**1.INVENTORY MANAGEMENT SYSTEM**  
  
  
  
import java.util.HashMap;

import java.util.Scanner;

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 "ID: " + productId + ", Name: " + productName + ", Qty: " + quantity + ", Price: " + price;

}

}

public class InventorySystem {

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

public static void addProduct(Product product) {

inventory.put(product.productId, product);

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

}

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

if (inventory.containsKey(productId)) {

Product p = inventory.get(productId);

p.quantity = newQty;

p.price = newPrice;

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

} else {

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

}

}

public static void deleteProduct(String productId) {

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

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

} else {

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

}

}

public static void displayInventory() {

if (inventory.isEmpty()) {

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

return;

}

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

System.out.println(p);

}

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

while (true) {

System.out.println(" 1.Add 2.Update 3.Delete 4.Display 5.Exit ");

int choice = sc.nextInt();

sc.nextLine();

switch (choice) {

case 1:

System.out.print("Product ID: ");

String id = sc.nextLine();

System.out.print("Name: ");

String name = sc.nextLine();

System.out.print("Quantity: ");

int qty = sc.nextInt();

System.out.print("Price: ");

double price = sc.nextDouble();

addProduct(new Product(id, name, qty, price));

break;

case 2:

System.out.print("Enter Product ID to update: ");

String upId = sc.nextLine();

System.out.print("New Quantity: ");

int upQty = sc.nextInt();

System.out.print("New Price: ");

double upPrice = sc.nextDouble();

updateProduct(upId, upQty, upPrice);

break;

case 3:

System.out.print("Enter Product ID to delete: ");

String delId = sc.nextLine();

deleteProduct(delId);

break;

case 4:

displayInventory();

break;

case 5:

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

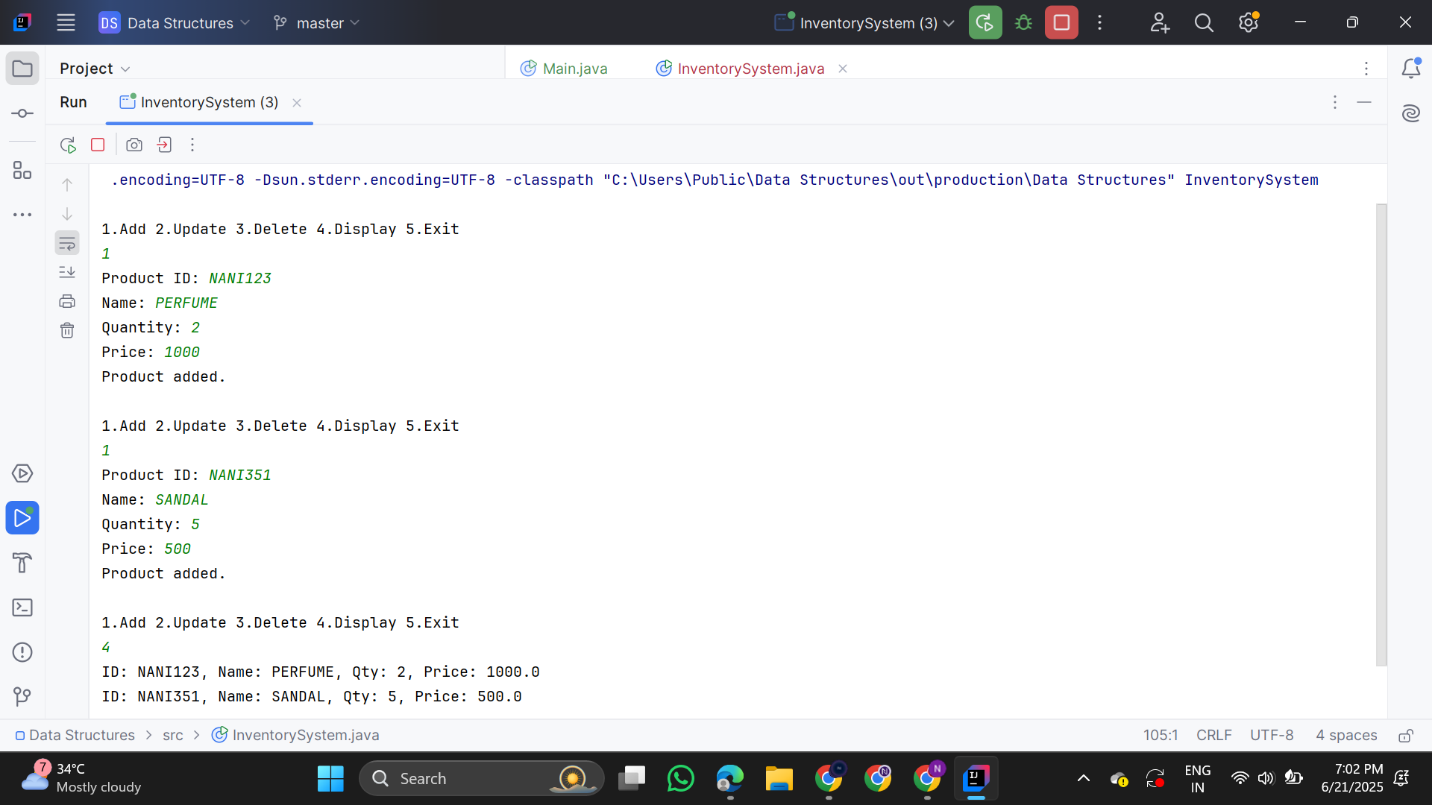
return;

}

}

}

}



**2. E-commerce Platform Search Function**

import java.util.\*;

class Item {

String productId, productName, category;

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

this.productId = productId;

this.productName = productName;

this.category = category;

}

public String toString() {

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

}

}

public class ECommerceSearch {

public static Item linearSearch(Item[] items, String targetName) {

for (Item i : items) {

if (i.productName.equalsIgnoreCase(targetName)) return i;

}

return null;

}

public static Item binarySearch(Item[] items, String targetName) {

Arrays.sort(items, Comparator.comparing(i -> i.productName.toLowerCase()));

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

while (left <= right) {

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

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

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

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

else right = mid - 1;

}

return null;

}

public static void main(String[] args) {

Item[] items = {

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

new Item("P002", "Shoes", "Fashion"),

new Item("P003", "Book", "Education"),

new Item("P004", "Smartphone", "Electronics"),

new Item("P005", "Pen", "Stationery")

};

Scanner sc = new Scanner(System.in);

System.out.print("Enter product name to search: ");

String input = sc.nextLine();

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

Item linearResult = linearSearch(items, input);

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

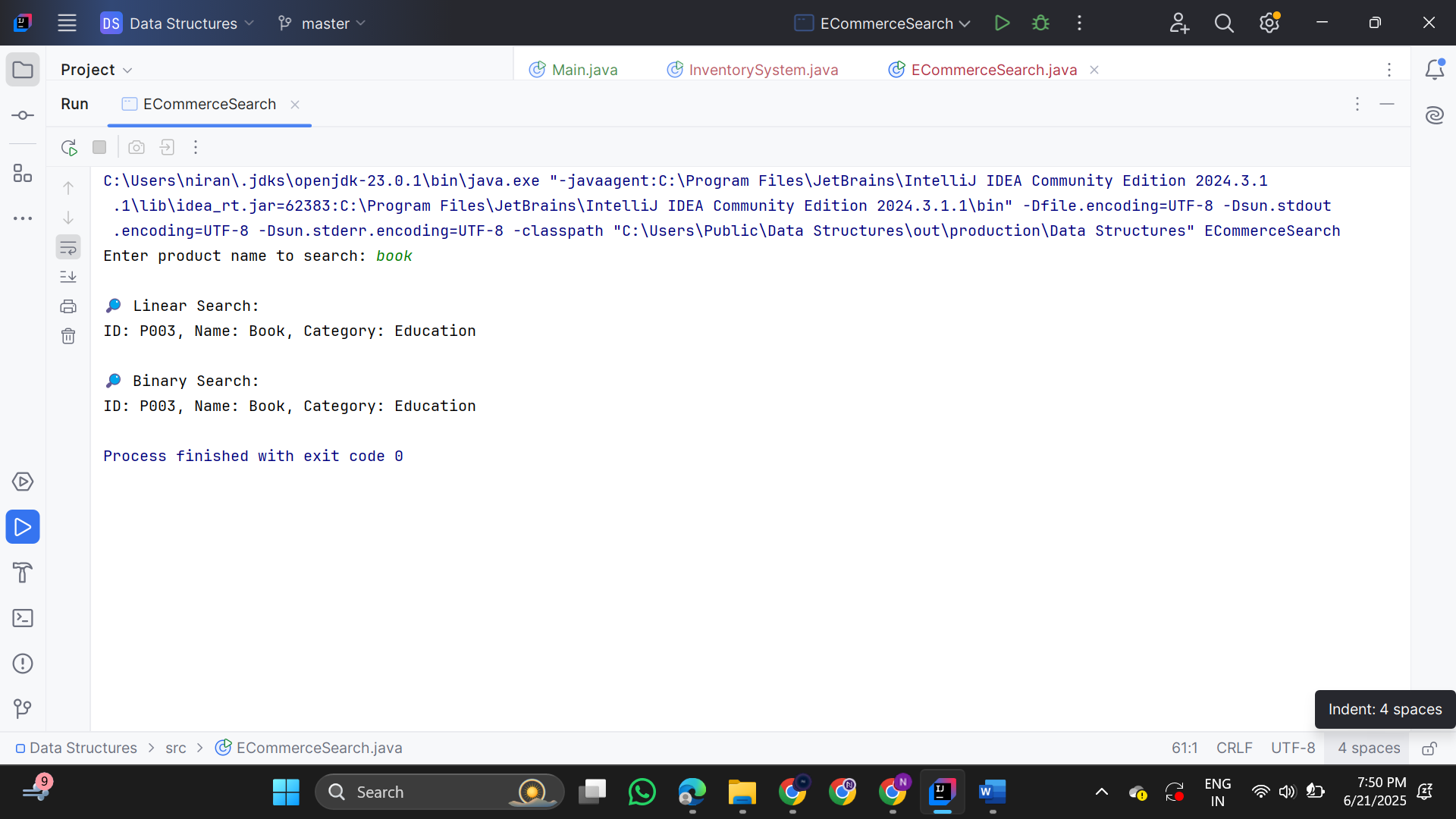
System.out.println(" Binary Search: ");

Item binaryResult = binarySearch(items, input);

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

}

}



**3.Sorting Customer Orders**import java.util.\*;

class Order {

String orderId;

String customerName;

double totalPrice;

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

this.orderId = orderId;

this.customerName = customerName;

this.totalPrice = totalPrice;

}

public String toString() {

return "OrderID: " + orderId + ", Customer: " + customerName + ", Total: ₹" + totalPrice;

}

}

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 - i - 1; j++) {

if (orders[j].totalPrice < orders[j + 1].totalPrice) {

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);

}

}

public 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;

}

public static void printOrders(Order[] orders) {

for (Order o : orders) {

System.out.println(o);

}

}

public static void main(String[] args) {

Order[] orders = {

new Order("O001", "Alice", 1500.0),

new Order("O002", "Bob", 3000.0),

new Order("O003", "Charlie", 2000.0),

new Order("O004", "David", 500.0),

new Order("O005", "Eve", 4000.0)

};

System.out.println(" Bubble Sort (by totalPrice):");

bubbleSort(orders);

printOrders(orders);

// Resetting orders for next test

orders = new Order[]{

new Order("O001", "Alice", 1500.0),

new Order("O002", "Bob", 3000.0),

new Order("O003", "Charlie", 2000.0),

new Order("O004", "David", 500.0),

new Order("O005", "Eve", 4000.0)

};

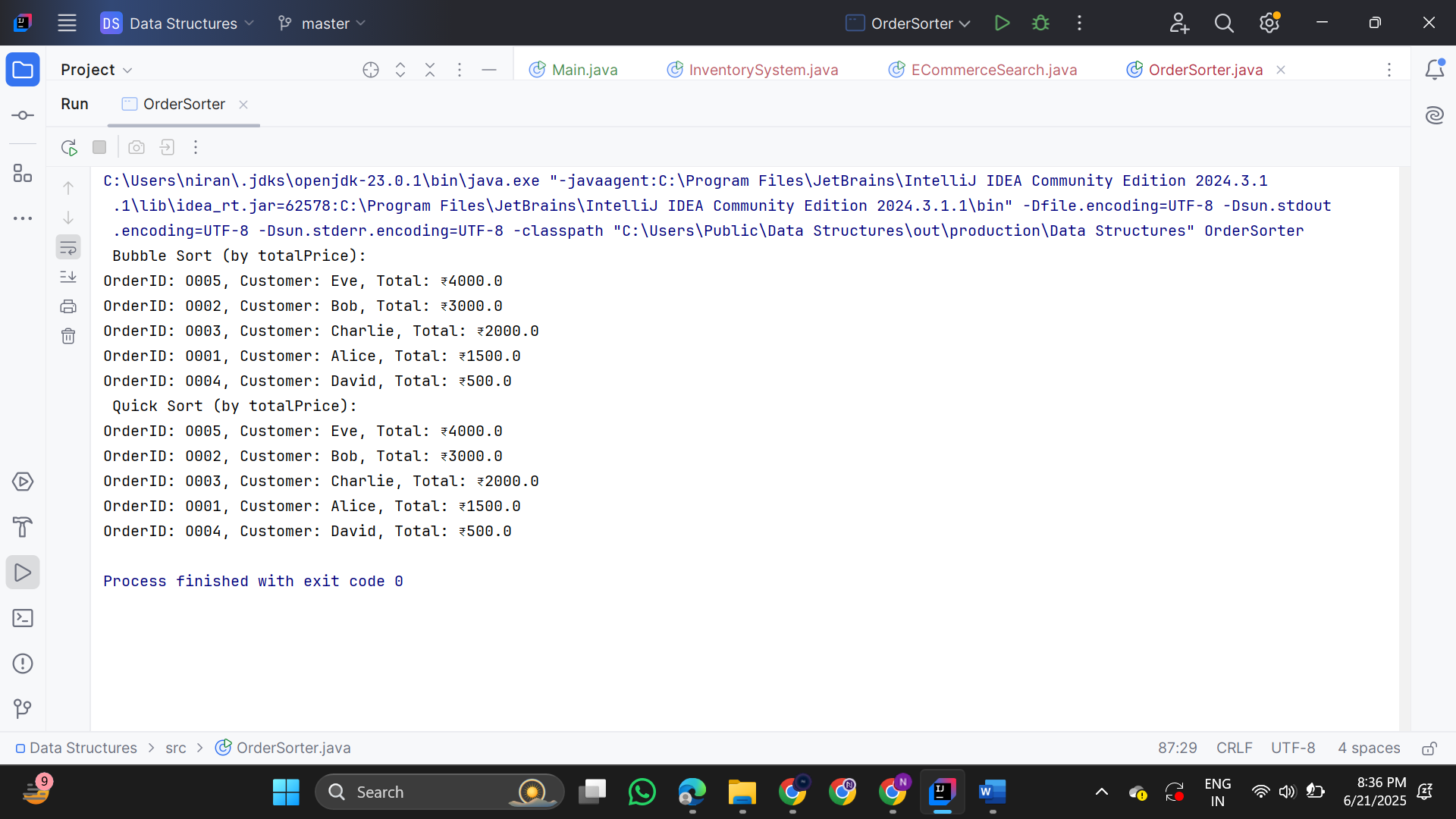
System.out.println(" Quick Sort (by totalPrice): ");

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

printOrders(orders);

}

}



**4.Employee Management System**

**Employee.java**  
public class Employee {

String employeeId;

String name;

String position;

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 toString() {

return "ID: " + employeeId + ", Name: " + name + ", Position: " + position + ", Salary: ₹" + salary;

}

}  
  
**EmployeeManagementSystem.java**import java.util.Scanner;

public class EmployeeManagementSystem {

static final int MAX = 100; // Max employees

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

static int count = 0;

public static void addEmployee(Employee emp) {

if (count < MAX) {

employees[count++] = emp;

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

} else {

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

}

}

public static Employee searchEmployee(String id) {

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

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

return employees[i];

}

}

return null;

}

public static void displayAll() {

if (count == 0) {

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

return;

}

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

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

}

}

public static void deleteEmployee(String id) {

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

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

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

employees[j] = employees[j + 1]; // shift left

}

employees[--count] = null; // remove last duplicate

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

return;

}

}

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

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

while (true) {

System.out.println("\n1.Add 2.Search 3.Display All 4.Delete 5.Exit");

int choice = sc.nextInt();

sc.nextLine(); // Consume newline

switch (choice) {

case 1:

System.out.print("ID: ");

String id = sc.nextLine();

System.out.print("Name: ");

String name = sc.nextLine();

System.out.print("Position: ");

String pos = sc.nextLine();

System.out.print("Salary: ");

double sal = sc.nextDouble();

addEmployee(new Employee(id, name, pos, sal));

break;

case 2:

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

String searchId = sc.nextLine();

Employee emp = searchEmployee(searchId);

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

break;

case 3:

displayAll();

break;

case 4:

System.out.print("Enter ID to delete: ");

String delId = sc.nextLine();

deleteEmployee(delId);

break;

case 5:

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

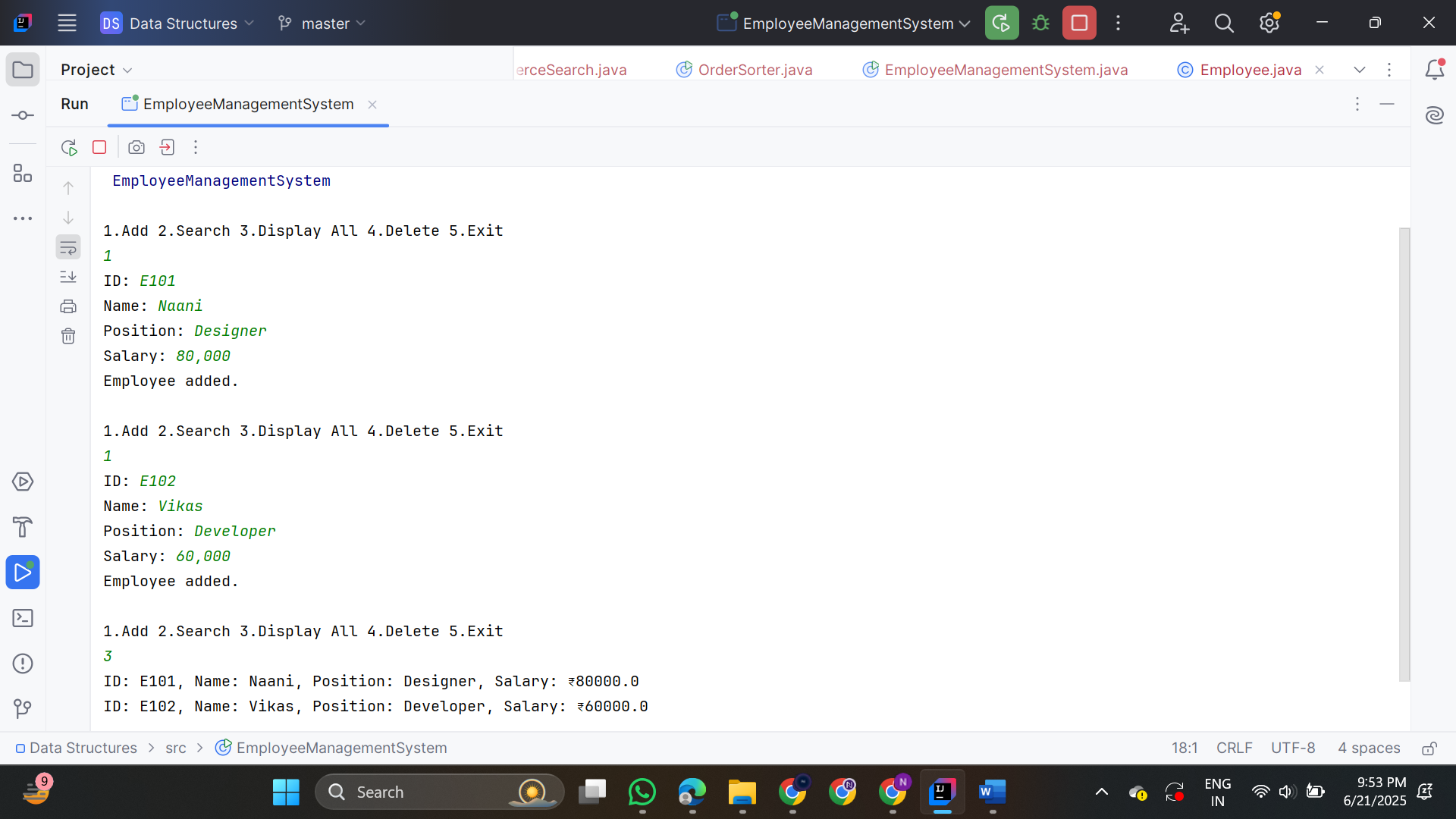
return;

}

}

}

}



**5. Task Management System**

**Task.java**  
class Task {

String taskId;

String taskName;

String status;

Task next;

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

this.taskId = taskId;

this.taskName = taskName;

this.status = status;

this.next = null;

}

public String toString() {

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

}

} **TaskManager.java**

public class TaskManager {

static Task head = null;

public static void addTask(Task newTask) {

if (head == null) {

head = newTask;

} else {

Task current = head;

while (current.next != null) {

current = current.next;

}

current.next = newTask;

}

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

}

public static Task searchTask(String taskId) {

Task current = head;

while (current != null) {

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

return current;

}

current = current.next;

}

return null;

}

public static void displayTasks() {

if (head == null) {

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

return;

}

Task current = head;

while (current != null) {

System.out.println(current);

current = current.next;

}

}

public static void deleteTask(String taskId) {

if (head == null) {

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

return;

}

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

head = head.next;

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

return;

}

Task prev = head;

Task current = head.next;

while (current != null) {

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

prev.next = current.next;

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

return;

}

prev = current;

current = current.next;

}

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

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

while (true) {

System.out.println(“ 1.Add Task 2.Search Task 3.Display All 4.Delete Task 5.Exit");

int choice = sc.nextInt();

sc.nextLine();

switch (choice) {

case 1:

System.out.print("Task ID: ");

String id = sc.nextLine();

System.out.print("Task Name: ");

String name = sc.nextLine();

System.out.print("Status: ");

String status = sc.nextLine();

addTask(new Task(id, name, status));

break;

case 2:

System.out.print("Enter Task ID to search: ");

String searchId = sc.nextLine();

Task found = searchTask(searchId);

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

break;

case 3:

displayTasks();

break;

case 4:

System.out.print("Enter Task ID to delete: ");

String delId = sc.nextLine();

deleteTask(delId);

break;

case 5:

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

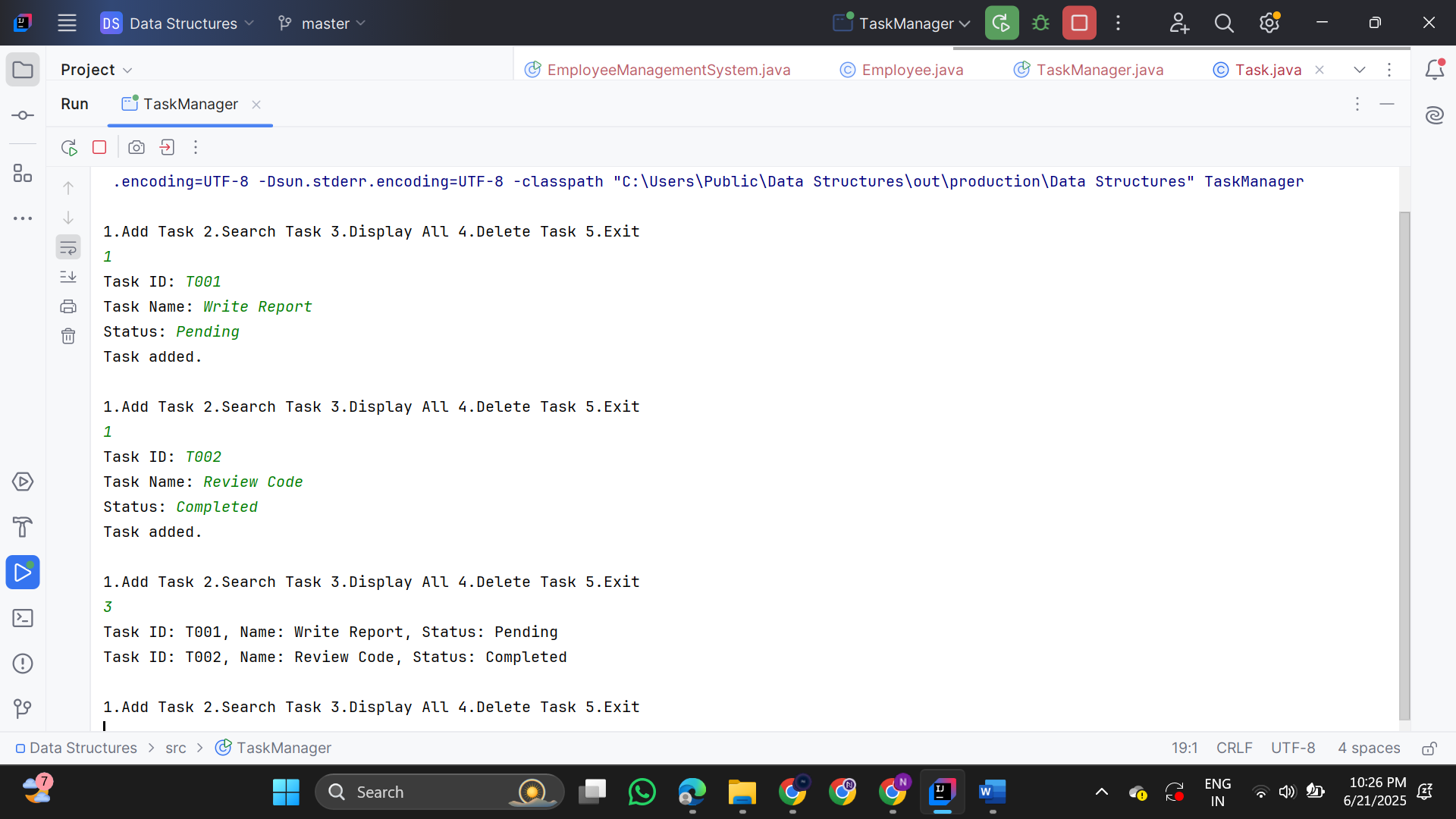
return;

}

}

}

}



**6.Library Management System**

class Book {

String bookId, title, author;

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

this.bookId = bookId;

this.title = title;

this.author = author;

}

public String toString() {

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

}

}

public class LibrarySearch {

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

for (Book b : books) {

if (b.title.equalsIgnoreCase(targetTitle)) {

return b;

}

}

return null;

}

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

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

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

while (left <= right) {

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

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

if (cmp == 0)

return books[mid];

else if (cmp < 0)

left = mid + 1;

else

right = mid - 1;

}

return null;

}

public static void main(String[] args) {

Book[] books = {

new Book("B101", "Java Basics", "Nisha Sharma"),

new Book("B102", "Data Structures", "Ravi Kumar"),

new Book("B103", "Operating Systems", "Anita Desai"),

new Book("B104", "Algorithms", "Ashok Mehta"),

new Book("B105", "Networking", "Ramesh Iyer")

};

Scanner sc = new Scanner(System.in);

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

String input = sc.nextLine();

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

Book result1 = linearSearch(books, input);

System.out.println(result1 != null ? result1 : "Book not found.");

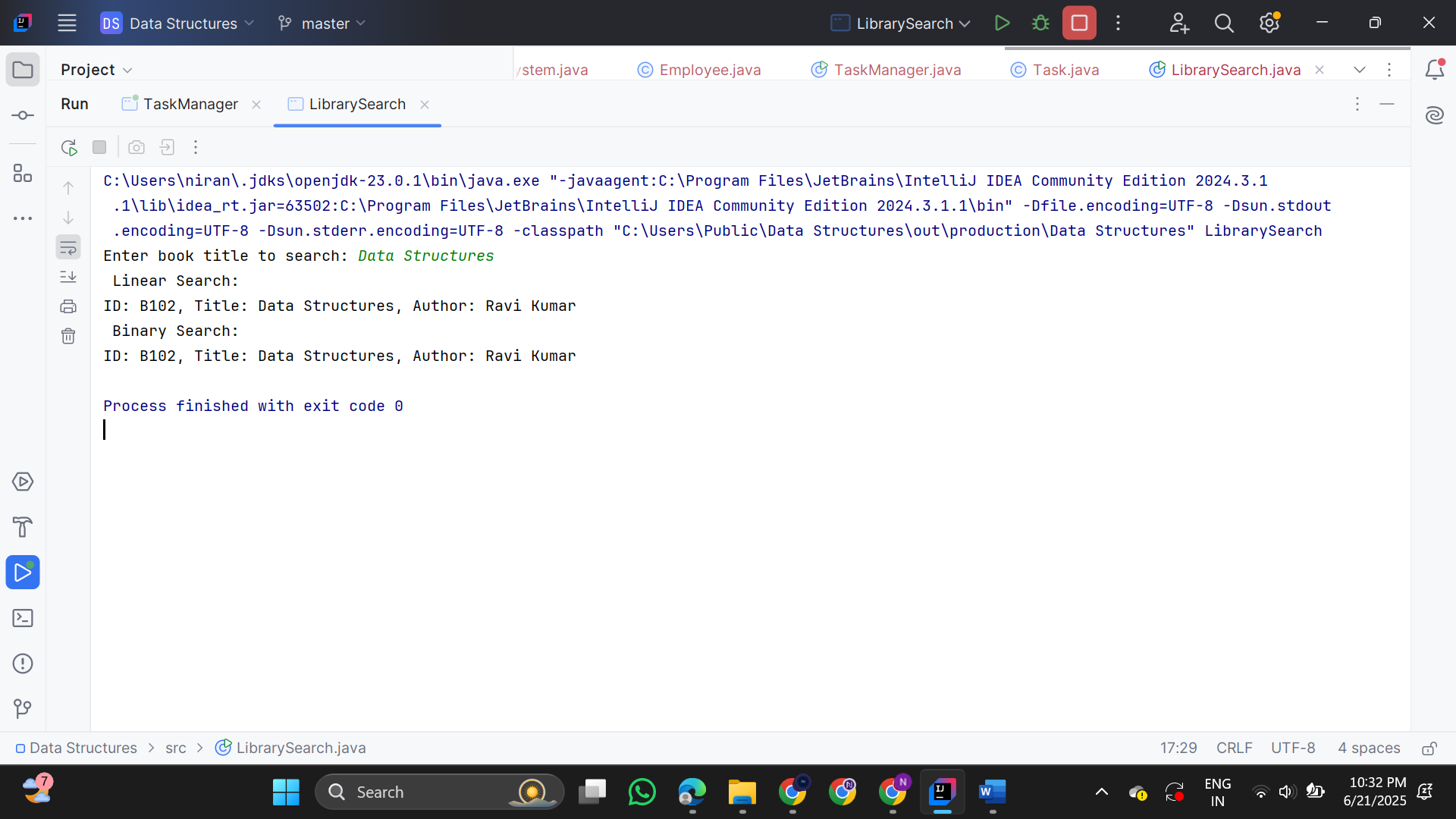
System.out.println(" Binary Search:");

Book result2 = binarySearch(books, input);

System.out.println(result2 != null ? result2 : "Book not found.");

}

}



**7. FinancialForecast**

public class FinancialForecast {

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

if (years == 0) {

return initialValue; }

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

}

public static void main(String[] args) {

double initialValue = 1000.0;

double growthRate = 0.10;

int years = 5;

double futureValue = forecast(initialValue, growthRate, years);

System.out.printf("Future Value after %d years: ₹%.2f", years, futureValue); }}

