**Cognizant\_Digital Nurture 4.0\_Deep skilling**

**Engineering concepts**

**Module 2 - Data Structures and Algorithms**

**Exercise 1**

**Project: InventoryManagementSystem  
Package: inventory**

**Product.java**

package inventory;

public class Product {

private String productId;

private String productName;

private int quantity;

private 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 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 "Product{" +

"ID='" + productId + '\'' +

", Name='" + productName + '\'' +

", Qty=" + quantity +

", Price=" + price +

'}';

}

}

**InventoryManager.java**

package inventory;

import java.util.HashMap;

import java.util.Map;

public class InventoryManager {

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

public void addProduct(Product product) {

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

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

} else {

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

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

}

}

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

Product p = inventory.get(productId);

if (p != null) {

p.setProductName(name);

p.setQuantity(quantity);

p.setPrice(price);

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

} else {

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

}

}

public void deleteProduct(String productId) {

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

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

} else {

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

}

}

public void displayAllProducts() {

if (inventory.isEmpty()) {

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

} else {

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

System.out.println(product);

}

}

}

}

**InventoryTest.java**

package inventory;

public class InventoryTest {

public static void main(String[] args) {

InventoryManager manager = new InventoryManager();

manager.addProduct(new Product("P001", "Laptop", 10, 75000.00));

manager.addProduct(new Product("P002", "Mouse", 50, 500.00));

manager.displayAllProducts();

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

manager.updateProduct("P001", "Laptop Pro", 8, 85000.00);

manager.deleteProduct("P002");

manager.displayAllProducts();

}

}

**Output:**

Product added successfully.

Product added successfully.

Product{ID='P001', Name='Laptop', Qty=10, Price=75000.0}

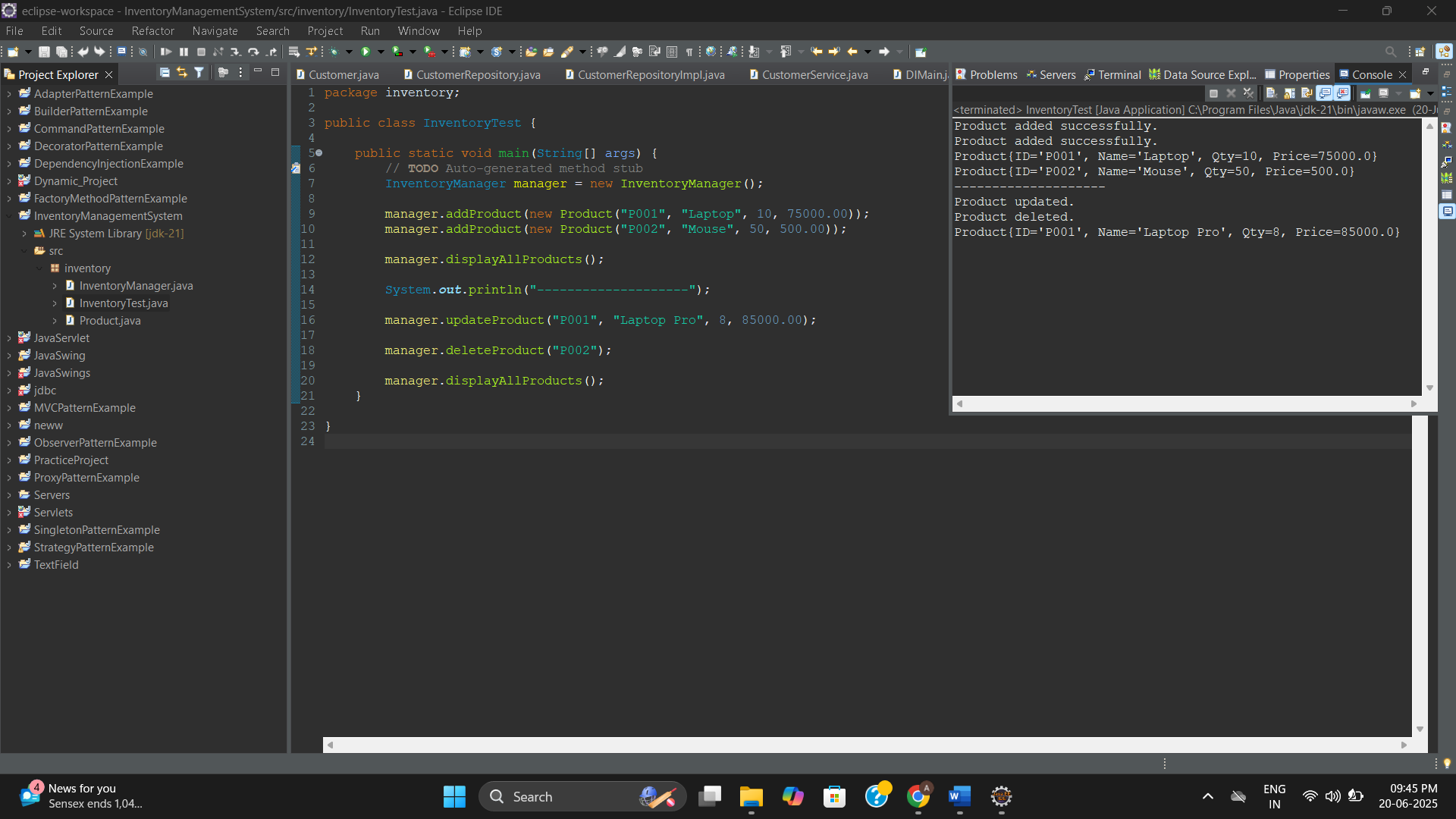
Product{ID='P002', Name='Mouse', Qty=50, Price=500.0}

--------------------

Product updated.

Product deleted.

Product{ID='P001', Name='Laptop Pro', Qty=8, Price=85000.0}

****

**Exercise 2**

**Project: EcommerceSearchExample**

**Package: ecommerce**

**Product.java**

package ecommerce;

public class Product {

private String productId;

private String productName;

private String category;

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

this.productId = productId;

this.productName = productName;

this.category = category;

}

public String getProductId() { return productId; }

public String getProductName() { return productName; }

public String getCategory() { return category; }

@Override

public String toString() {

return "Product{" +

"ID='" + productId + '\'' +

", Name='" + productName + '\'' +

", Category='" + category + '\'' +

'}';

}

}

**SearchAlgorithms.java**

package ecommerce;

import java.util.Arrays;

import java.util.Comparator;

public class SearchAlgorithms {

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

for (Product product : products) {

if (product.getProductName().equalsIgnoreCase(name)) {

return product;

}

}

return null;

}

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

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

while (left <= right) {

int mid = left + (right - left) / 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 sortProductsByName(Product[] products) {

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

}

}

**SearchTest.java**

package ecommerce;

public class SearchTest {

public static void main(String[] args) {

Product[] products = {

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

new Product("P002", "Shoes", "Footwear"),

new Product("P003", "Mobile", "Electronics"),

new Product("P004", "T-shirt", "Clothing")

};

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

Product found1 = SearchAlgorithms.linearSearch(products, "Mobile");

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

System.out.println("\nSorting for Binary Search...");

SearchAlgorithms.sortProductsByName(products);

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

Product found2 = SearchAlgorithms.binarySearch(products, "Mobile");

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

}

}

**Output:**

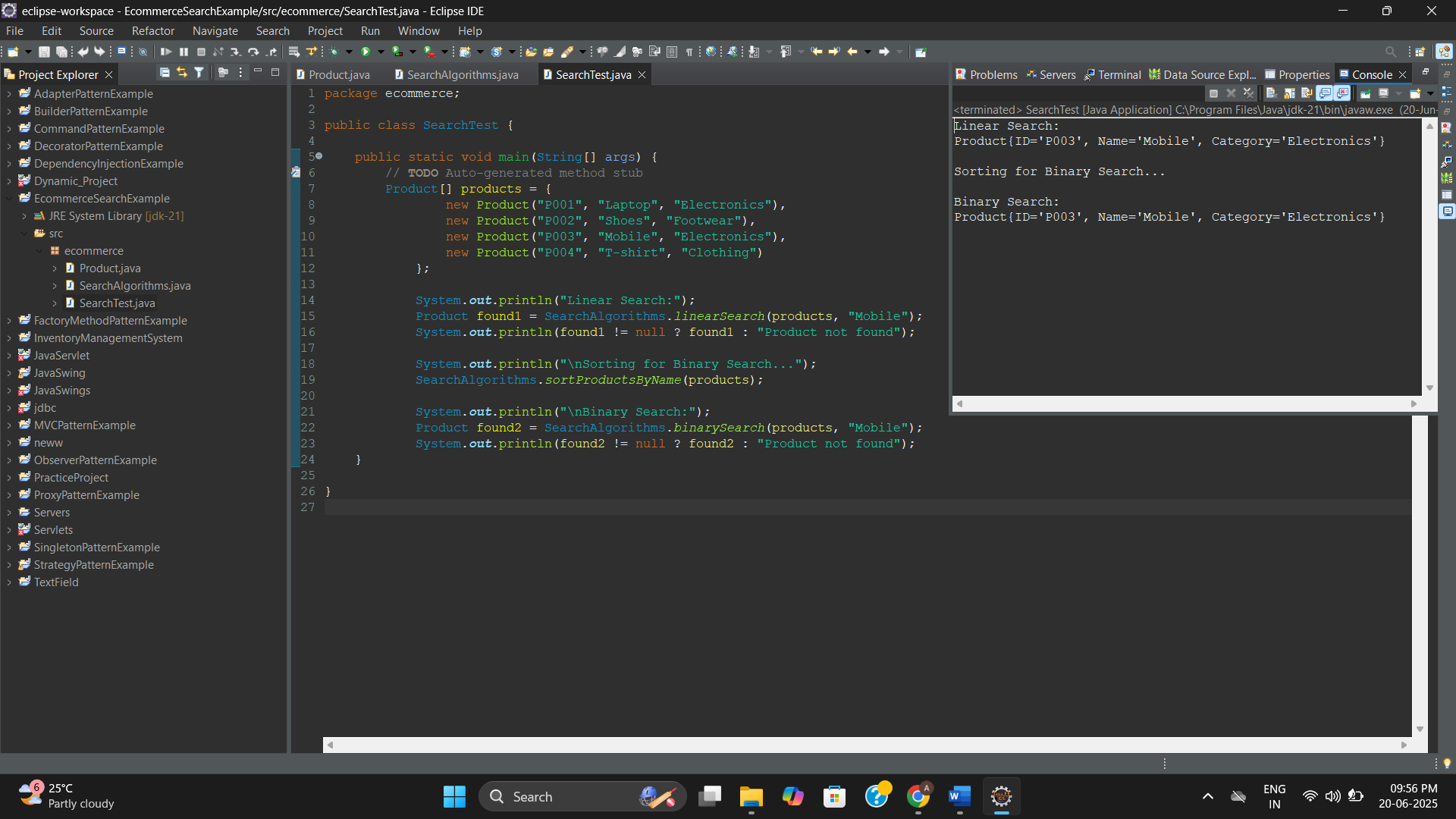
Linear Search:

Product{ID='P003', Name='Mobile', Category='Electronics'}

Sorting for Binary Search...

Binary Search:

Product{ID='P003', Name='Mobile', Category='Electronics'}

****

**Exercise 3**

**Project: CustomerOrderSorting**

**Package: ecommerce**

**Order.java**

package ecommerce;

public class Order {

private String orderId;

private String customerName;

private double totalPrice;

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

this.orderId = orderId;

this.customerName = customerName;

this.totalPrice = totalPrice;

}

public String getOrderId() { return orderId; }

public String getCustomerName() { return customerName; }

public double getTotalPrice() { return totalPrice; }

@Override

public String toString() {

return "Order{" +

"ID='" + orderId + '\'' +

", Customer='" + customerName + '\'' +

", Total=₹" + totalPrice +

'}';

}

}

**OrderSorter.java**

package ecommerce;

public class OrderSorter {

public static void bubbleSort(Order[] orders) {

int n = orders.length;

for (int i = 0; i < n-1; i++) {

for (int j = 0; j < n-1-i; j++) {

if (orders[j].getTotalPrice() > orders[j+1].getTotalPrice()) {

// Swap

Order temp = orders[j];

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

orders[j+1] = temp;

}

}

}

}

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) {

i++;

// Swap

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);

}

}

}

**OrderTest.java**

package ecommerce;

public class OrderTest {

public static void main(String[] args) {

// **TODO** Auto-generated method stub

Order[] orders = {

new Order("O001", "Ananth", 850.50),

new Order("O002", "Roopine", 3200.00),

new Order("O003", "Krish", 1500.00),

new Order("O004", "Goldbell", 999.99)

};

System.***out***.println("Bubble Sort by Total Price:");

OrderSorter.*bubbleSort*(orders);

OrderSorter.*displayOrders*(orders);

orders = new Order[]{

new Order("O001", "Ananth", 850.50),

new Order("O002", "Roopine", 3200.00),

new Order("O003", "Krish", 1500.00),

new Order("O004", "Goldbell", 999.99)

};

System.***out***.println("\nQuick Sort by Total Price:");

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

OrderSorter.*displayOrders*(orders);

}

}

**Output:**

Bubble Sort by Total Price:

Order{ID='O001', Customer='Ananth', Total=₹850.5}

Order{ID='O004', Customer='Goldbell', Total=₹999.99}

Order{ID='O003', Customer='Krish', Total=₹1500.0}

Order{ID='O002', Customer='Roopine', Total=₹3200.0}

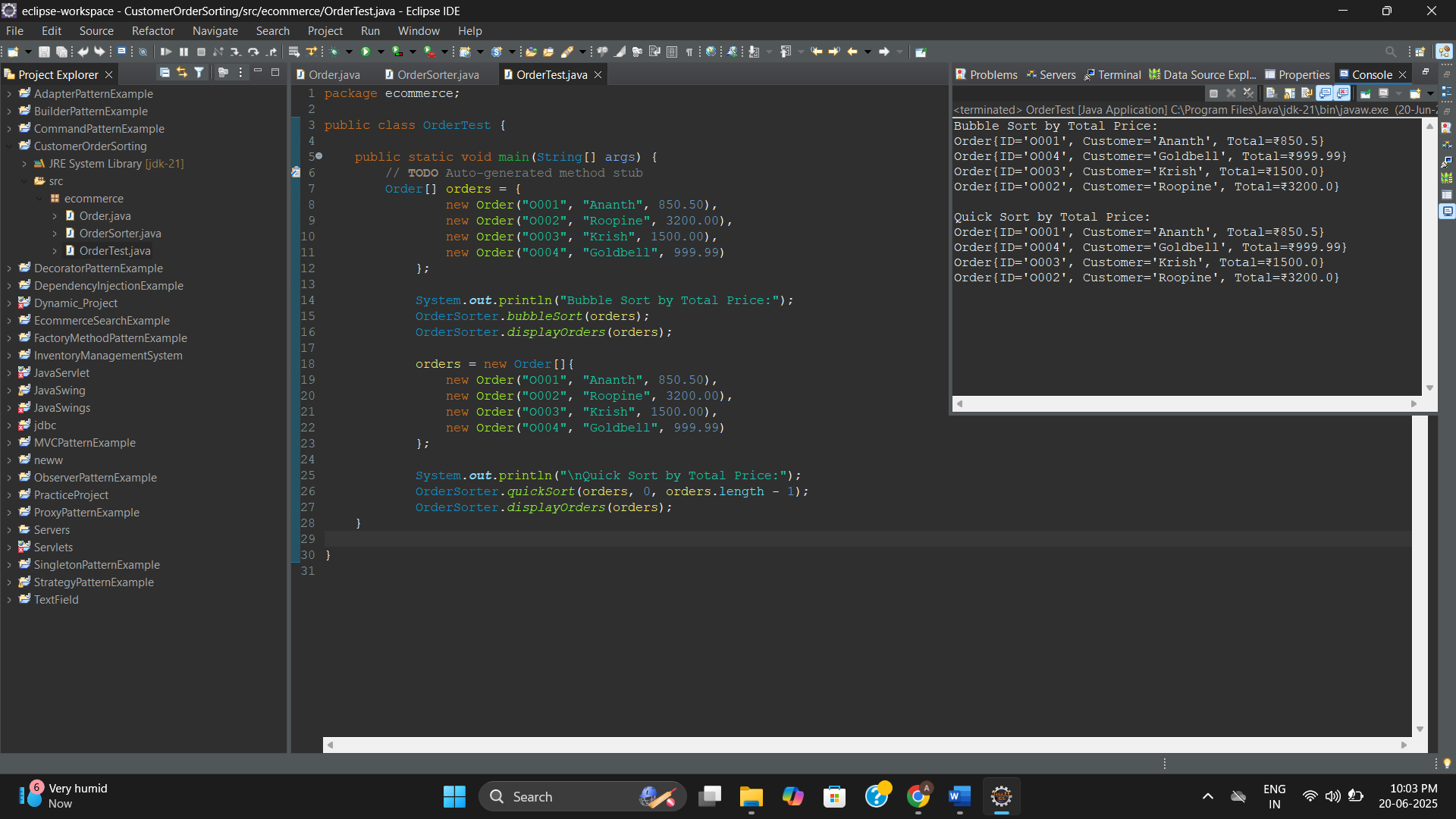
Quick Sort by Total Price:

Order{ID='O001', Customer='Ananth', Total=₹850.5}

Order{ID='O004', Customer='Goldbell', Total=₹999.99}

Order{ID='O003', Customer='Krish', Total=₹1500.0}

Order{ID='O002', Customer='Roopine', Total=₹3200.0}

****

**Exercise 4**

**Project: EmployeeManagementSystem**

**Package: employees**

**Employee.java**

package employeesystem;

public class Employee {

private String employeeId;

private String name;

private String position;

private double salary;

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

this.employeeId = employeeId;

this.name = name;

this.position = position;

this.salary = salary;

}

public String getEmployeeId() { return employeeId; }

public String getName() { return name; }

public String getPosition() { return position; }

public double getSalary() { return salary; }

@Override

public String toString() {

return "Employee{" +

"ID='" + employeeId + '\'' +

", Name='" + name + '\'' +

", Position='" + position + '\'' +

", Salary=₹" + salary +

'}';

}

}

**EmployeeManager.java**

package employeesystem;

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) {

employees[size++] = e;

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

} else {

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

}

}

public Employee searchEmployee(String empId) {

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

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

return employees[i];

}

}

return null;

}

public void deleteEmployee(String empId) {

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

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

// Shift remaining elements 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.");

}

public void displayAll() {

if (size == 0) {

System.out.println("No employees in system.");

return;

}

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

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

}

}

}

**EmployeeSystemTest.java**

package employees;

public class EmployeeSystemTest {

public static void main(String[] args) {

// **TODO** Auto-generated method stub

EmployeeManager manager = new EmployeeManager(5);

manager.addEmployee(new Employee("E001", "Ananth", "Developer", 50000));

manager.addEmployee(new Employee("E002", "Roopine", "Manager", 80000));

manager.addEmployee(new Employee("E003", "Krish", "Tester", 40000));

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

manager.displayAll();

System.***out***.println("\nSearching for E002:");

Employee found = manager.searchEmployee("E002");

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

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

manager.deleteEmployee("E001");

System.***out***.println("\nUpdated Employee List:");

manager.displayAll();

}

}

**Output:**

Employee added.

Employee added.

Employee added.

All Employees:

Employee{ID='E001', Name='Ananth', Position='Developer', Salary=₹50000.0}

Employee{ID='E002', Name='Roopine', Position='Manager', Salary=₹80000.0}

Employee{ID='E003', Name='Krish', Position='Tester', Salary=₹40000.0}

Searching for E002:

Employee{ID='E002', Name='Roopine', Position='Manager', Salary=₹80000.0}

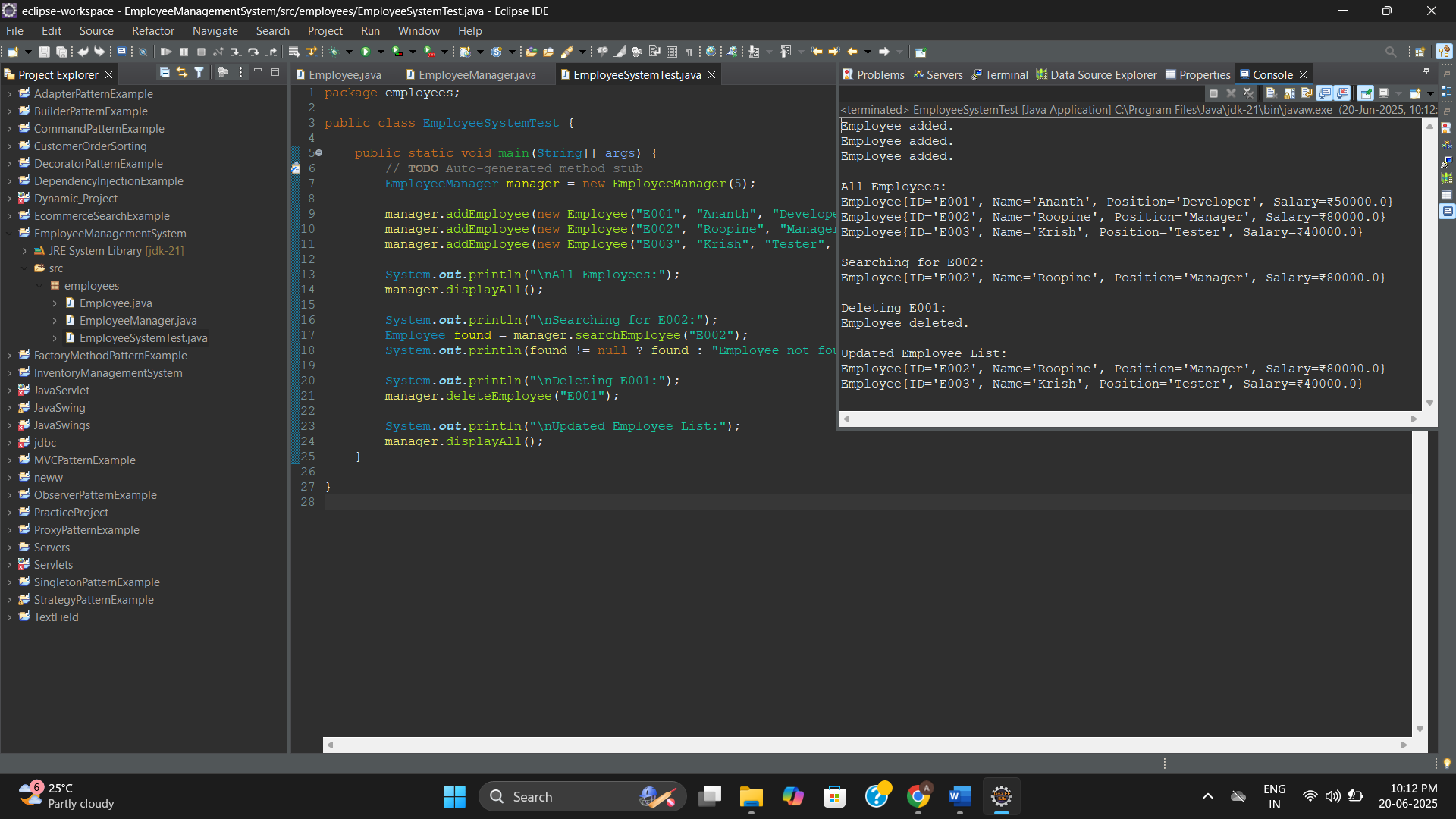
Deleting E001:

Employee deleted.

Updated Employee List:

Employee{ID='E002', Name='Roopine', Position='Manager', Salary=₹80000.0}

Employee{ID='E003', Name='Krish', Position='Tester', Salary=₹40000.0}

****

**Exercise 5**

**Project: TaskManagementSystem**

**Package: task**

**Task.java**

package taskmanager;

public class Task {

String taskId;

String taskName;

String status;

public Task(String 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 taskmanager;

public class TaskNode {

Task task;

TaskNode next;

public TaskNode(Task task) {

this.task = task;

this.next = null;

}

}

**TaskManager.java**

package taskmanager;

public class TaskManager {

private 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;

}

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

}

public Task searchTask(String taskId) {

TaskNode current = head;

while (current != null) {

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

return current.task;

}

current = current.next;

}

return null;

}

public void deleteTask(String taskId) {

if (head == null) return;

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

head = head.next;

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

return;

}

TaskNode current = head;

while (current.next != null) {

if (current.next.task.taskId.equalsIgnoreCase(taskId)) {

current.next = current.next.next;

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

return;

}

current = current.next;

}

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

}

public void displayAll() {

if (head == null) {

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

return;

}

TaskNode current = head;

while (current != null) {

System.out.println(current.task);

current = current.next;

}

}

}

**TaskManagerTest.java**

package taskmanager;

public class TaskManagerTest {

public static void main(String[] args) {

// **TODO** Auto-generated method stub

TaskManager manager = new TaskManager();

manager.addTask(new Task("T001", "Design Module", "Pending"));

manager.addTask(new Task("T002", "Code Module", "In Progress"));

manager.addTask(new Task("T003", "Test Module", "Pending"));

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

manager.displayAll();

System.out.println("\nSearching for T002:");

Task found = manager.searchTask("T002");

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

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

manager.deleteTask("T001");

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

manager.displayAll();

}

}

**Output:**

Task added.

Task added.

Task added.

All Tasks:

Task{ID='T001', Name='Design Module', Status='Pending'}

Task{ID='T002', Name='Code Module', Status='In Progress'}

Task{ID='T003', Name='Test Module', Status='Pending'}

Searching for T002:

Task{ID='T002', Name='Code Module', Status='In Progress'}

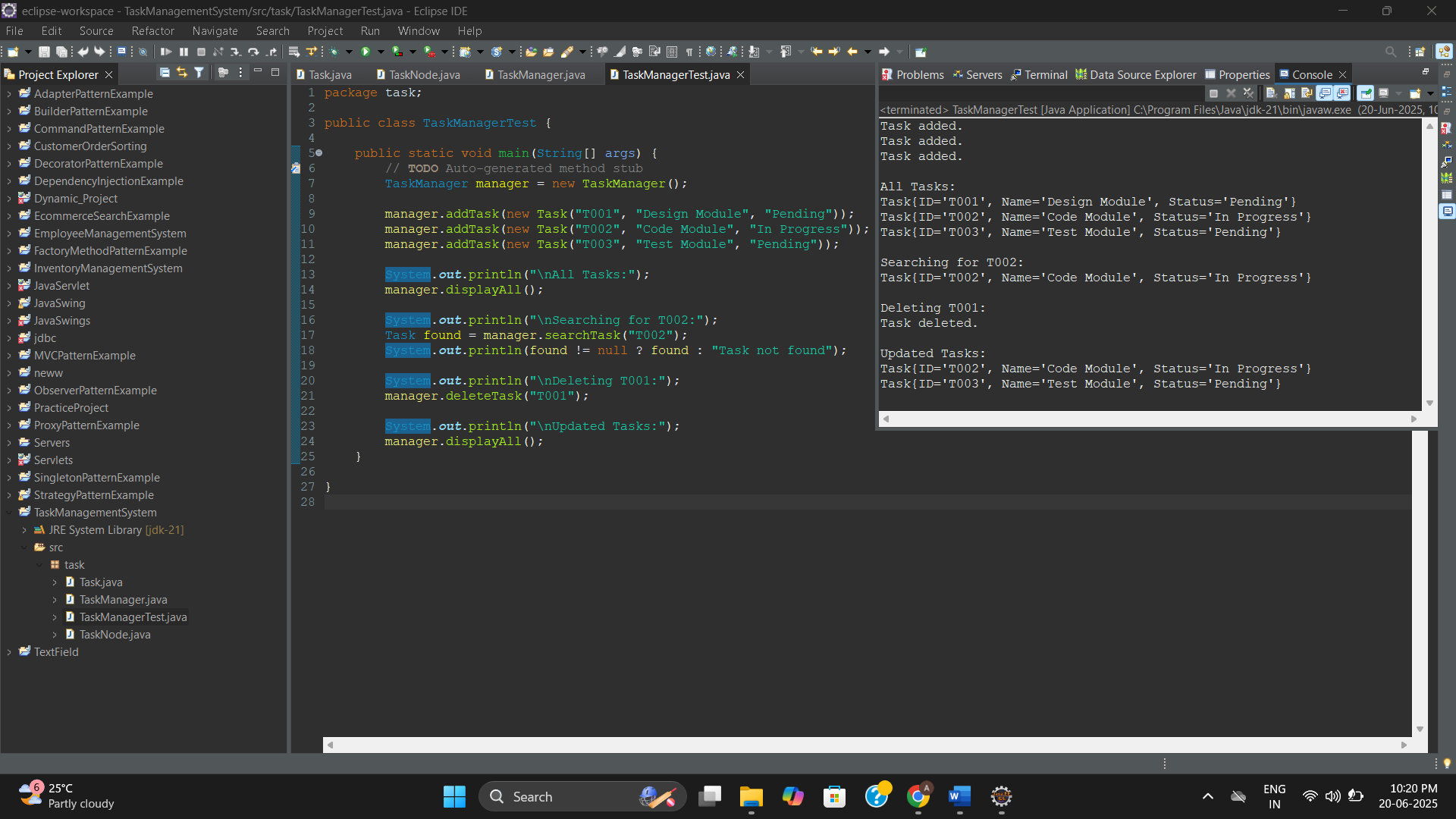
Deleting T001:

Task deleted.

Updated Tasks:

Task{ID='T002', Name='Code Module', Status='In Progress'}

Task{ID='T003', Name='Test Module', Status='Pending'}

****

**Exercise 6**

**Project: LibraryManagementSystem**

**Package: library**

**Book.java**

package library;

public class Book {

private String bookId;

private String title;

private String author;

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

this.bookId = bookId;

this.title = title;

this.author = author;

}

public String getBookId() { return bookId; }

public String getTitle() { return title; }

public String getAuthor() { return author; }

@Override

public String toString() {

return "Book{" +

"ID='" + bookId + '\'' +

", Title='" + title + '\'' +

", Author='" + author + '\'' +

'}';

}

}

**BookSearch.java**

package library;

import java.util.Arrays;

import java.util.Comparator;

public class BookSearch {

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

for (Book book : books) {

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

return book;

}

}

return null;

}

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

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

while (left <= right) {

int mid = (left + right) / 2;

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

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

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

else right = mid - 1;

}

return null;

}

public static void sortBooks(Book[] books) {

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

}

public static void displayBooks(Book[] books) {

for (Book book : books) {

System.out.println(book);

}

}

}

**LibraryTest.java**

package library;

public class LibraryTest {

public static void main(String[] args) {

// **TODO** Auto-generated method stub

Book[] books = {

new Book("B001", "The Alchemist", "Paulo Coelho"),

new Book("B002", "Atomic Habits", "James Clear"),

new Book("B003", "Wings of Fire", "A.P.J Abdul Kalam"),

new Book("B004", "Clean Code", "Robert C. Martin")

};

System.***out***.println("Linear Search:");

Book found1 = BookSearch.*linearSearch*(books, "Atomic Habits");

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

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

BookSearch.*sortBooks*(books);

System.***out***.println("\nBinary Search:");

Book found2 = BookSearch.*binarySearch*(books, "Wings of Fire");

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

}

}

**Output:**

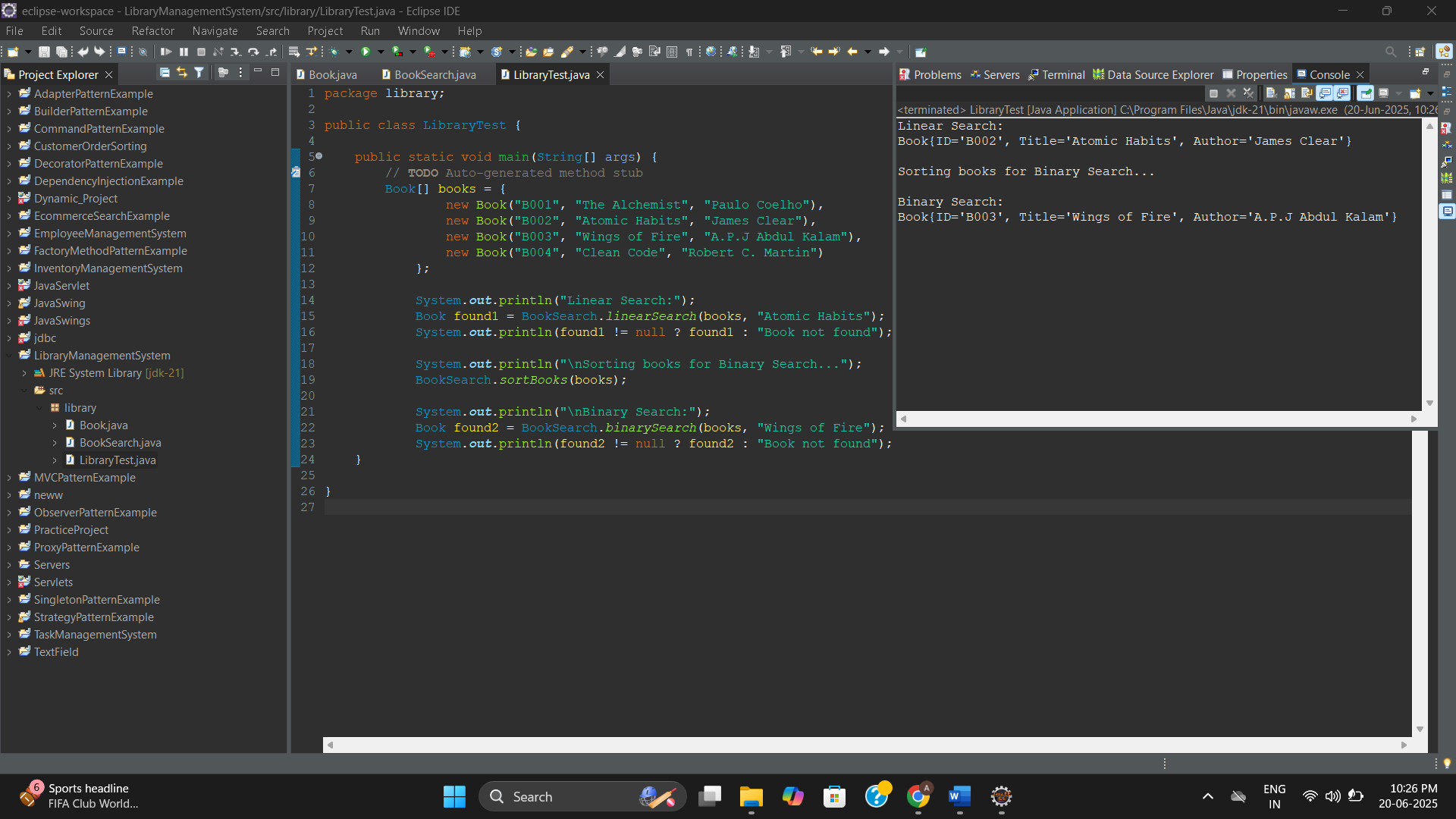
Linear Search:

Book{ID='B002', Title='Atomic Habits', Author='James Clear'}

Sorting books for Binary Search...

Binary Search:

Book{ID='B003', Title='Wings of Fire', Author='A.P.J Abdul Kalam'}

****

**Exercise 7**

**Project: FinancialForecasting**

**Package: finance**

**FinancialForecast.java**

package finance;

public class FinancialForecast {

// Recursive method to calculate future value

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

if (years == 0) return presentValue;

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

}

// Optimized version using Memoization (if needed)

public static double calculateFutureValueDP(double presentValue, double rate, int years, Double[] memo) {

if (years == 0) return presentValue;

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

memo[years] = (1 + rate) \* *calculateFutureValueDP*(presentValue, rate, years - 1, memo);

return memo[years];

}

}

**ForecastTest.java**

package finance;

public class ForecastTest {

public static void main(String[] args) {

// **TODO** Auto-generated method stub

double presentValue = 10000;

double annualRate = 0.05; // 5%

int years = 5;

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

double futureValue = FinancialForecast.*calculateFutureValue*(presentValue, annualRate, years);

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

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

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

double futureValueMemo = FinancialForecast.*calculateFutureValueDP*(presentValue, annualRate, years, memo);

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

}

}

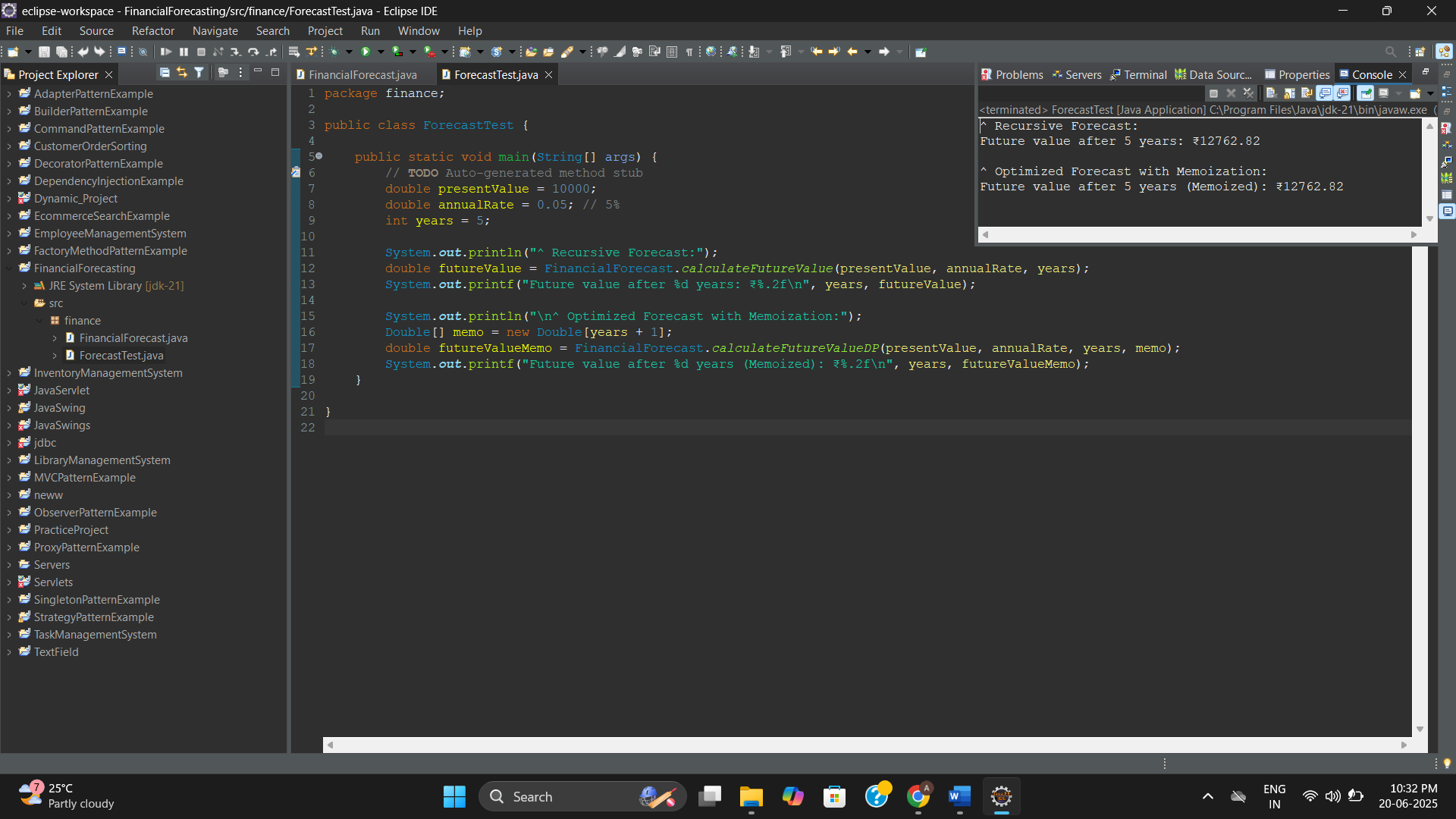
**Output:**

Recursive Forecast:

Future value after 5 years: ₹12762.82

Optimized Forecast with Memoization:

Future value after 5 years (Memoized): ₹12762.82

****