**WEEK - 1**

**Data Structures and Algorithms**

**Exercise 1: Inventory Management System**

**Product.java**

package com.inventorymanagement;

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 name) { this.productName = name; }

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 + ", Qty: " + quantity + ", Price: Rs" + price;

}

}

**InventoryManager.java**

package com.inventorymanagement;

import java.util.HashMap;

public class InventoryManager {

private HashMap<Integer, Product> inventory;

public InventoryManager() {

inventory = new HashMap<>();

}

public void addProduct(Product product) {

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

}

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

Product product = inventory.get(productId);

if (product != null) {

product.setProductName(name);

product.setQuantity(quantity);

product.setPrice(price);

} else {

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

}

}

public void deleteProduct(int productId) {

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

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

}

}

public void printInventory() {

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

System.out.println(product);

}

}

}

**Main.java**

package com.inventorymanagement;

public class Main {

public static void main(String[] args) {

InventoryManager manager = new InventoryManager();

manager.addProduct(new Product(1, "Laptop", 10, 40000.0));

manager.addProduct(new Product(2, "Monitor", 20, 9500.0));

System.out.println("Initial Inventory:");

manager.printInventory();

manager.updateProduct(1, "Gaming Laptop", 8, 60990.0);

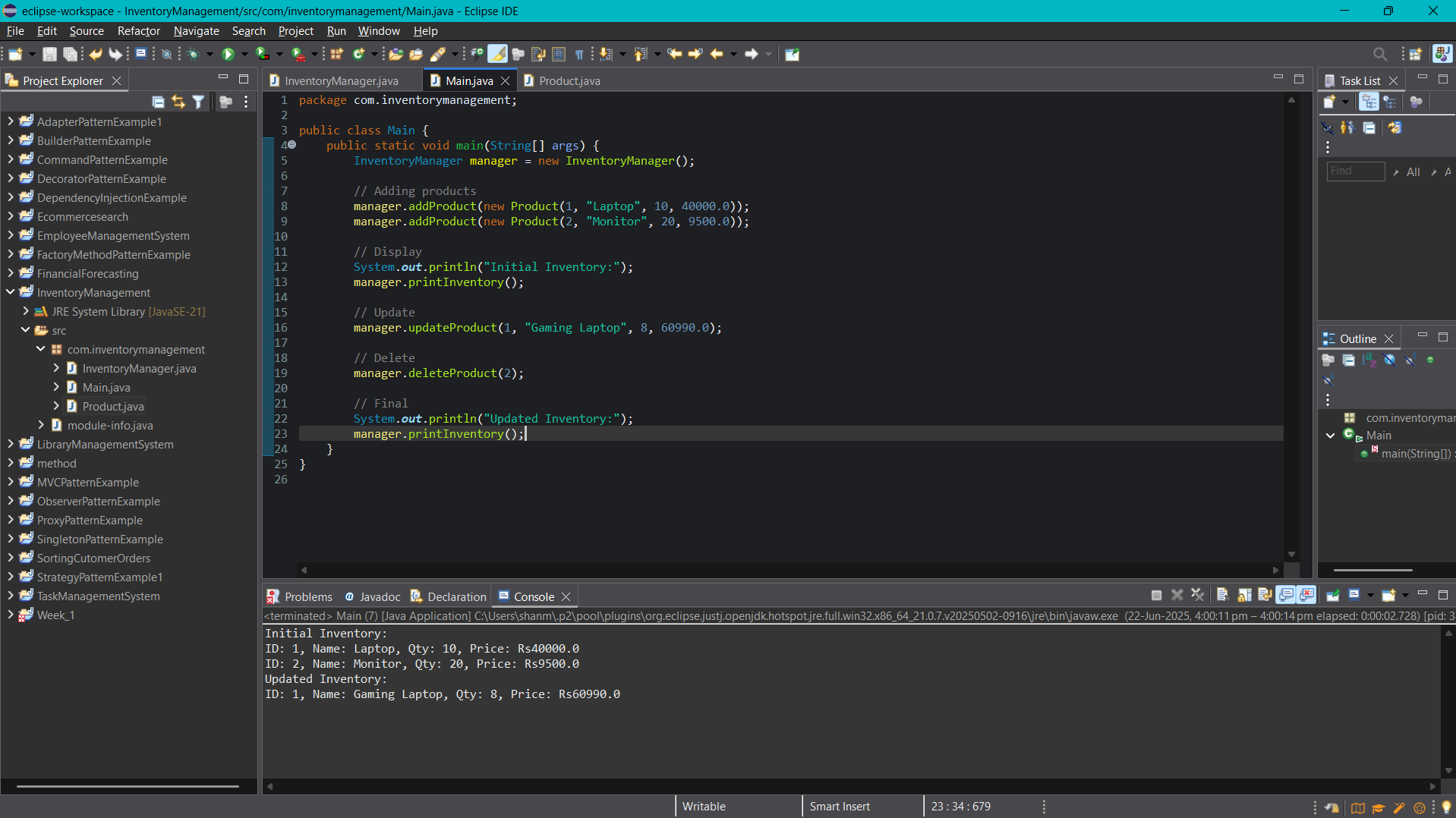
manager.deleteProduct(2);

System.out.println("Updated Inventory:");

manager.printInventory();

}

}



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

**Product.java**

package com.ecommerce;

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

}

}

**SearchService.java**

package com.ecommerce;

import java.util.Arrays;

import java.util.Comparator;

public class SearchService {

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

for (Product product : products) {

if (product.productName.equalsIgnoreCase(nameToSearch)) {

return product;

}

}

return null;

}

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

int left = 0;

int right = products.length - 1;

while (left <= right) {

int mid = (left + right) / 2;

Int compareResult = products[mid].productName.compareToIgnoreCase(nameToSearch);

if (compareResult == 0) {

return products[mid];

} else if (compareResult < 0) {

left = mid + 1;

} else {

right = mid - 1;

}

}

return null;

}

public static void sortProducts(Product[] products) {

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

}

}

**MainApp.java**

package com.ecommerce;

public class MainApp {

public static void main(String[] args) {

Product[] products = {

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

new Product(102, "Shoes", "Fashion"),

new Product(103, "Camera", "Electronics"),

new Product(104, "Watch", "Accessories")

};

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

Product found1 = SearchService.linearSearch(products, "Camera");

if (found1 != null) {

System.out.println("Found: " + found1);

} else {

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

}

SearchService.sortProducts(products);

System.out.println("\n>Binary Search ");

Product found2 = SearchService.binarySearch(products, "Camera");

if (found2 != null) {

System.out.println("Found: " + found2);

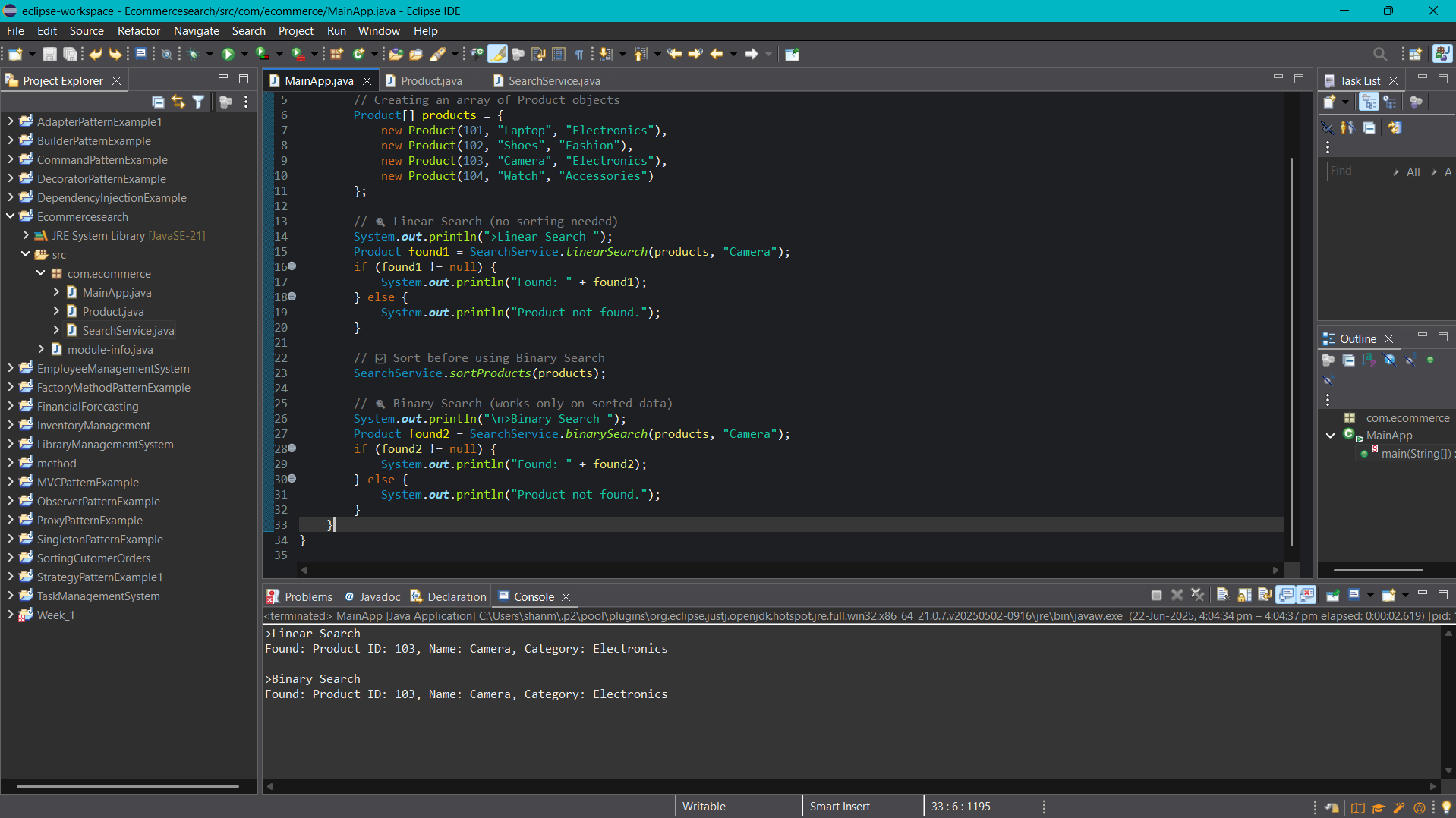
} else {

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

}

}

}



**Exercise 3: Sorting Customer Orders**

**Order.java**

package com.sorting;

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 "Order{" +

"ID=" + orderId +

", Name='" + customerName + '\'' +

", Price=" + totalPrice +

'}';

}

}

**OrderSorter.java**

package com.sorting;

public interface OrderSorter {

void sort(Order[] orders);

}

**BubbleSortOrderSorter.java**

package com.sorting;

public class BubbleSortOrderSorter implements OrderSorter{

@Override

public void sort(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;

}

}

}

}

}

**QuickSortOrderSorter.java**

package com.sorting;

public class QuickSortOrderSorter implements OrderSorter{

@Override

public void sort(Order[] orders) {

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

}

private void quickSort(Order[] orders, int low, int high) {

if (low < high) {

int pivotIndex = partition(orders, low, high);

quickSort(orders, low, pivotIndex - 1);

quickSort(orders, pivotIndex + 1, high);

}

}

private 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;

}

}

**Main.java**

package com.sorting;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Order[] orders = {

new Order(101, "Apple", 100.75),

new Order(102, "Banana", 70.50),

new Order(103, "Cherry", 120.00),

new Order(104, "Mango", 200.00)

};

System.out.println("Choose Sorting Method:");

System.out.println("1. Bubble Sort (slow)");

System.out.println("2. Quick Sort (fast)");

System.out.print("Your choice: ");

Scanner sc = new Scanner(System.in);

int choice = sc.nextInt();

OrderSorter sorter;

if (choice == 1) {

sorter = new BubbleSortOrderSorter();

} else {

sorter = new QuickSortOrderSorter();

}

System.out.println("\nOrders Before Sorting:");

for (Order order : orders) {

System.out.println(order);

}

sorter.sort(orders);

System.out.println("\nOrders After Sorting by Price (Low to High):");

for (Order order : orders) {

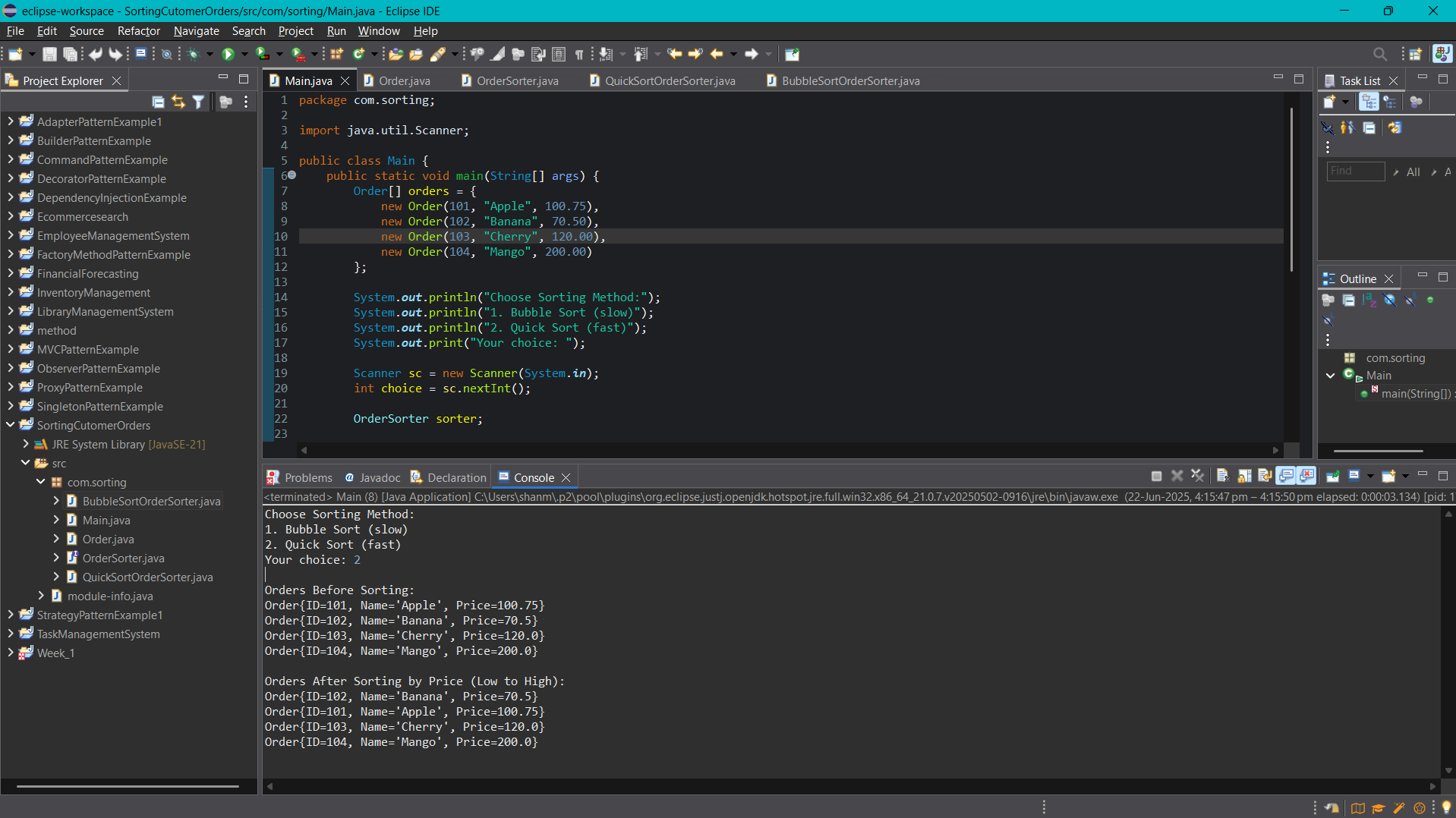
System.out.println(order);

}

sc.close();

}

}



**Exercise 4: Employee Management System**

**Employee.java**

package com.employeemanagement;

public class Employee {

int employeeId;

String name;

String position;

double salary;

// Constructor

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

this.employeeId = employeeId;

this.name = name;

this.position = position;

this.salary = salary;

}

public void display() {

System.out.println("ID: " + employeeId + ", Name: " + name +

", Position: " + position + ", Salary: " + salary);

}

}

**EmployeeManagementSystem.java**

**Employee.java**

package com.employeemanagement;

public class Employee {

int employeeId;

String name;

String position;

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 void display() {

System.out.println("ID: " + employeeId + ", Name: " + name +

", Position: " + position + ", Salary: Rs" + salary);

}

}

**EmployeeManagement.java**

package com.employeemanagement;

public class EmployeeManagementSystem {

private Employee[] employees;

private int size;

public EmployeeManagementSystem(int capacity) {

employees = new Employee[capacity];

size = 0;

}

public void addEmployee(Employee emp) {

if (size < employees.length) {

employees[size++] = emp;

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

} else {

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

}

}

public Employee searchEmployee(int id) {

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

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

return employees[i];

}

}

return null;

}

public void displayAllEmployees() {

if (size == 0) {

System.out.println("No employees to display.");

return;

}

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

employees[i].display();

}

}

public void deleteEmployee(int id) {

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

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

// Shift remaining elements to left

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

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

}

employees[--size] = null;

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

return;

}

}

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

}

}

**Main.java**

package com.employeemanagement;

public class Main {

public static void main(String[] args) {

EmployeeManagementSystem system = new EmployeeManagementSystem(5);

system.addEmployee(new Employee(101, "Ram", "Manager", 150000));

system.addEmployee(new Employee(102, "Pawan", "Developer", 90000));

system.addEmployee(new Employee(103, "Chiru", "Analyst", 100000));

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

system.displayAllEmployees();

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

Employee found = system.searchEmployee(102);

if (found != null) {

found.display();

} else {

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

}

System.out.println("\n Deleting employee with ID 102:");

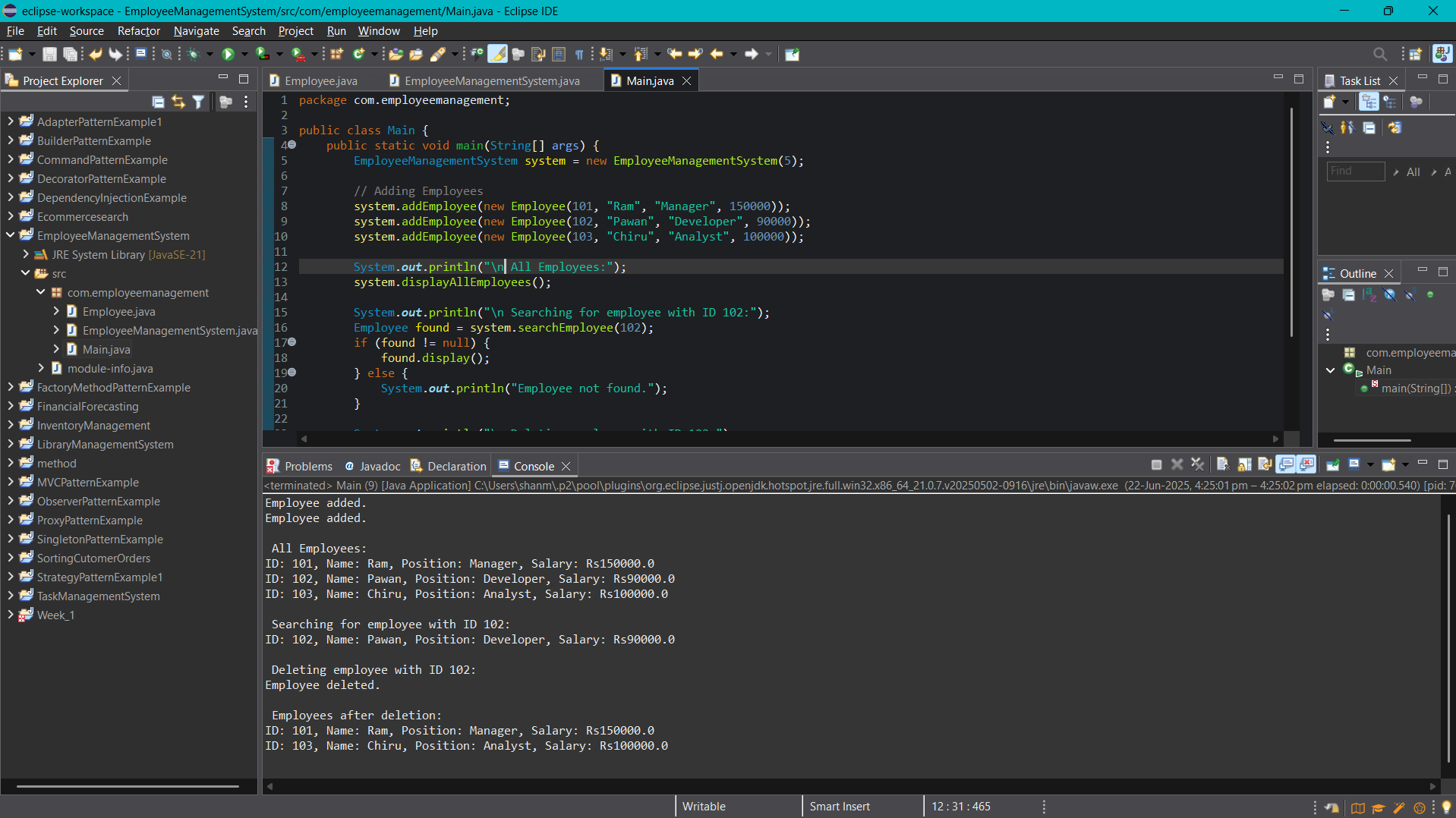
system.deleteEmployee(102);

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

system.displayAllEmployees();

}

}



**Exercise 5: Task Management System**

**Task.java**

package com.taskmanagement;

public class Task {

int taskId;

String taskName;

String status;

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

this.taskId = taskId;

this.taskName = taskName;

this.status = status;

}

@Override

public String toString() {

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

}

}

**TaskNode.java**

package com.taskmanagement;

public class TaskNode {

Task task;

TaskNode next;

public TaskNode(Task task) {

this.task = task;

this.next = null;

}

}

**TaskLinkedList.java**

package com.taskmanagement;

public class TaskLinkedList {

TaskNode head;

public void addTask(Task task) {

TaskNode newNode = new TaskNode(task);

if (head == null) {

head = newNode;

} else {

TaskNode current = head;

while (current.next != null) {

current = current.next;

}

current.next = newNode;

}

}

public Task searchTask(int taskId) {

TaskNode current = head;

while (current != null) {

if (current.task.taskId == taskId) {

return current.task;

}

current = current.next;

}

return null;

}

public boolean deleteTask(int taskId) {

if (head == null) return false;

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

head = head.next;

return true;

}

TaskNode current = head;

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

current = current.next;

}

if (current.next != null) {

current.next = current.next.next;

return true;

}

return false;

}

public void traverseTasks() {

TaskNode current = head;

if (current == null) {

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

return;

}

while (current != null) {

System.out.println(current.task);

current = current.next;

}

}

}

**Main.java**

package com.taskmanagement;

public class Main {

public static void main(String[] args) {

TaskLinkedList taskList = new TaskLinkedList();

taskList.addTask(new Task(1, "UI Designing", "Pending"));

taskList.addTask(new Task(2, "Backend Coding", "In Progress"));

taskList.addTask(new Task(3, "System Testing", "Pending"));

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

taskList.traverseTasks();

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

Task foundTask = taskList.searchTask(2);

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

System.out.println("\n Deleting Task ID 1...");

boolean isDeleted = taskList.deleteTask(1);

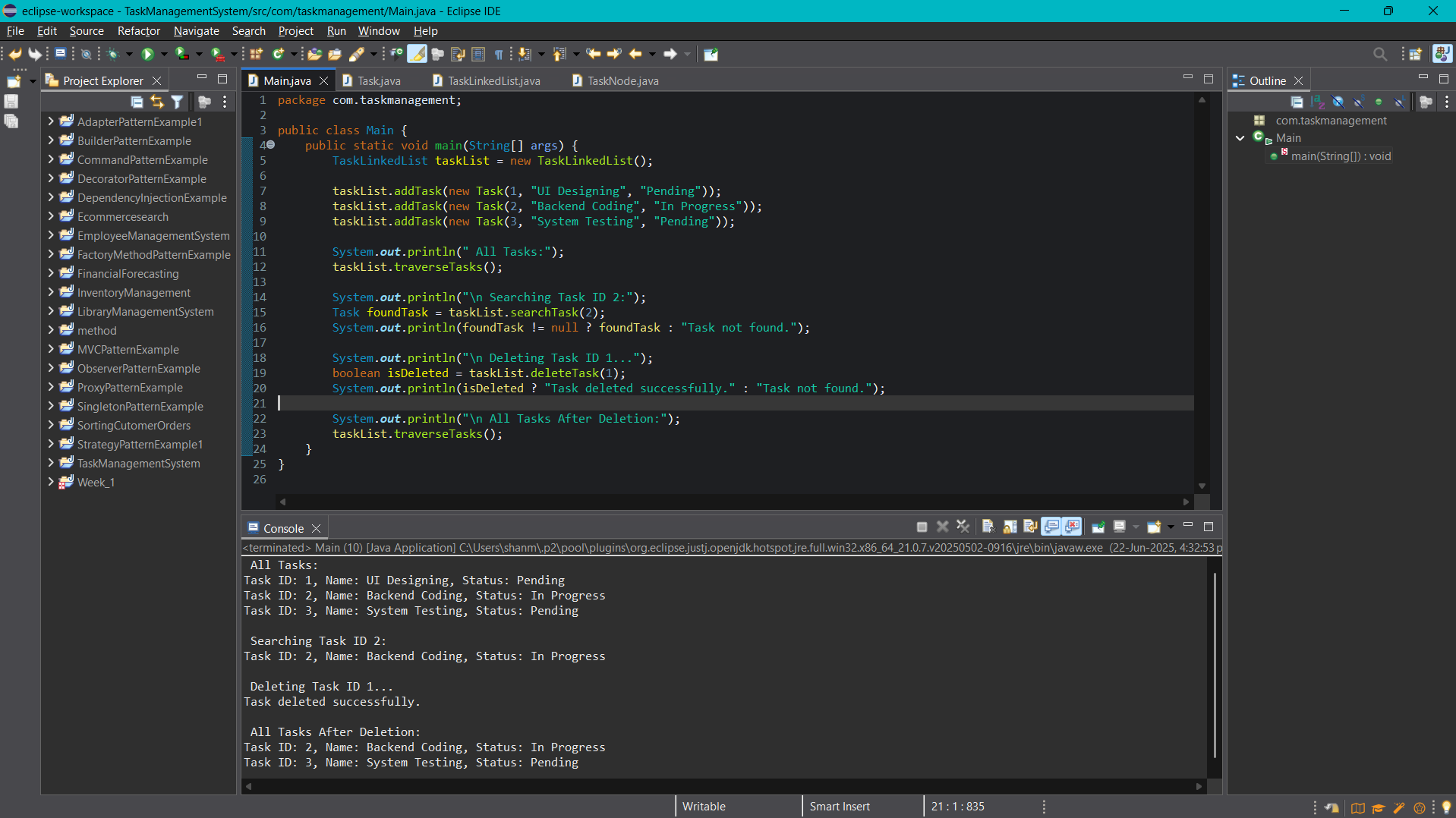
System.out.println(isDeleted ? "Task deleted successfully." : "Task not found.");

System.out.println("\n All Tasks After Deletion:");

taskList.traverseTasks();

}

}



**Exercise 6: Library Management System**

**Book.java**

package com.librarymanagement;

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.toLowerCase(); // for case-insensitive search

this.author = author.toLowerCase();

}

public int getBookId() {

return bookId;

}

public String getTitle() {

return title;

}

public String getAuthor() {

return author;

}

@Override

public String toString() {

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

}

}

**Library.java**

package com.librarymanagement;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Comparator;

import java.util.List;

public class Library {

public static List<Book> linearSearchByTitle(List<Book> books, String title) {

List<Book> result = new ArrayList<>();

String searchTitle = title.toLowerCase();

for (Book book : books) {

if (book.getTitle().contains(searchTitle)) {

result.add(book);

}

}

return result;

}

public static int binarySearchByTitle(List<Book> books, String title) {

String searchTitle = title.toLowerCase();

int low = 0, high = books.size() - 1;

while (low <= high) {

int mid = (low + high) / 2;

int cmp = books.get(mid).getTitle().compareTo(searchTitle);

if (cmp == 0) {

return mid;

} else if (cmp < 0) {

low = mid + 1;

} else {

high = mid - 1;

}

}

return -1; // not found

}

public static void sortBooksByTitle(List<Book> books) {

Collections.sort(books, Comparator.comparing(Book::getTitle));

}

}

**Main.java**

package com.librarymanagement;

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

List<Book> books = new ArrayList<>();

books.add(new Book(1, "The Alchemist", "Paulo Coelho"));

books.add(new Book(2, "Java Programming", "James Gosling"));

books.add(new Book(3, "Data Structures", "Seymour Lipschutz"));

books.add(new Book(4, "Algorithms Unlocked", "Thomas Cormen"));

System.out.println("===== Library Search Menu =====");

System.out.println("1. Linear Search by Title");

System.out.println("2. Binary Search by Title");

System.out.print("Enter your choice (1 or 2): ");

int choice = sc.nextInt();

sc.nextLine(); // consume newline

System.out.print("Enter book title to search: ");

String titleToSearch = sc.nextLine();

if (choice == 1) {

List<Book> result = Library.linearSearchByTitle(books, titleToSearch);

if (!result.isEmpty()) {

System.out.println("Books found:");

for (Book book : result) {

System.out.println(book);

}

} else {

System.out.println("No book found with title containing: " + titleToSearch);

}

} else if (choice == 2) {

Library.sortBooksByTitle(books);

int index = Library.binarySearchByTitle(books, titleToSearch);

if (index != -1) {

System.out.println("Book found:");

System.out.println(books.get(index));

} else {

System.out.println("Book not found with exact title: " + titleToSearch);

}

} else {

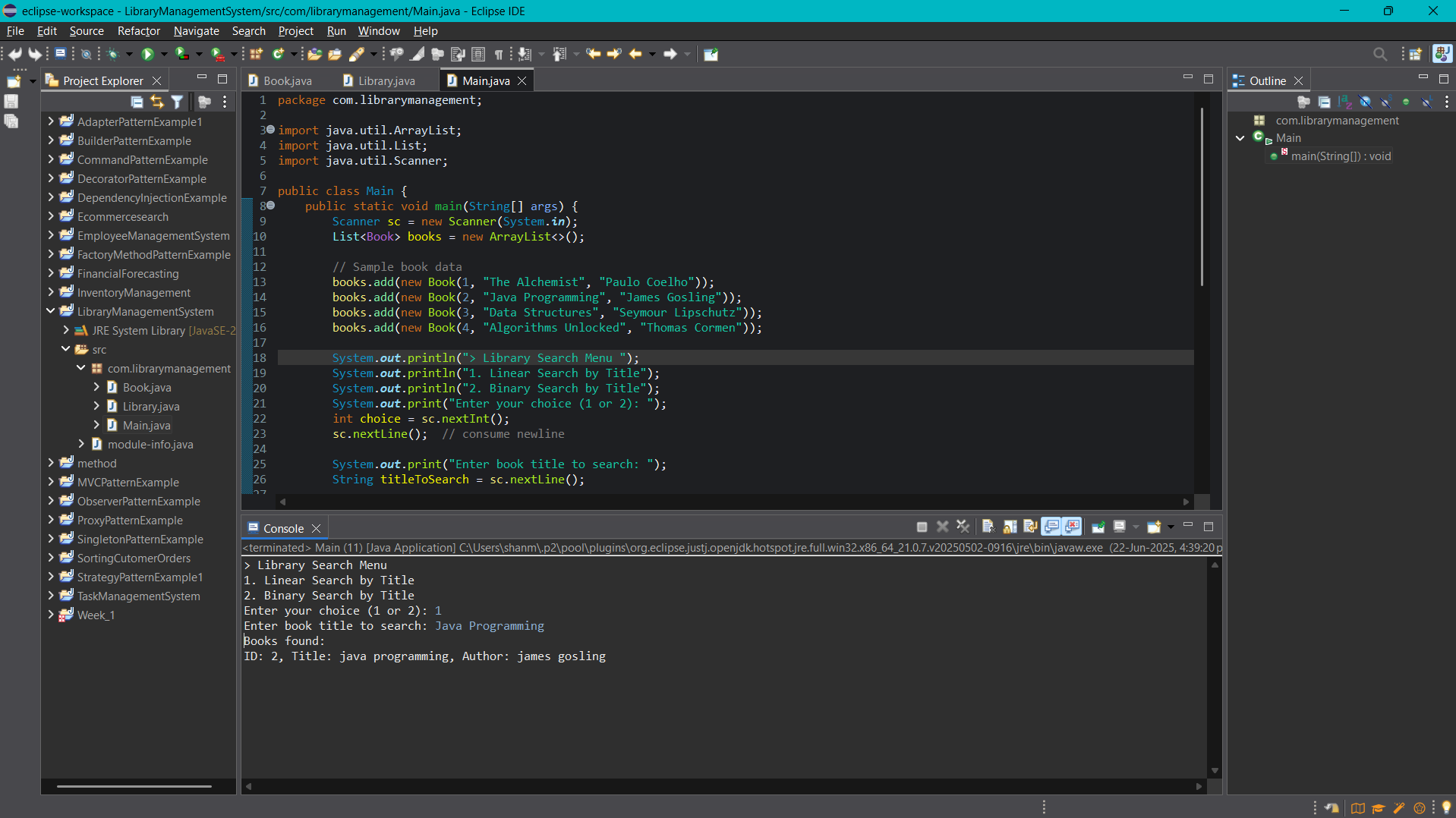
System.out.println("Invalid choice.");

}

sc.close();

}

}



**Exercise 7: Financial Forecasting**

**ForecastModel.java**

package com.financialforecasting;

public class ForecastModel {

private double presentValue;

private double growthRate;

private int years;

public ForecastModel(double presentValue, double growthRate, int years) {

this.presentValue = presentValue;

this.growthRate = growthRate;

this.years = years;

}

public double getPresentValue() {

return presentValue;

}

public double getGrowthRate() {

return growthRate;

}

public int getYears() {

return years;

}

}

**ForecastService.java**

package com.financialforecasting;

public class ForecastService {

public double calculateRecursively(ForecastModel model) {

return futureValueRecursive(model.getPresentValue(), model.getGrowthRate(), model.getYears());

}

private double futureValueRecursive(double presentValue, double rate, int years) {

if (years == 0) {

return presentValue;

}

return futureValueRecursive(presentValue, rate, years - 1) \* (1 + rate);

}

public double calculateIteratively(ForecastModel model) {

double result = model.getPresentValue();

for (int i = 0; i < model.getYears(); i++) {

result \*= (1 + model.getGrowthRate());

}

return result;

}

public double calculateUsingFormula(ForecastModel model) {

return model.getPresentValue() \* Math.pow(1 + model.getGrowthRate(), model.getYears());

}

}

**Main.java**

package com.financialforecasting;

public class Main {

public static void main(String[] args) {

ForecastModel model = new ForecastModel(1000.0, 0.05, 5);

ForecastService service = new ForecastService();

double recursiveResult = service.calculateRecursively(model);

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

double iterativeResult = service.calculateIteratively(model);

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

double formulaResult = service.calculateUsingFormula(model);

System.out.printf("Formula-based Future Value: %.2f%n", formulaResult);

}

}

