**Week 1**

Mandatory

Data Structures and Algorithms

1. Ecommerce search

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 String getProductName() {**

**return productName;**

**}**

**@Override**

**public String toString() {**

**return "Product [ID=" + productId + ", Name=" + productName + ", Category=" + category + "]";**

**}**

**}**

SearchFunctions.java

**import java.util.\*;**

**public class SearchFunctions {**

**// Linear search method**

**public static Product linearSearch(List<Product> products, String targetName) {**

**for (Product p : products) {**

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

**return p;**

**}**

**}**

**return null;**

**}**

**// Binary search method**

**public static Product binarySearch(List<Product> products, String targetName) {**

**int low = 0;**

**int high = products.size() - 1;**

**while (low <= high) {**

**int mid = (low + high) / 2;**

**String midName = products.get(mid).getProductName().toLowerCase();**

**int cmp = midName.compareTo(targetName.toLowerCase());**

**if (cmp == 0)**

**return products.get(mid);**

**else if (cmp < 0)**

**low = mid + 1;**

**else**

**high = mid - 1;**

**}**

**return null;**

**}**

**public static void main(String[] args) {**

**List<Product> products = new ArrayList<>();**

**products.add(new Product(101, "Mobile", "Electronics"));**

**products.add(new Product(102, "Earpods", "Electronics"));**

**products.add(new Product(103, "Heels", "Apparel"));**

**products.add(new Product(104, "FashWash", "Beauty"));**

**// Sorting for binary search**

**products.sort(Comparator.*comparing*(p -> p.getProductName().toLowerCase()));**

**String searchQuery = "Earpods";**

**// Linear Search**

**Product linearResult = *linearSearch*(products, searchQuery);**

**System.*out*.println("Linear Search Result: " + (linearResult != null ? linearResult : "Product not found"));**

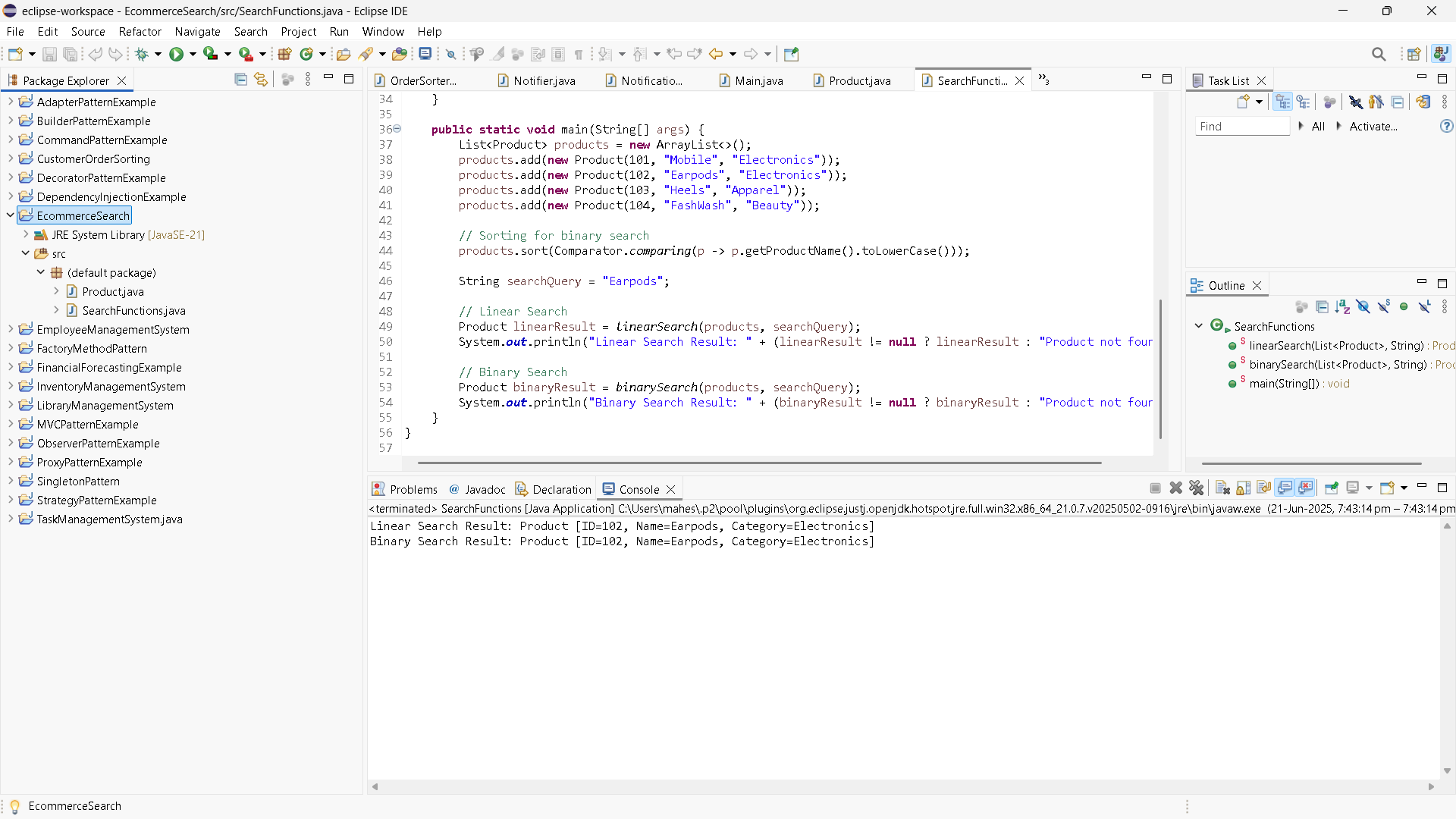
**// Binary Search**

**Product binaryResult = *binarySearch*(products, searchQuery);**

**System.*out*.println("Binary Search Result: " + (binaryResult != null ? binaryResult : "Product not found"));**

**}**

**}**



1. FinancialForecast.java

**public** **class** FinancialForecast {

**public** **static** **double** futureValue(**double** principal, **double** rate, **int** periods) {

**if** (periods == 0) {

**return** principal;

}

**return** *futureValue*(principal, rate, periods - 1) \* (1 + rate);

}

**public** **static** **void** main(String[] args) {

**double** presentValue = 10000.0; // Initial investment

**double** annualGrowthRate = 0.05; // 5% growth

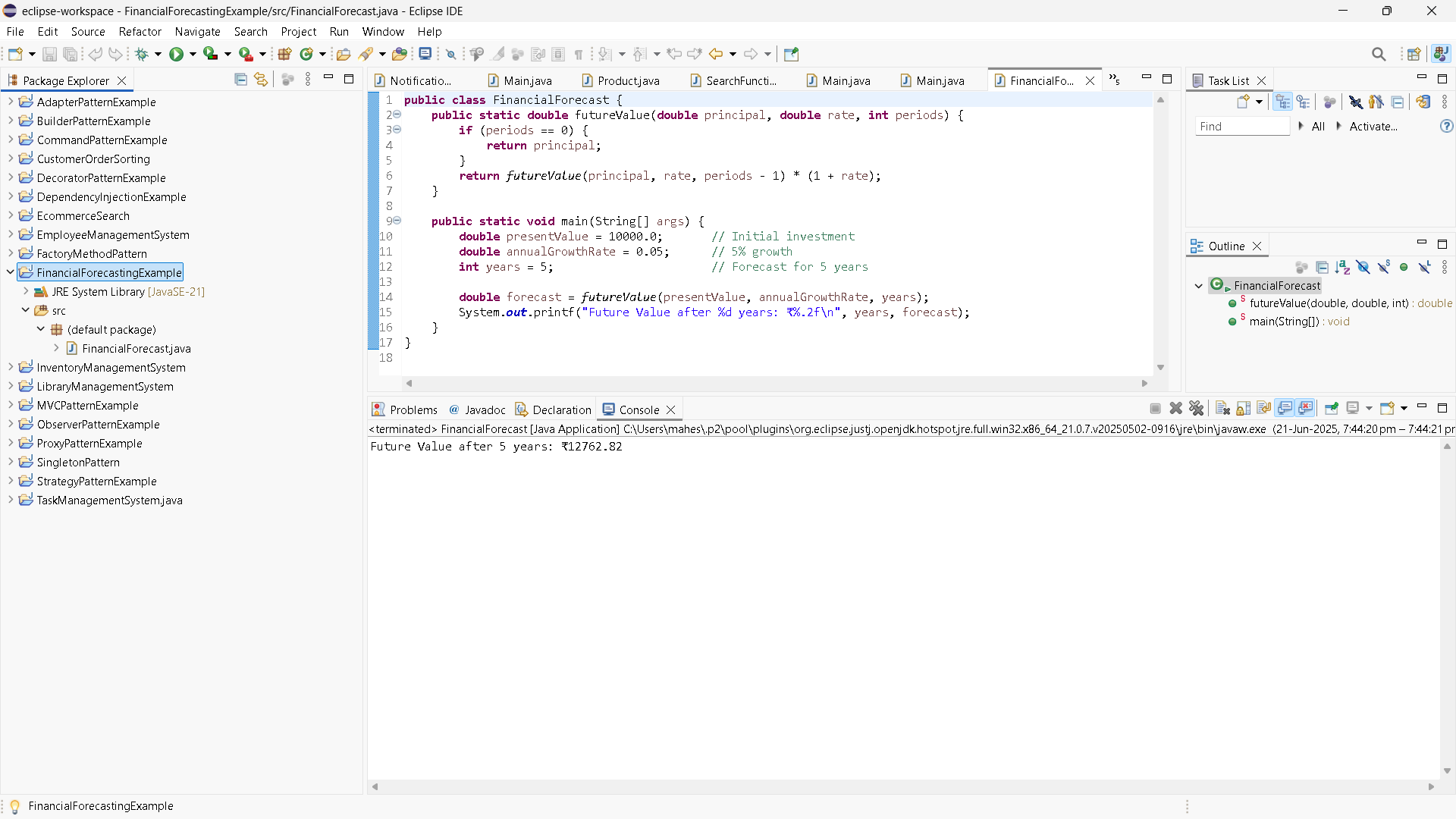
**int** years = 5; // Forecast for 5 years

**double** forecast = *futureValue*(presentValue, annualGrowthRate, years);

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

}

}



Practice problems

1. InventoryManagementSystem

AddProductService.java

**package** inventory;

**public** **class** AddProductService {

**private** InventoryManager manager;

**public** AddProductService(InventoryManager manager) {

**this**.manager = manager;

}

**public** **void** execute(Product product) {

manager.addProduct(product);

}

}

DeleteProductService.java

**package** inventory;

**public** **class** DeleteProductService {

**private** InventoryManager manager;

**public** DeleteProductService(InventoryManager manager) {

**this**.manager = manager;

}

**public** **void** execute(**int** productId) {

manager.deleteProduct(productId);

}

}

InventoryManager.java

**package** inventory;

**import** java.util.HashMap;

**public** **class** InventoryManager {

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

**public** **void** addProduct(Product product) {

**if** (inventory.containsKey(product.getProductId())) {

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

} **else** {

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

System.***out***.println("Product added.");

}

}

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

Product product = inventory.get(productId);

**if** (product != **null**) {

product.setQuantity(newQty);

product.setPrice(newPrice);

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 removed.");

} **else** {

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

}

}

**public** **void** displayInventory() {

**if** (inventory.isEmpty()) {

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

} **else** {

inventory.values().forEach(System.***out***::println);

}

}

}

Main.java

**package** inventory;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

InventoryManager manager = **new** InventoryManager();

// Create service instances

AddProductService addService = **new** AddProductService(manager);

UpdateProductService updateService = **new** UpdateProductService(manager);

DeleteProductService deleteService = **new** DeleteProductService(manager);

// Add products

addService.execute(**new** Product(101, "Laptop", 10, 50000));

addService.execute(**new** Product(102, "Mouse", 50, 500));

addService.execute(**new** Product(103, "Keyboard", 30, 1500));

System.***out***.println("\n--- Initial Inventory ---");

manager.displayInventory();

// Update product

updateService.execute(102, 60, 550);

// Delete product

deleteService.execute(103);

System.***out***.println("\n--- Updated Inventory ---");

manager.displayInventory();

}

}

Product.java

**package** inventory;

**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** setQuantity(**int** quantity) {

**this**.quantity = quantity;

}

**public** **void** setPrice(**double** price) {

**this**.price = price;

}

@Override

**public** String toString() {

**return** productId + " | " + productName + " | Qty: " + quantity + " | Rs. " + price;

}

}

UpdateProductService.java

**package** inventory;

**public** **class** UpdateProductService {

**private** InventoryManager manager;

**public** UpdateProductService(InventoryManager manager) {

**this**.manager = manager;

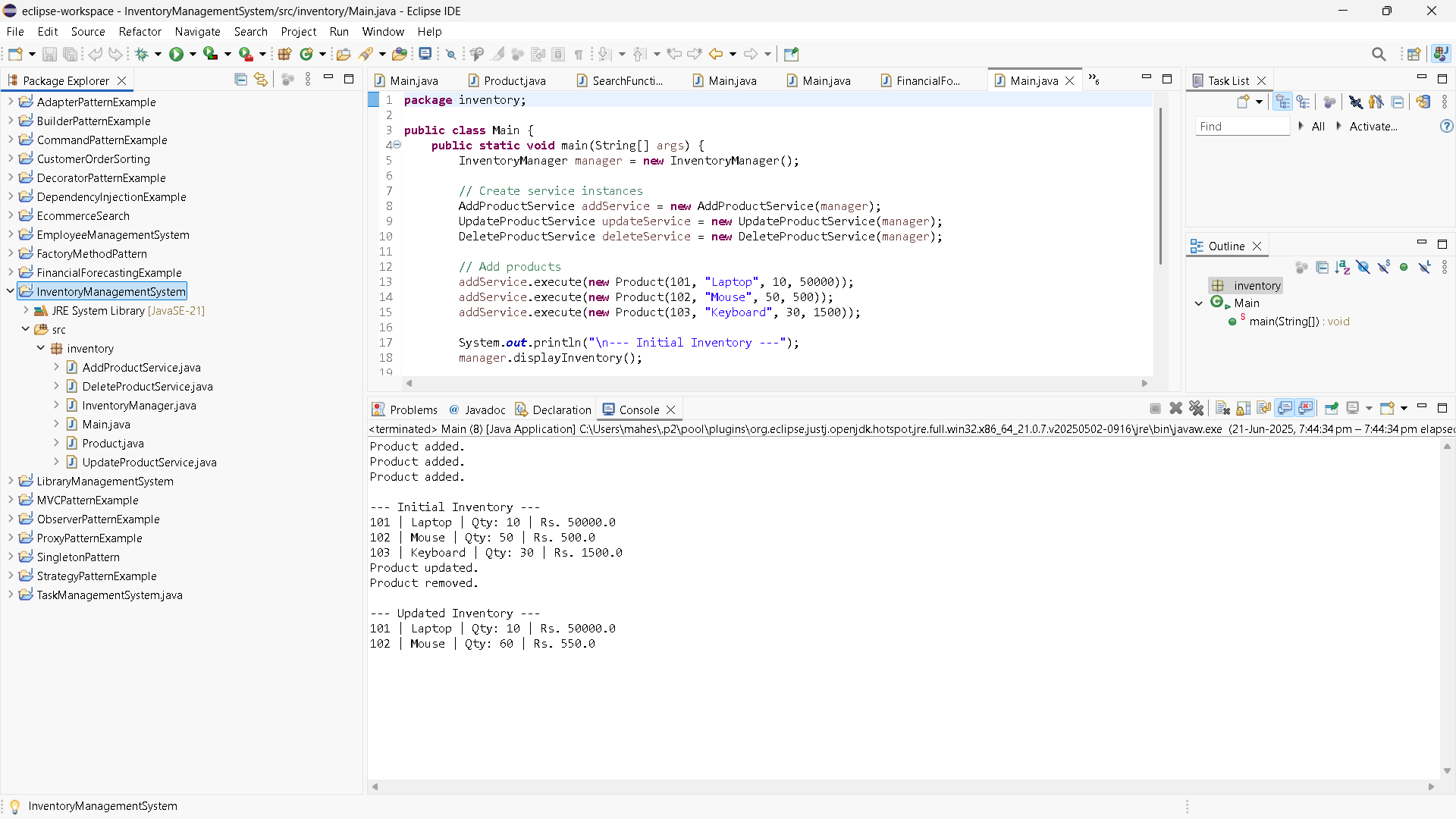
}

**public** **void** execute(**int** productId, **int** newQuantity, **double** newPrice) {

manager.updateProduct(productId, newQuantity, newPrice);

}

}



1. CustomerOrderSorting

BubbleSort.java

**public** **class** BubbleSort {

**public** **static** **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].totalPrice > orders[j + 1].totalPrice) {

Order temp = orders[j];

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

orders[j + 1] = temp;

}

}

}

}

}

Order.java

**public** **class** Order {

**int** orderId;

String customerName;

**double** totalPrice;

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

**this**.orderId = orderId;

**this**.customerName = customerName;

**this**.totalPrice = totalPrice;

}

**public** **void** display() {

System.***out***.println("Order ID: " + orderId + ", Customer: " + customerName + ", Total: RS. " + totalPrice);

}

}

OrderSorterMain.java

**public** **class** OrderSorterMain {

**public** **static** **void** main(String[] args) {

Order[] orders1 = {

**new** Order(101, "Yamini", 250.0),

**new** Order(102, "Priya", 100.0),

**new** Order(103, "Lasya", 400.0),

**new** Order(104, "Preeti", 150.0)

};

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

**for** (Order o : orders1) o.display();

System.***out***.println("\nSorted by Bubble Sort:");

BubbleSort.*sort*(orders1);

**for** (Order o : orders1) o.display();

Order[] orders2 = {

**new** Order(101, "Yamini", 250.0),

**new** Order(102, "Priya", 100.0),

**new** Order(103, "Lasya", 400.0),

**new** Order(104, "Preeti", 150.0)

};

System.***out***.println("\nSorted by Quick Sort:");

QuickSort.*sort*(orders2, 0, orders2.length - 1);

**for** (Order o : orders2) o.display();

}

}

QuickSort.java

**public** **class** QuickSort {

**public** **static** **void** sort(Order[] orders, **int** low, **int** high) {

**if** (low < high) {

**int** pi = *partition*(orders, low, high);

*sort*(orders, low, pi - 1);

*sort*(orders, pi + 1, high);

}

}

**private** **static** **int** partition(Order[] orders, **int** low, **int** high) {

**double** pivot = orders[high].totalPrice;

**int** i = low - 1;

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

**if** (orders[j].totalPrice <= pivot) {

i++;

Order temp = orders[i];

orders[i] = orders[j];

orders[j] = temp;

}

}

Order temp = orders[i + 1];

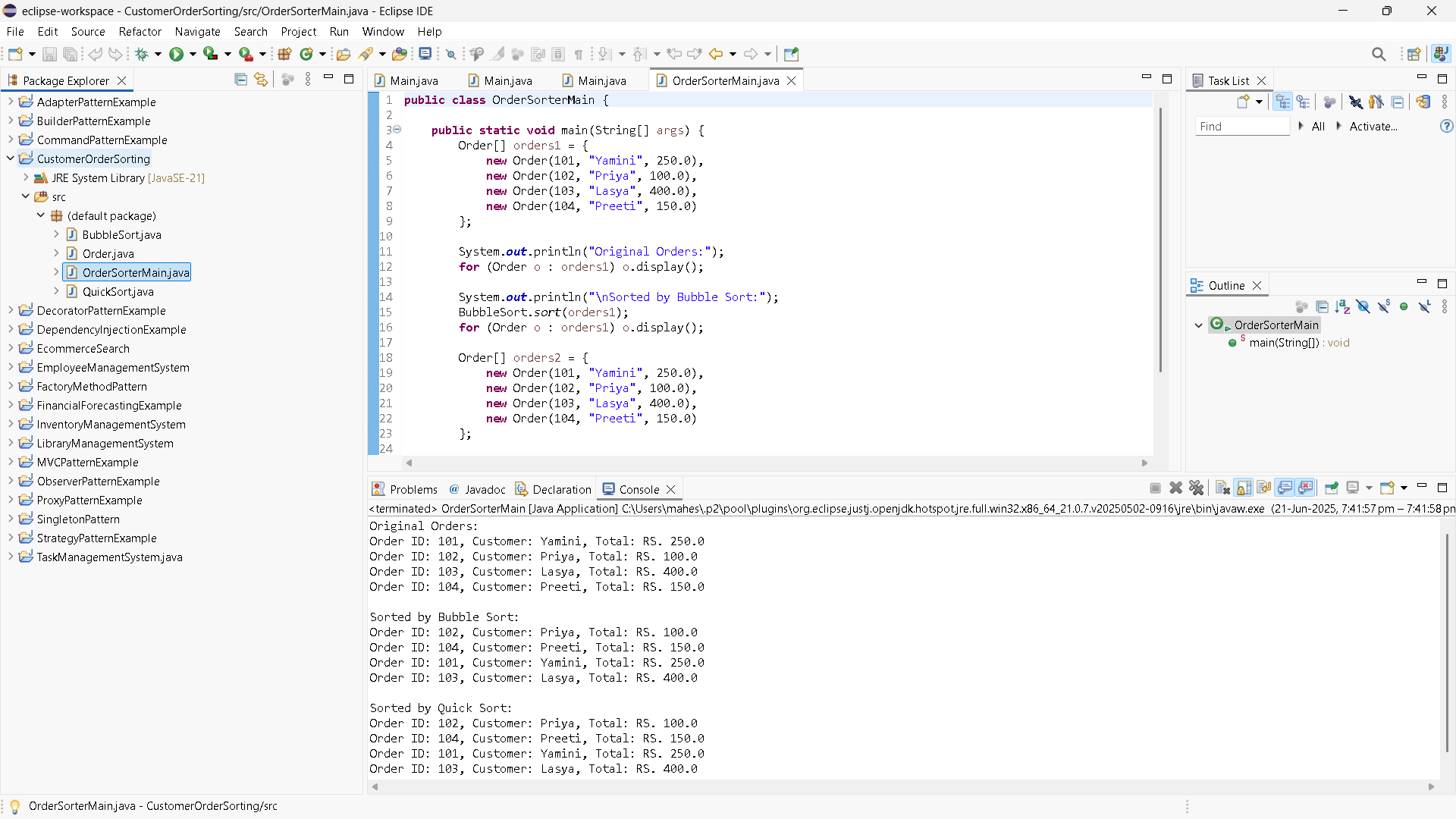
orders[i + 1] = orders[high];

orders[high] = temp;

**return** i + 1;

}

}



1. EmployeeManagementSystem

AddEmployee.java

**public** **class** AddEmployee {

**public** **static** **void** add(Employee[] employees, **int**[] count, Employee emp) {

**if** (count[0] < employees.length) {

employees[count[0]++] = emp;

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

} **else** {

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

}

}

}

DeleteEmployee.java

**public** **class** DeleteEmployee {

**public** **static** **void** delete(Employee[] employees, **int**[] count, **int** id) {

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

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

**for** (**int** j = i; j < count[0] - 1; j++) {

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

}

employees[--count[0]] = **null**;

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

**return**;

}

}

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

}

}

Employee.java

**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: $" + salary);

}

}

Main.java

**public** **class** Main {

**static** **final** **int** ***MAX*** = 100;

**static** Employee[] *employees* = **new** Employee[***MAX***];

**static** **int**[] *count* = {0}; // use array for pass-by-reference behavior

**public** **static** **void** displayAll() {

**if** (*count*[0] == 0) {

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

**return**;

}

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

*employees*[i].display();

}

}

**public** **static** **void** main(String[] args) {

AddEmployee.*add*(*employees*, *count*, **new** Employee(101, "Shalini", "Manager", 70000));

AddEmployee.*add*(*employees*, *count*, **new** Employee(102, "Rishitha", "Engineer", 50000));

AddEmployee.*add*(*employees*, *count*, **new** Employee(103, "Kumar", "HR", 45000));

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

*displayAll*();

System.***out***.println("\nSearching for ID 102:");

Employee found = SearchEmployee.*search*(*employees*, *count*[0], 102);

**if** (found != **null**) found.display();

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

System.***out***.println("\nUpdating ID 103:");

UpdateEmployee.*update*(*employees*, *count*[0], 103, "Charles", "HR Lead", 48000);

System.***out***.println("\nDeleting ID 101:");

DeleteEmployee.*delete*(*employees*, *count*, 101);

System.***out***.println("\nAll Employees After Update & Delete:");

*displayAll*();

}

}

SearchEmployee.java

**public** **class** SearchEmployee {

**public** **static** Employee search(Employee[] employees, **int** count, **int** id) {

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

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

**return** employees[i];

}

}

**return** **null**;

}

}

UpdateEmployee.java

**public** **class** UpdateEmployee {

**public** **static** **void** update(Employee[] employees, **int** count, **int** id, String newName, String newPosition, **double** newSalary) {

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

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

employees[i].name = newName;

employees[i].position = newPosition;

employees[i].salary = newSalary;

System.***out***.println("Employee updated.");

**return**;

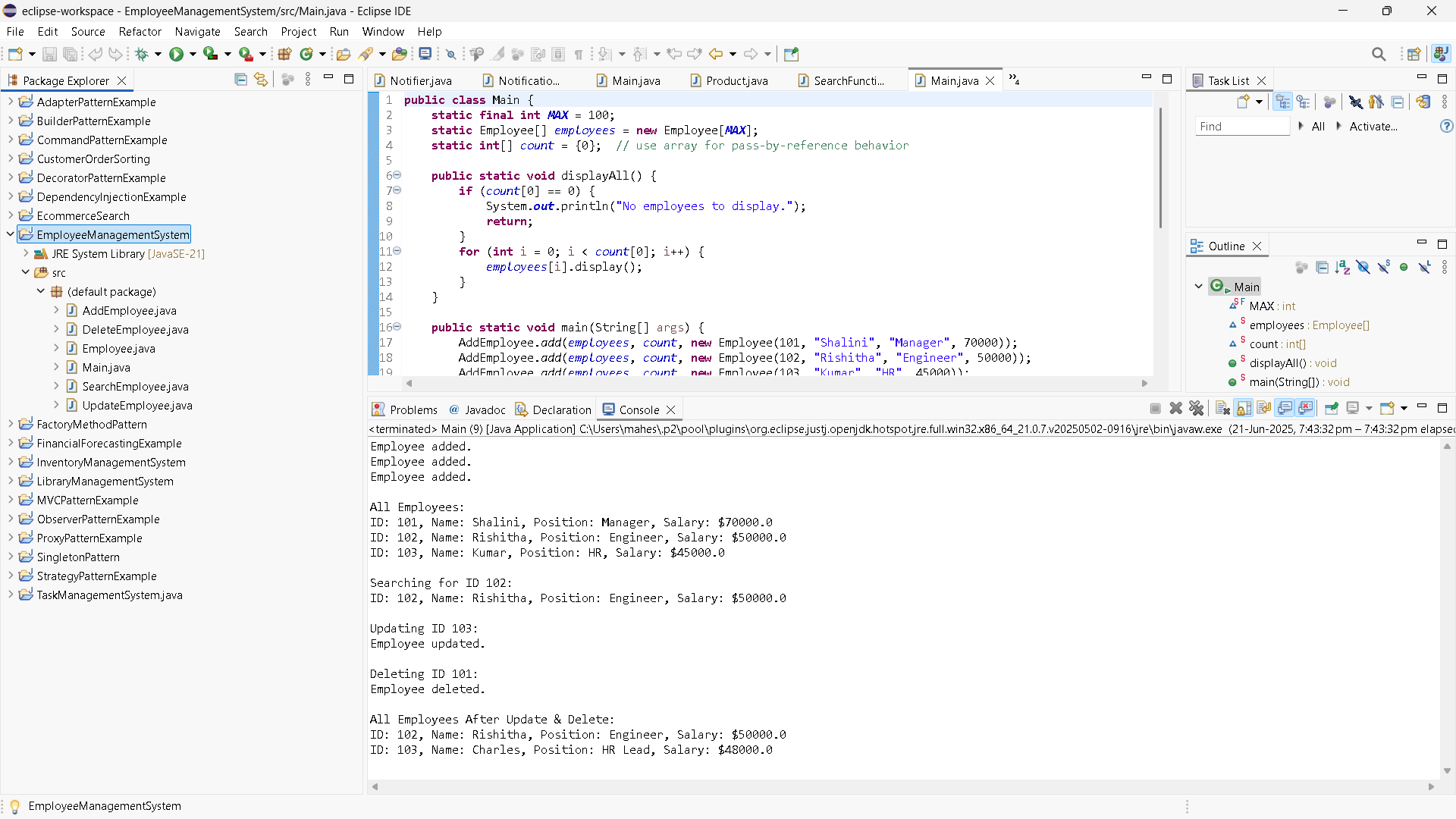
}

}

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

}

}



1. TaskManagementSystem

AddTask.java

**public** **class** AddTask {

**public** **static** **void** add(TaskList list, **int** taskId, String taskName, String status) {

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

**if** (list.head == **null**) {

list.head = newTask;

} **else** {

Task current = list.head;

**while** (current.next != **null**) {

current = current.next;

}

current.next = newTask;

}

System.***out***.println("Task added: " + taskName);

}

}

DeleteTask.java

**public** **class** DeleteTask {

**public** **static** **void** delete(TaskList list, **int** id) {

**if** (list.head == **null**) {

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

**return**;

}

**if** (list.head.taskId == id) {

list.head = list.head.next;

System.***out***.println("Task deleted.");

**return**;

}

Task current = list.head;

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

current = current.next;

}

**if** (current.next == **null**) {

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

} **else** {

current.next = current.next.next;

System.***out***.println("Task deleted.");

}

}

}

DisplayTask.java

**public** **class** DisplayTasks {

**public** **static** **void** display(TaskList list) {

**if** (list.head == **null**) {

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

**return**;

}

Task current = list.head;

**while** (current != **null**) {

current.display();

current = current.next;

}

}

}

SearchTassk.java

**public** **class** SearchTask {

**public** **static** Task search(TaskList list, **int** id) {

Task current = list.head;

**while** (current != **null**) {

**if** (current.taskId == id) {

**return** current;

}

current = current.next;

}

**return** **null**;

}

}

Task.java

**public** **class** Task {

**int** taskId;

String taskName;

String status;

Task next;

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

**this**.taskId = taskId;

**this**.taskName = taskName;

**this**.status = status;

**this**.next = **null**;

}

**public** **void** display() {

System.***out***.println("Task ID: " + taskId + ", Name: " + taskName + ", Status: " + status);

}

}

TaskList.java

**public** **class** TaskList {

**public** Task head = **null**;

}

TaskManagerMain.java

**public** **class** TaskManagerMain {

**public** **static** **void** main(String[] args) {

TaskList list = **new** TaskList();

AddTask.*add*(list, 1, "Design Database", "Pending");

AddTask.*add*(list, 2, "Develop API", "In Progress");

AddTask.*add*(list, 3, "UI Integration", "Not Started");

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

DisplayTasks.*display*(list);

System.***out***.println("\nSearching for Task ID 2:");

Task found = SearchTask.*search*(list, 2);

**if** (found != **null**) found.display();

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

System.***out***.println("\nDeleting Task ID 1:");

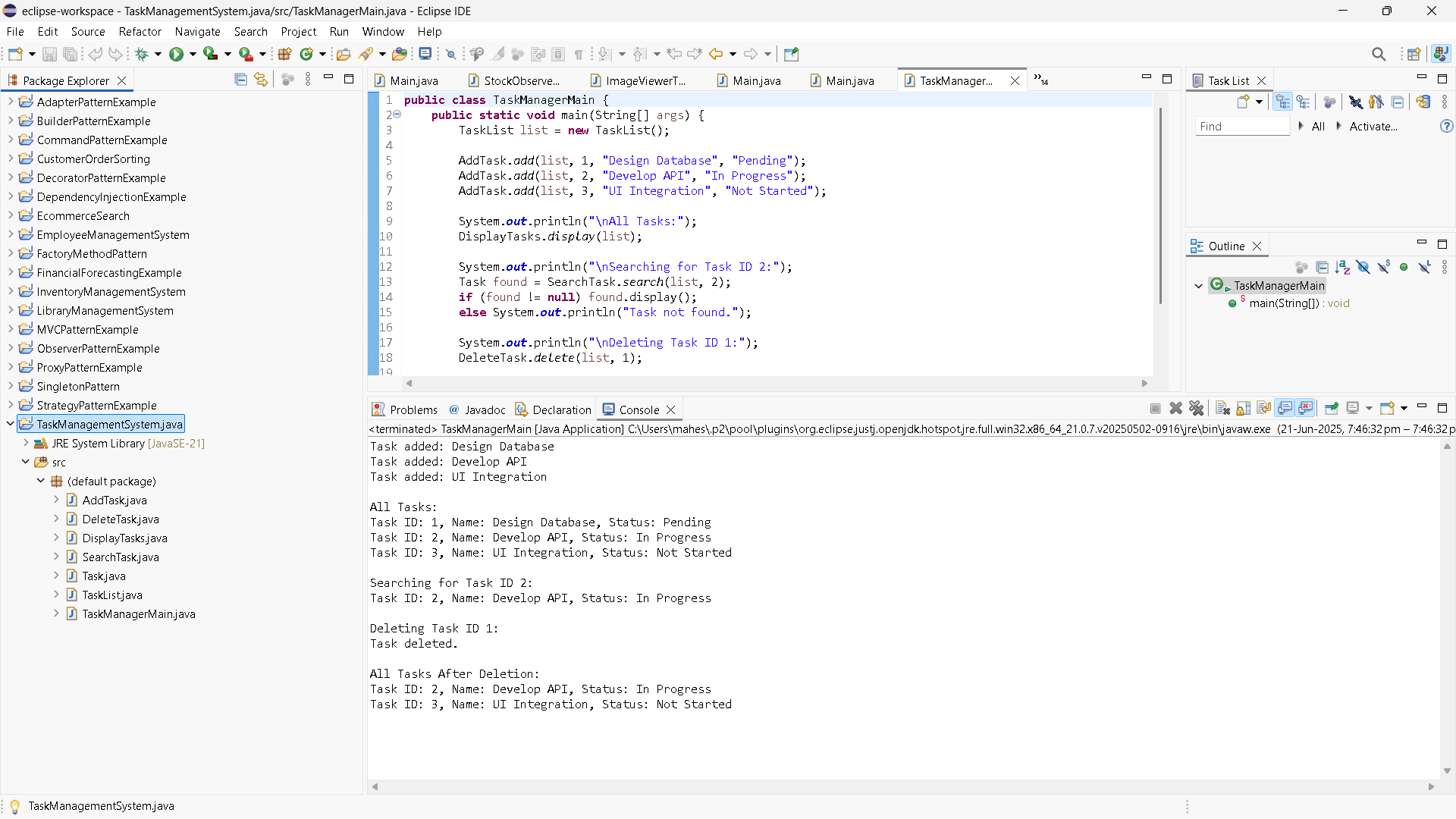
DeleteTask.*delete*(list, 1);

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

DisplayTasks.*display*(list);

}

}



6. LibraryManagementSystem

BinarySearch.java

**import** java.util.Arrays;

**import** java.util.Comparator;

**public** **class** BinarySearch {

// Sort books by title

**public** **static** **void** sortBooks(Book[] books) {

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

}

// Binary search by title

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

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

**while** (left <= right) {

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

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

**if** (cmp == 0) **return** books[mid];

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

**else** right = mid - 1;

}

**return** **null**;

}

}

Book.java

**import** java.util.Arrays;

**import** java.util.Comparator;

**public** **class** BinarySearch {

// Sort books by title

**public** **static** **void** sortBooks(Book[] books) {

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

}

// Binary search by title

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

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

**while** (left <= right) {

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

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

**if** (cmp == 0) **return** books[mid];

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

**else** right = mid - 1;

}

**return** **null**;

}

}

LibraryMain.java

**public** **class** LibraryMain {

**public** **static** **void** main(String[] args) {

Book[] books = {

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

**new** Book(102, "Atomic Habits", "James Clear"),

**new** Book(103, "Clean Code", "Robert C. Martin"),

**new** Book(104, "Deep Work", "Cal Newport"),

**new** Book(105, "Thinking Fast and Slow", "Daniel Kahneman")

};

System.***out***.println("Linear Search for 'Clean Code':");

Book found1 = LinearSearch.*search*(books, "Clean Code");

**if** (found1 != **null**) found1.display();

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

System.***out***.println("\n Sorting books for Binary Search...");

BinarySearch.*sortBooks*(books);

System.***out***.println(" Binary Search for 'Atomic Habits':");

Book found2 = BinarySearch.*search*(books, "Atomic Habits");

**if** (found2 != **null**) found2.display();

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

}

}

LinearSearch.java

**public** **class** LinearSearch {

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

**for** (Book book : books) {

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

**return** book;

}

}

**return** **null**;

}

}

