exercise 1: Inventory Management System**

**InventoryManagementSystem.java**

import java.util.HashMap;

import java.util.Map;

import java.util.Scanner;

// Product class

class Product {

String productId;

String productName;

int quantity;

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 toString() {

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

}

}

// Inventory class using HashMap for fast access

class Inventory {

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

public void addProduct(Product p) {

inventory.put(p.productId, p);

System.out.println("Added: " + p.productName);

}

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

Product p = inventory.get(productId);

if (p != null) {

p.quantity = quantity;

p.price = price;

System.out.println("Updated: " + p.productName);

} else {

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

}

}

public void deleteProduct(String productId) {

Product removed = inventory.remove(productId);

if (removed != null) {

System.out.println("Deleted: " + removed.productName);

} else {

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

}

}

public void displayInventory() {

if (inventory.isEmpty()) {

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

} else {

System.out.println("----- Inventory -----");

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

System.out.println(p);

}

}

}

}

// Client/Main

public class InventoryManagementSystem {

public static void main(String[] args) {

Inventory inv = new Inventory();

Scanner sc = new Scanner(System.in);

inv.addProduct(new Product("P101", "Mouse", 10, 299.0));

inv.addProduct(new Product("P102", "Keyboard", 5, 699.0));

inv.addProduct(new Product("P103", "Monitor", 3, 5999.0));

inv.displayInventory();

System.out.println("Updating P102...");

inv.updateProduct("P102", 7, 749.0);

System.out.println("Deleting P101...");

inv.deleteProduct("P101");

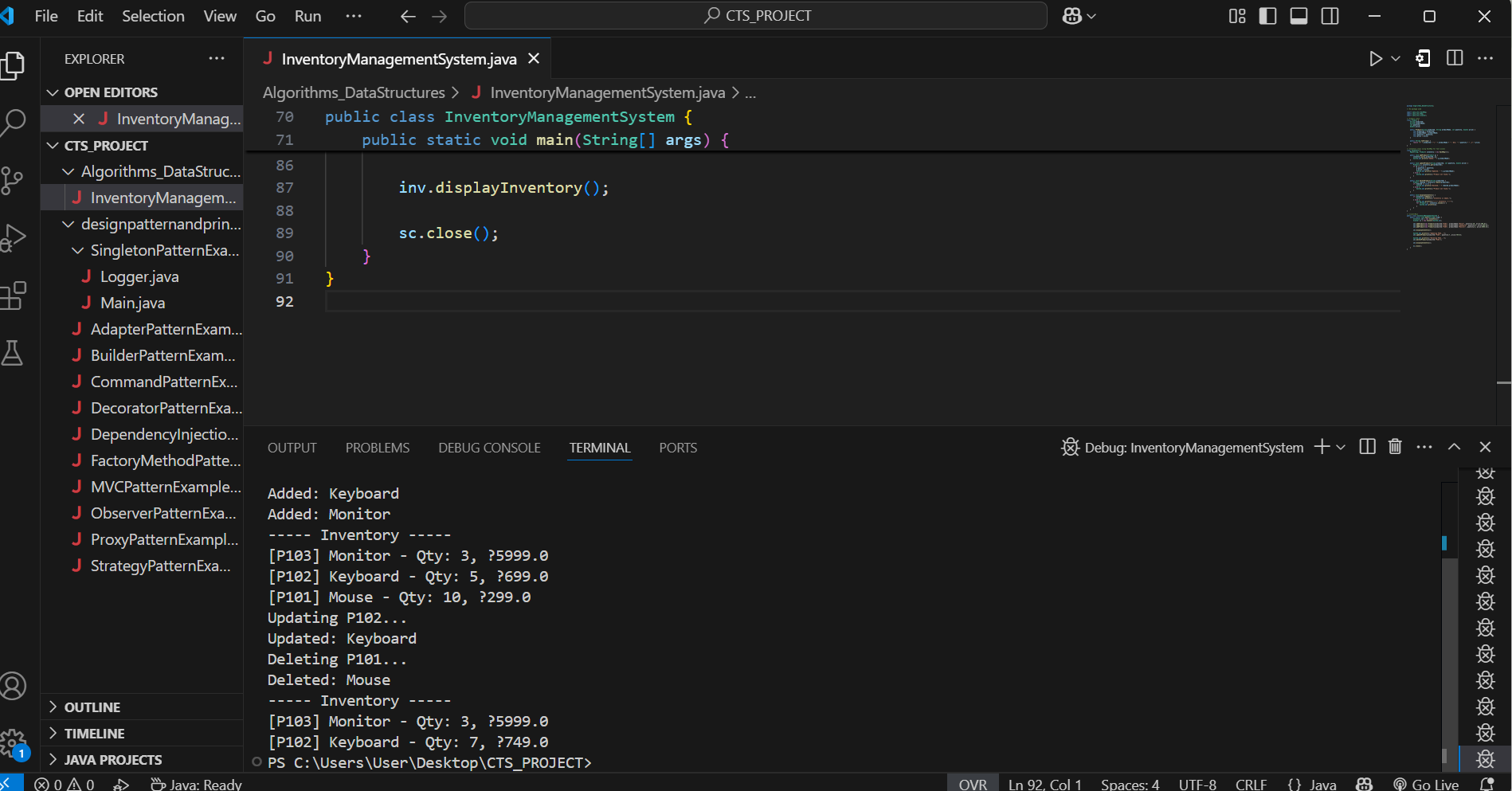
inv.displayInventory();

sc.close();

}

}

**Output**

****

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

EcommerceSearchFunction.java

import java.util.Arrays;

// Product class

class Product implements Comparable<Product> {

String productId;

String productName;

String category;

public Product(String id, String name, String category) {

this.productId = id;

this.productName = name;

this.category = category;

}

public String toString() {

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

}

public int compareTo(Product other) {

return this.productName.compareToIgnoreCase(other.productName);

}

}

// Search utility class

class 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;

}

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

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

while (low <= high) {

int mid = (low + high) / 2;

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

if (cmp == 0) return mid;

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

else high = mid - 1;

}

return -1;

}

}

// Main class

public class EcommerceSearchFunction {

public static void main(String[] args) {

Product[] products = {

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

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

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

new Product("P004", "Watch", "Accessories"),

new Product("P005", "Mobile", "Electronics")

};

// Linear Search

String searchName = "Shoes";

int index = Search.linearSearch(products, searchName);

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

if (index != -1)

System.out.println("Found: " + products[index]);

else

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

// Binary Search

Arrays.sort(products); // Required for binary search

searchName = "Mobile";

index = Search.binarySearch(products, searchName);

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

if (index != -1)

System.out.println("Found: " + products[index]);

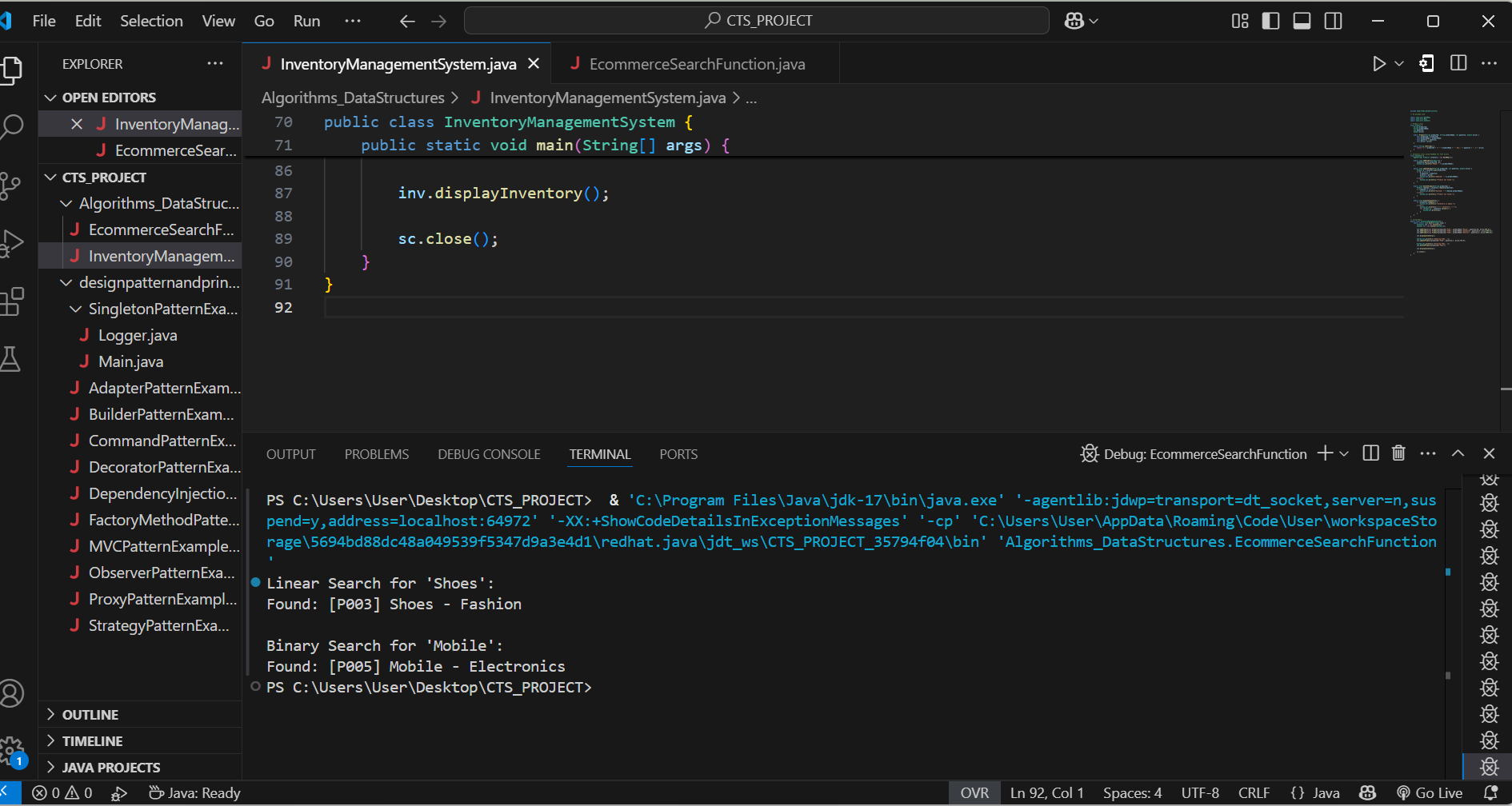
else

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

}

}

**Output**

****

**Exercise 3: Sorting Customer Orders**

**SortCustomerOrders.java**

package Algorithms\_DataStructures;

// No package used

import java.util.HashMap;

import java.util.Map;

import java.util.Scanner;

// Product class

class Product {

    String productId;

    String productName;

    int quantity;

    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 toString() {

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

    }

}

// Inventory class using HashMap for fast access

class Inventory {

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

    public void addProduct(Product p) {

        inventory.put(p.productId, p);

        System.out.println("Added: " + p.productName);

    }

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

        Product p = inventory.get(productId);

        if (p != null) {

            p.quantity = quantity;

            p.price = price;

            System.out.println("Updated: " + p.productName);

        } else {

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

        }

    }

    public void deleteProduct(String productId) {

        Product removed = inventory.remove(productId);

        if (removed != null) {

            System.out.println("Deleted: " + removed.productName);

        } else {

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

        }

    }

    public void displayInventory() {

        if (inventory.isEmpty()) {

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

        } else {

            System.out.println("----- Inventory -----");

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

                System.out.println(p);

            }

        }

    }

}

// Client/Main

public class InventoryManagementSystem {

    public static void main(String[] args) {

        Inventory inv = new Inventory();

        Scanner sc = new Scanner(System.in);

        inv.addProduct(new Product("P101", "Mouse", 10, 299.0));

        inv.addProduct(new Product("P102", "Keyboard", 5, 699.0));

        inv.addProduct(new Product("P103", "Monitor", 3, 5999.0));

        inv.displayInventory();

        System.out.println("Updating P102...");

        inv.updateProduct("P102", 7, 749.0);

        System.out.println("Deleting P101...");

        inv.deleteProduct("P101");

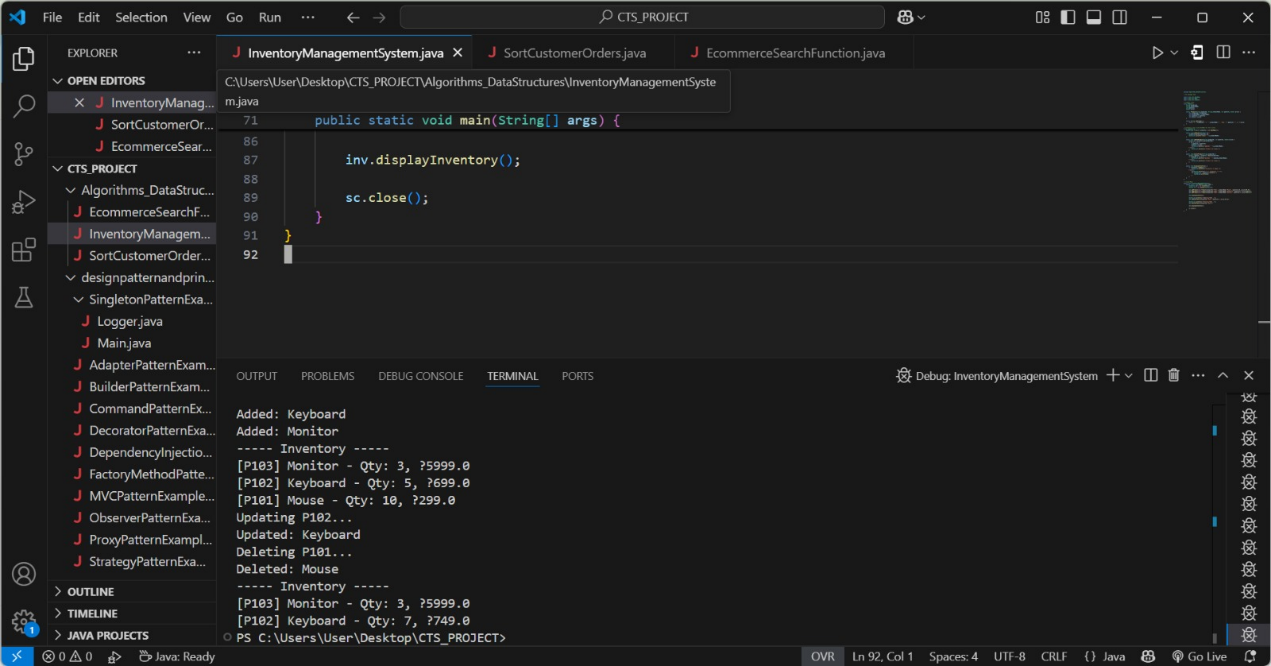
        inv.displayInventory();

        sc.close();

    }

}

**Output**

****

**Exercise 4: Employee Management System**

**EmployeeManagementSystem.java**

// No package used

import java.util.Scanner;

// Employee class

class Employee {

String employeeId;

String name;

String position;

double salary;

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

this.employeeId = id;

this.name = name;

this.position = position;

this.salary = salary;

}

public String toString() {

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

}

}

// Employee Manager using Array

class EmployeeManager {

Employee[] employees;

int count = 0;

public EmployeeManager(int size) {

employees = new Employee[size];

}

public void addEmployee(Employee emp) {

if (count < employees.length) {

employees[count++] = emp;

System.out.println("Employee added: " + emp.name);

} else {

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

}

}

public void searchEmployee(String id) {

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

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

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

return;

}

}

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

}

public void deleteEmployee(String id) {

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

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

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

// Shift elements

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 listEmployees() {

if (count == 0) {

System.out.println("No employees available.");

} else {

System.out.println("--- Employee List ---");

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

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

}

}

}

}

// Main

public class EmployeeManagementSystem {

public static void main(String[] args) {

EmployeeManager manager = new EmployeeManager(5); // max 5 employees

manager.addEmployee(new Employee("E001", "Mathii", "Developer", 45000));

manager.addEmployee(new Employee("E002", "Ravi", "Manager", 65000));

manager.addEmployee(new Employee("E003", "Anu", "Tester", 35000));

manager.listEmployees();

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

manager.searchEmployee("E002");

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

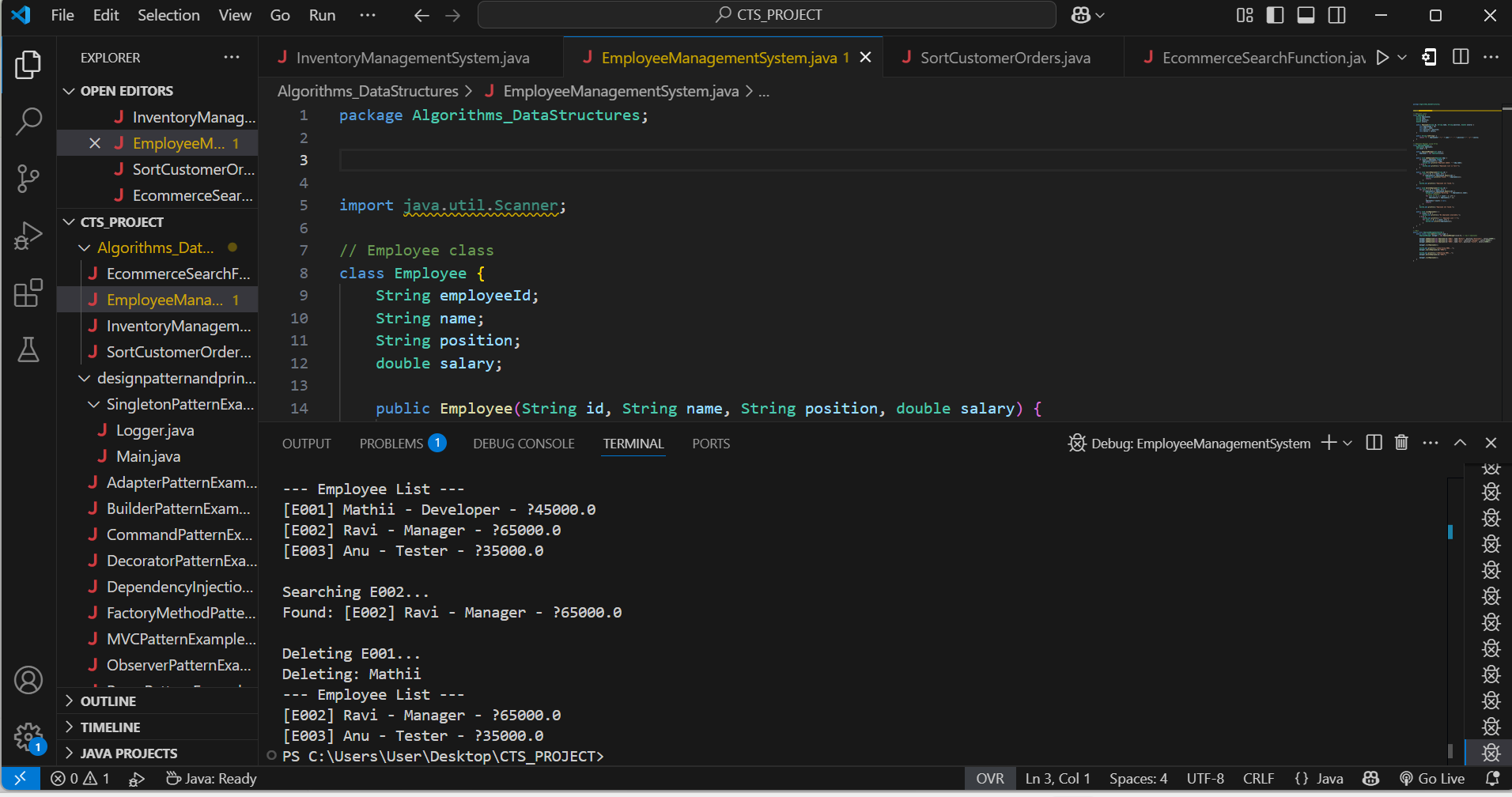
manager.deleteEmployee("E001");

manager.listEmployees();

}

}

**Output**

****

**Exercise 5: Task Management System**

**TaskManagementSystem.java**

// No package used

// Task node class

class Task {

String taskId;

String taskName;

String status;

Task next;

public Task(String id, String name, String status) {

this.taskId = id;

this.taskName = name;

this.status = status;

this.next = null;

}

public String toString() {

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

}

}

// Task Manager using Singly Linked List

class TaskManager {

private Task head;

public void addTask(String 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("Added task: " + name);

}

public void listTasks() {

if (head == null) {

System.out.println("No tasks available.");

return;

}

System.out.println("--- Task List ---");

Task current = head;

while (current != null) {

System.out.println(current);

current = current.next;

}

}

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.");

}

public void deleteTask(String taskId) {

if (head == null) {

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

return;

}

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

System.out.println("Deleting: " + head.taskName);

head = head.next;

return;

}

Task current = head;

while (current.next != null) {

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

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

current.next = current.next.next;

return;

}

current = current.next;

}

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

}

}

// Main

public class TaskManagementSystem {

public static void main(String[] args) {

TaskManager manager = new TaskManager();

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

manager.addTask("T002", "Connect DB", "In Progress");

manager.addTask("T003", "Implement Login", "Pending");

manager.listTasks();

System.out.println("\nSearching task T002...");

manager.searchTask("T002");

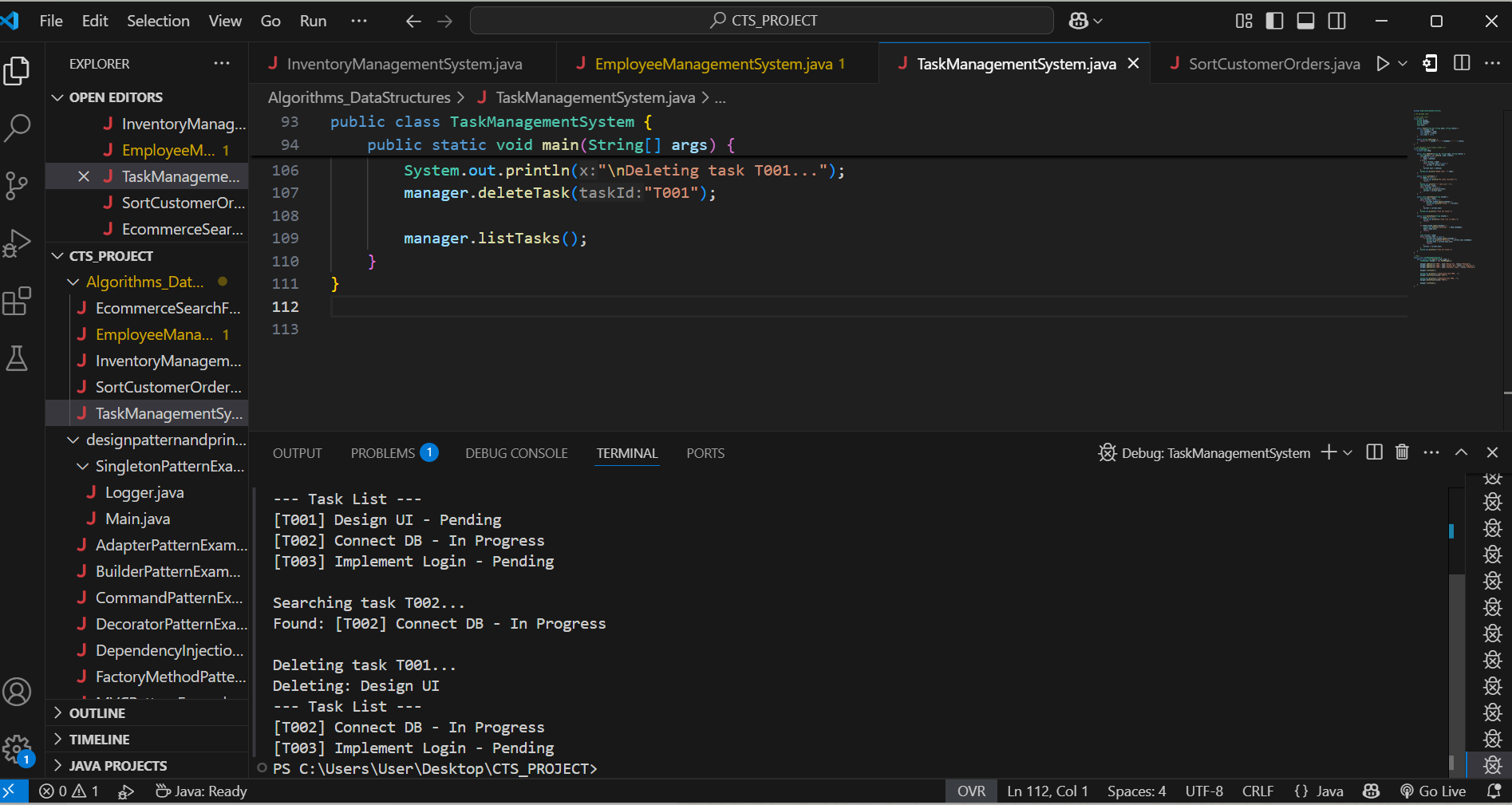
System.out.println("\nDeleting task T001...");

manager.deleteTask("T001");

manager.listTasks();

}}

**Output**

****

**Exercise 6: Library Management System**

**LibraryManagementSystem.java**

// No package used

import java.util.Arrays;

// Book class

class Book implements Comparable<Book> {

String bookId;

String title;

String author;

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

this.bookId = id;

this.title = title;

this.author = author;

}

public String toString() {

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

}

public int compareTo(Book other) {

return this.title.compareToIgnoreCase(other.title);

}

}

// Search class

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[] sortedBooks, String targetTitle) {

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

while (low <= high) {

int mid = (low + high) / 2;

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

if (cmp == 0) return mid;

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

else high = mid - 1;

}

return -1;

}

}

// Main

public class LibraryManagementSystem {

public static void main(String[] args) {

Book[] books = {

new Book("B001", "Java Programming", "James Gosling"),

new Book("B002", "Data Structures", "Mark Allen"),

new Book("B003", "Operating Systems", "Silberschatz"),

new Book("B004", "Computer Networks", "Tanenbaum"),

new Book("B005", "Machine Learning", "Andrew Ng")

};

// Linear Search

String searchTitle = "Operating Systems";

int index = LibrarySearch.linearSearch(books, searchTitle);

System.out.println("Linear Search for \"" + searchTitle + "\":");

if (index != -1)

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

else

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

// Binary Search

Arrays.sort(books); // Sort by title before binary search

searchTitle = "Data Structures";

index = LibrarySearch.binarySearch(books, searchTitle);

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

if (index != -1)

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

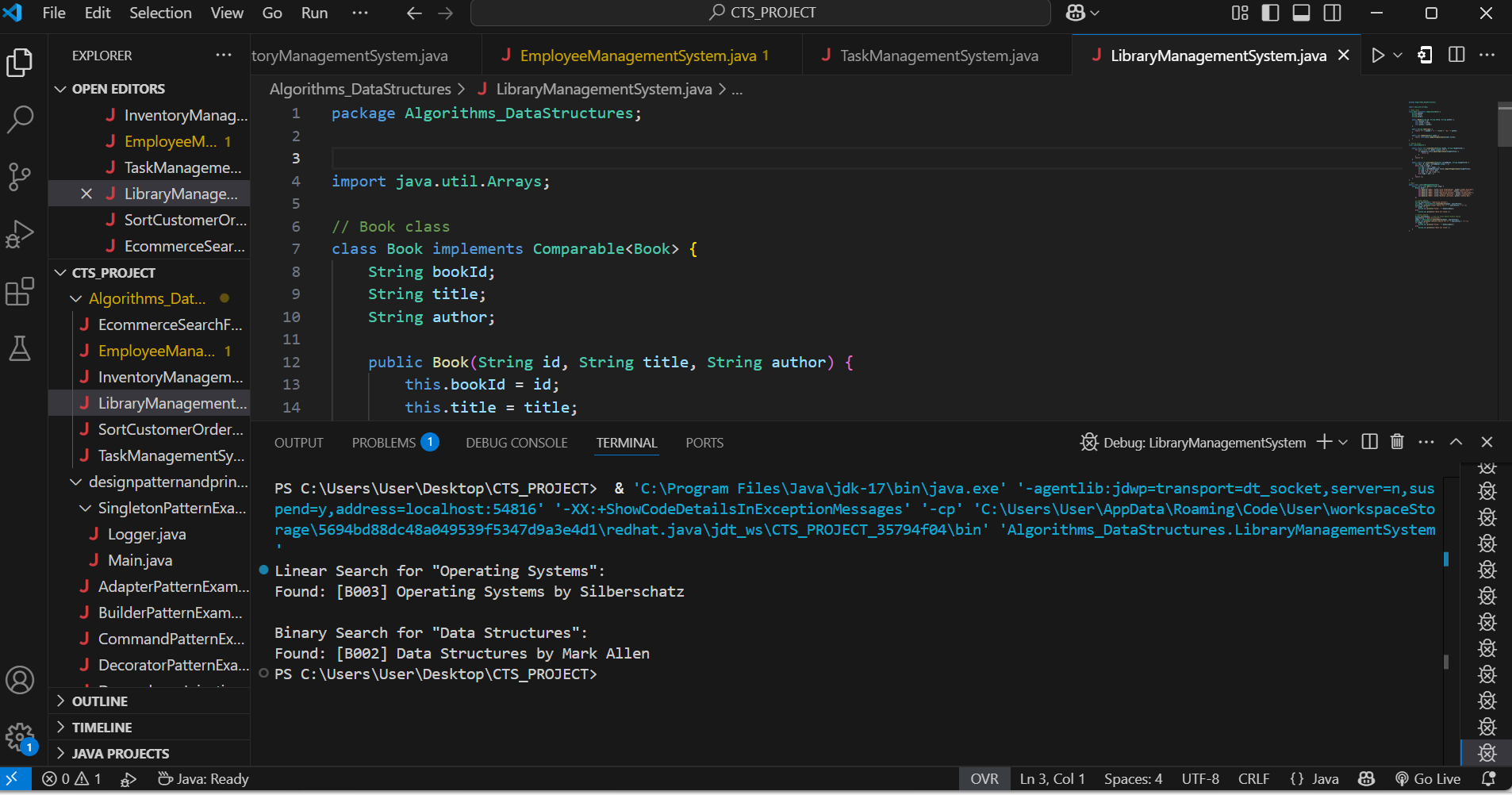
else

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

}

}

**Output**

****

**Exercise 7: Financial Forecasting**

**FinancialForecasting.java**

// Forecast utility

class Forecast {

// Recursive method to compute future value

public static double predictValue(double presentValue, double rate, int years) {

if (years == 0) return presentValue;

return predictValue(presentValue \* (1 + rate / 100), rate, years - 1);

}

// Optimized version using iteration (for comparison)

public static double predictIterative(double presentValue, double rate, int years) {

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

presentValue \*= (1 + rate / 100);

}

return presentValue;

}

}

// Main class

public class FinancialForecasting {

public static void main(String[] args) {

double currentAmount = 10000.0; // ₹10,000

double annualRate = 7.5; // 7.5% growth

int years = 5; // Forecast for 5 years

double futureValueRecursive = Forecast.predictValue(currentAmount, annualRate, years);

double futureValueIterative = Forecast.predictIterative(currentAmount, annualRate, years);

System.out.println("Present Value: ₹" + currentAmount);

System.out.println("Annual Growth Rate: " + annualRate + "%");

System.out.println("Forecast Period: " + years + " years");

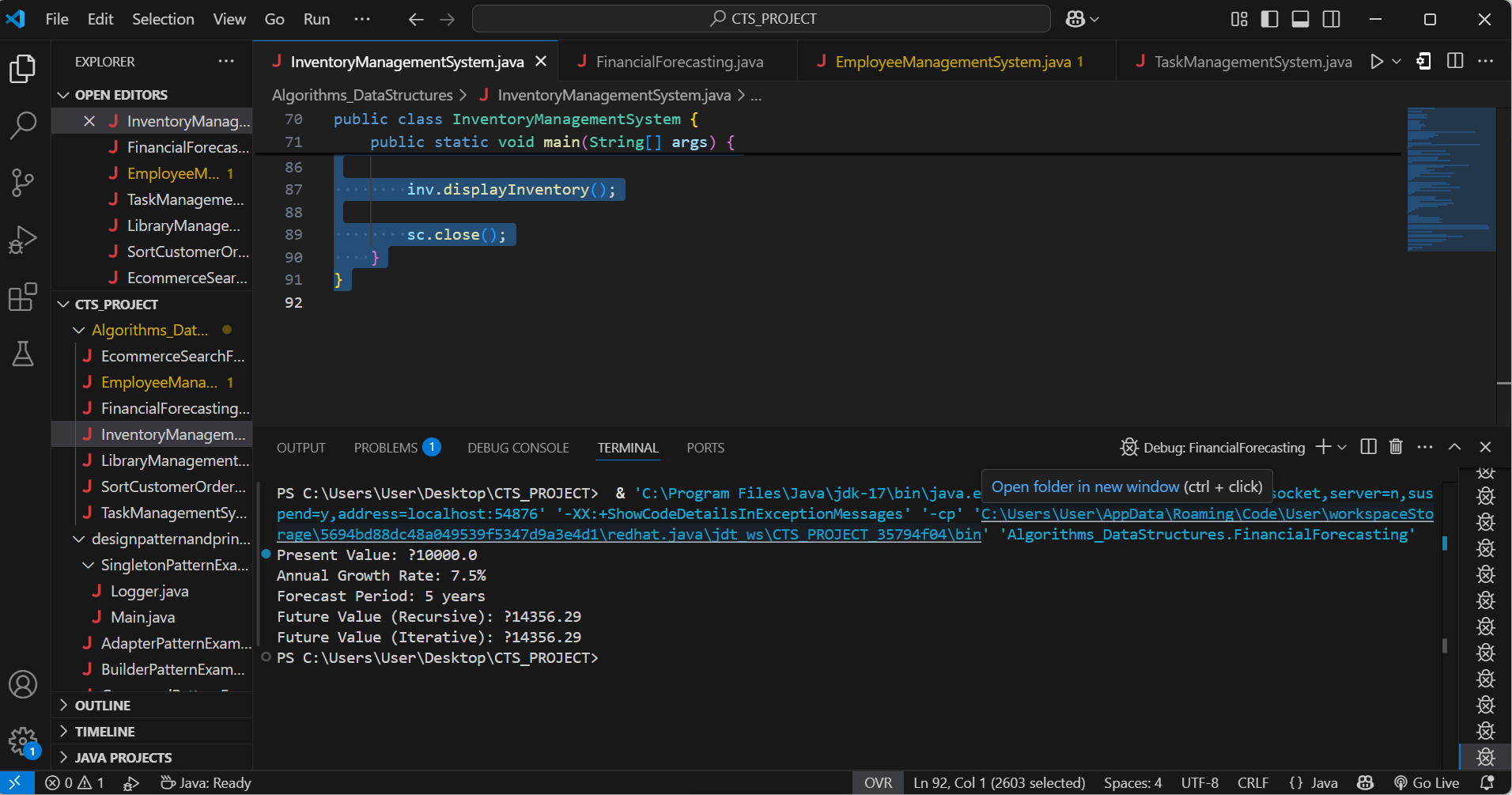
System.out.printf("Future Value (Recursive): ₹%.2f\n", futureValueRecursive);

System.out.printf("Future Value (Iterative): ₹%.2f\n", futureValueIterative);

}

}

**Output**

****