COGNIZANT SUPERSET ID : 6424916

**ALGORITHMS\_ DATA STRUCTURES.**

**Exercise 1: Inventory Management System**

**Inventorymanager.java**

**Code:**

package Inventory;

import java.util.HashMap;

public class InventoryManager {

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

    public void addProduct(Product product) {

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

    }

    public void updateProduct(String 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(String productId) {

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

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

        }

    }

    public void displayInventory() {

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

            System.out.println(product);

        }

    }

}

**Main.java:**

**Code:**

package Inventory;

public class Main {

    public static void main(String[] args) {

        InventoryManager manager = new InventoryManager();

        manager.addProduct(new Product("P001", "Laptop", 10, 55000.00));

        manager.addProduct(new Product("P002", "Mouse", 150, 350.00));

        manager.displayInventory();

        manager.updateProduct("P001", "Gaming Laptop", 8, 60000.00);

        manager.deleteProduct("P002");

        manager.displayInventory();

    }

}

**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

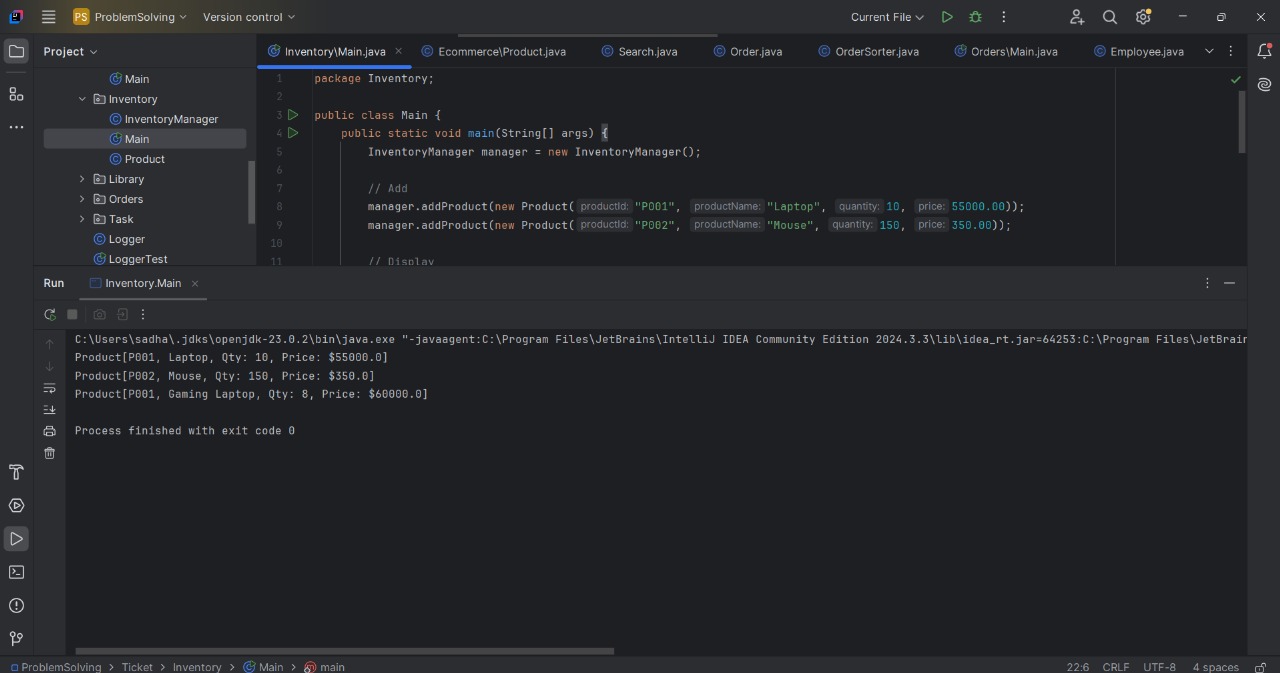
    public String toString() {

        return "Product[" + productId + ", " + productName + ", Qty: " + quantity + ", Price: $" + price + "]";

    }

}

**OUTPUT:**

****

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

**Main.java:**

package Ecommerce;

public class Main {

    public static void main(String[] args) {

        Product[] products = {

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

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

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

                new Product("P004", "Shirt", "Clothing"),

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

        };

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

        Product foundLinear = Search.linearSearch(products, "Mouse");

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

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

        Search.sortProductsByName(products);

        Product foundBinary = Search.binarySearch(products, "Mouse");

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

    }

}

**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[" + productId + ", " + productName + ", " + category + "]";

    }

}

**Search.java:**

package Ecommerce;

import java.util.Arrays;

import java.util.Comparator;

public class Search {

    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;

        int right = products.length - 1;

        while (left <= right) {

            int mid = (left + right) / 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;

    }

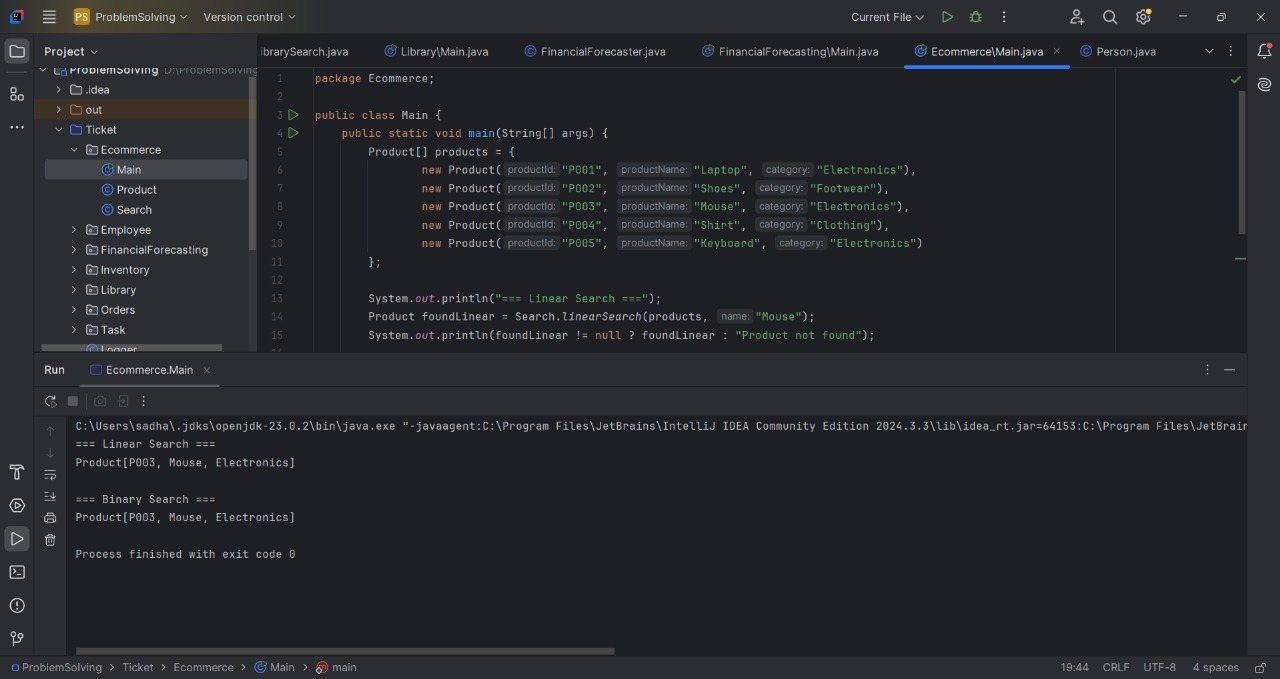
    public static void sortProductsByName(Product[] products) {

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

    }

}

**Output:**

****

**Exercise 3: Sorting Customer Orders**

**Main.c**

package Orders;

public class Main {

    public static void main(String[] args) {

        Order[] orders = {

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

                new Order("O002", "Bob", 950.0),

                new Order("O003", "Charlie", 2050.5),

                new Order("O004", "David", 1200.0)

        };

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

        for (Order order : orders) System.out.println(order);

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

        OrderSorter.bubbleSort(orders);

        for (Order order : orders) System.out.println(order);

        orders = new Order[]{

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

                new Order("O002", "Bob", 950.0),

                new Order("O003", "Charlie", 2050.5),

                new Order("O004", "David", 1200.0)

        };

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

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

        for (Order order : orders) System.out.println(order);

    }

}

**Order.java:**

package Orders;

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 double getTotalPrice() { return totalPrice; }

    @Override

    public String toString() {

        return "Order[" + orderId + ", " + customerName + ", ₹" + totalPrice + "]";

    }

}

**Ordersorted.java:**

package Orders;

public class OrderSorter {

    public static void bubbleSort(Order[] orders) {

        int n = orders.length;

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

            boolean swapped = false;

            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;

                    swapped = true;

                }

            }

            if (!swapped) break;

        }

    }

    public static 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 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++;

                Order temp = orders[i];

                orders[i] = orders[j];

                orders[j] = temp;

            }

        }

        Order temp = orders[i + 1];

        orders[i + 1] = orders[high];

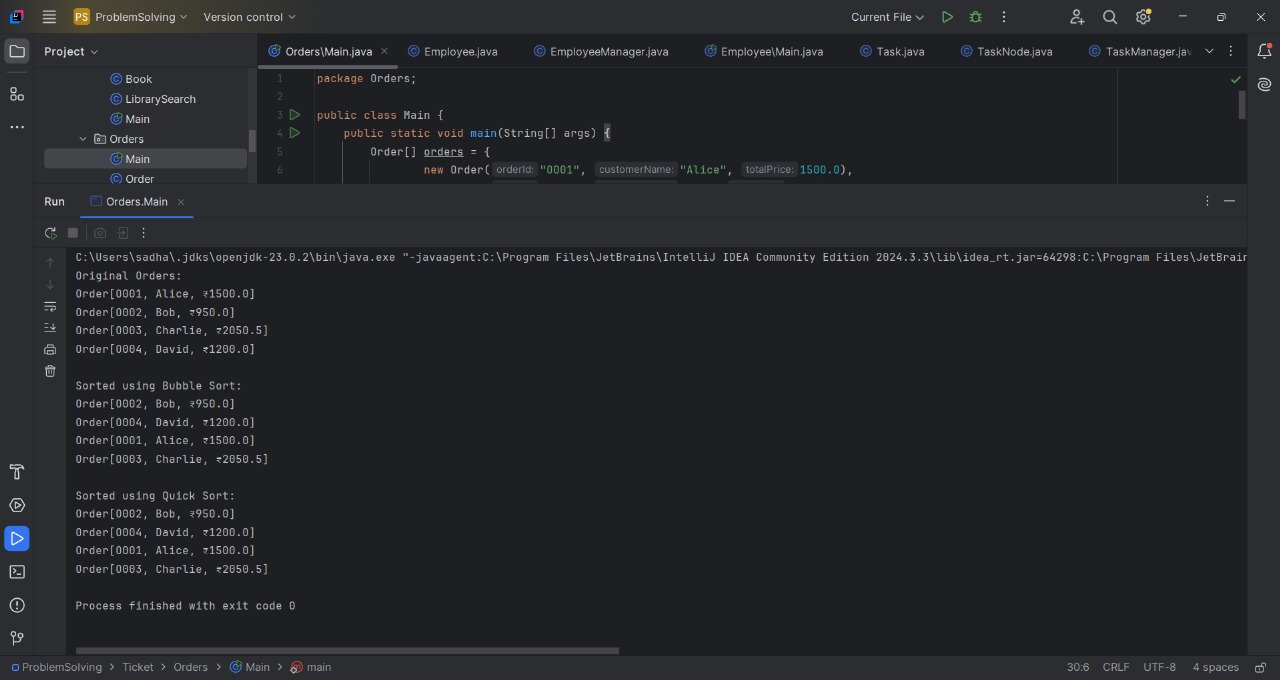
        orders[high] = temp;

        return i + 1;

    }

}

**Output:**

****

**Exercise 4: Employee Management System**

**Main.java**

package Employee;

public class Main {

    public static void main(String[] args) {

        EmployeeManager manager = new EmployeeManager(5);

        manager.addEmployee(new Employee("E001", "Alice", "Manager", 75000));

        manager.addEmployee(new Employee("E002", "Bob", "Engineer", 50000));

        manager.addEmployee(new Employee("E003", "Charlie", "HR", 40000));

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

        manager.displayEmployees();

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

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

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

        System.out.println("\nDelete E002:");

        manager.deleteEmployee("E002");

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

        manager.displayEmployees();

    }

}

**Employee.java**

package Employee;

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[" + employeeId + ", " + name + ", " + position + ", ₹" + salary + "]";

    }

}

**Employeemanager.java:**

package Employee;

public class EmployeeManager {

    private Employee[] employees;

    private int size;

    public EmployeeManager(int capacity) {

        employees = new Employee[capacity];

        size = 0;

    }

    public void addEmployee(Employee emp) {

        if (size < employees.length) {

            employees[size++] = emp;

        } else {

            System.out.println("Cannot add more employees. Array is full.");

        }

    }

    public Employee searchEmployee(String empId) {

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

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

                return employees[i];

            }

        }

        return null;

    }

    public void displayEmployees() {

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

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

        }

    }

    public void deleteEmployee(String empId) {

        int index = -1;

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

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

                index = i;

                break;

            }

        }

        if (index != -1) {

            // Shift left

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

                employees[i] = employees[i + 1];

            }

            employees[--size] = null;

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

        } else {

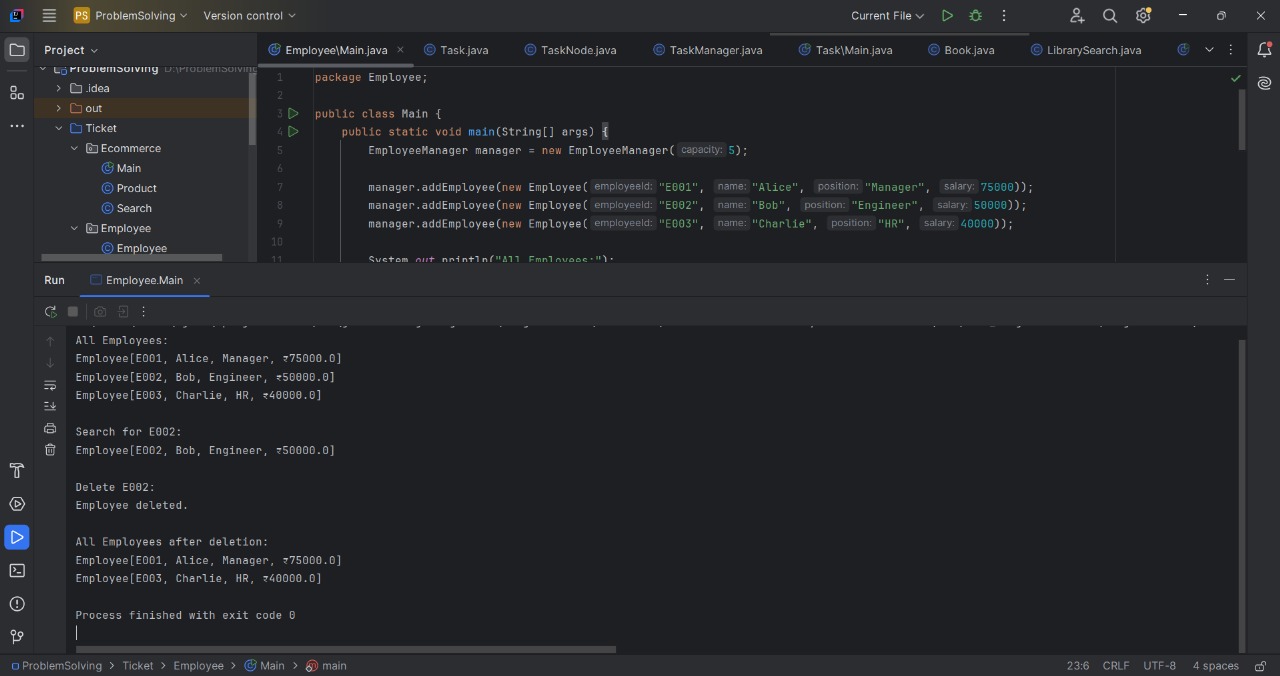
            System.out.println("Employee not found.");

        }

    }

}

**OUTPUT:**



**Exercise 5: Task Management System**

**Main.java:**

package Task;

public class Main {

    public static void main(String[] args) {

        TaskManager manager = new TaskManager();

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

        manager.addTask(new Task("T002", "Implement Backend", "In Progress"));

        manager.addTask(new Task("T003", "Write Tests", "Pending"));

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

        manager.displayTasks();

        System.out.println("\nSearch T002:");

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

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

        System.out.println("\nDeleting T002:");

        manager.deleteTask("T002");

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

        manager.displayTasks();

    }

}

**Task.java:**

package Task;

public class Task {

    private String taskId;

    private String taskName;

    private String status;

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

        this.taskId = taskId;

        this.taskName = taskName;

        this.status = status;

    }

    public String getTaskId() { return taskId; }

    public String getTaskName() { return taskName; }

    public String getStatus() { return status; }

    public void setStatus(String status) { this.status = status; }

    @Override

    public String toString() {

        return "Task[" + taskId + ", " + taskName + ", " + status + "]";

    }

}

**Taskmanager.java:**

package Task;

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;

        }

    }

    public Task searchTask(String taskId) {

        TaskNode current = head;

        while (current != null) {

            if (current.task.getTaskId().equals(taskId)) {

                return current.task;

            }

            current = current.next;

        }

        return null;

    }

    public void displayTasks() {

        TaskNode current = head;

        while (current != null) {

            System.out.println(current.task);

            current = current.next;

        }

    }

    public void deleteTask(String taskId) {

        if (head == null) return;

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

            head = head.next;

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

            return;

        }

        TaskNode current = head;

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

            current = current.next;

        }

        if (current.next != null) {

            current.next = current.next.next;

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

        } else {

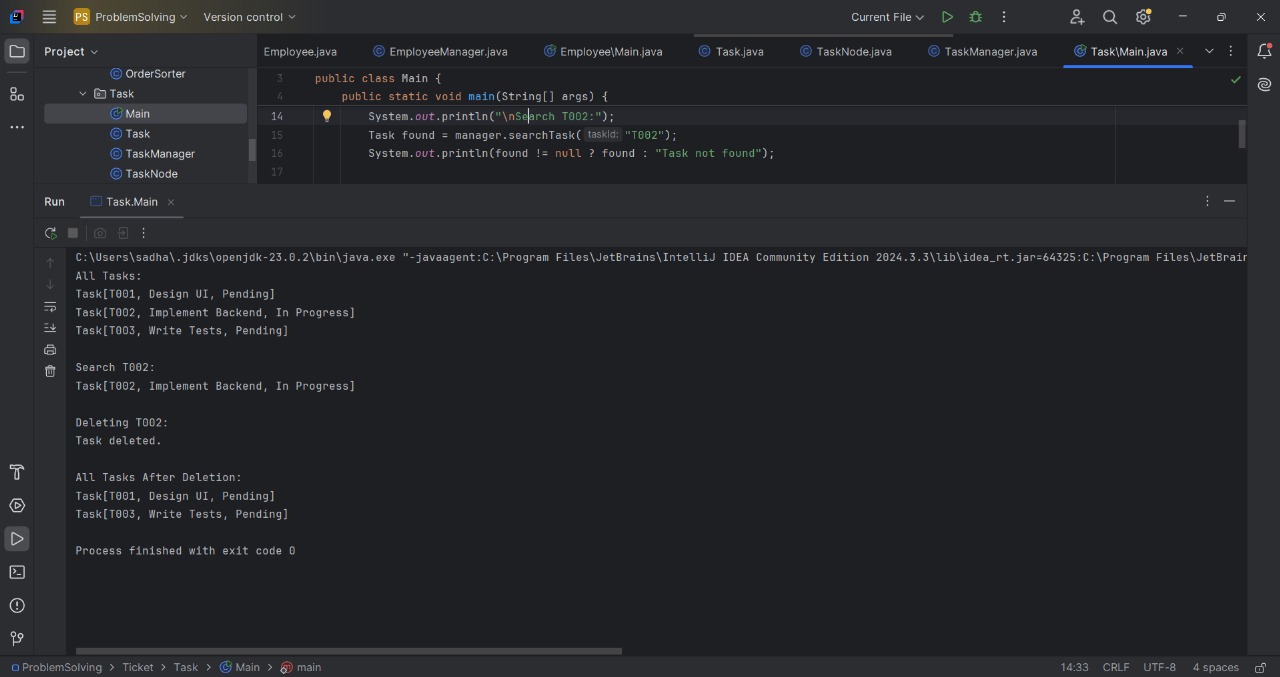
            System.out.println("Task not found.");

        }

    }

}

**OUTPUT:**



**Exercise 6: Library Management System**

**Main.java**

package Library;

public class Main {

    public static void main(String[] args) {

        Book[] books = {

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

                new Book("B002", "Clean Code", "Robert C. Martin"),

                new Book("B003", "Atomic Habits", "James Clear"),

                new Book("B004", "1984", "George Orwell")

        };

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

        Book foundLinear = LibrarySearch.linearSearch(books, "Atomic Habits");

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

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

        LibrarySearch.sortByTitle(books); // Must sort first

        Book foundBinary = LibrarySearch.binarySearch(books, "Atomic Habits");

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

    }

}

**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 getTitle() { return title; }

    public String getAuthor() { return author; }

    @Override

    public String toString() {

        return "Book[" + bookId + ", " + title + ", " + author + "]";

    }

}

**Library.java:**

package Library;

import java.util.Arrays;

import java.util.Comparator;

public class LibrarySearch {

    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;

    }

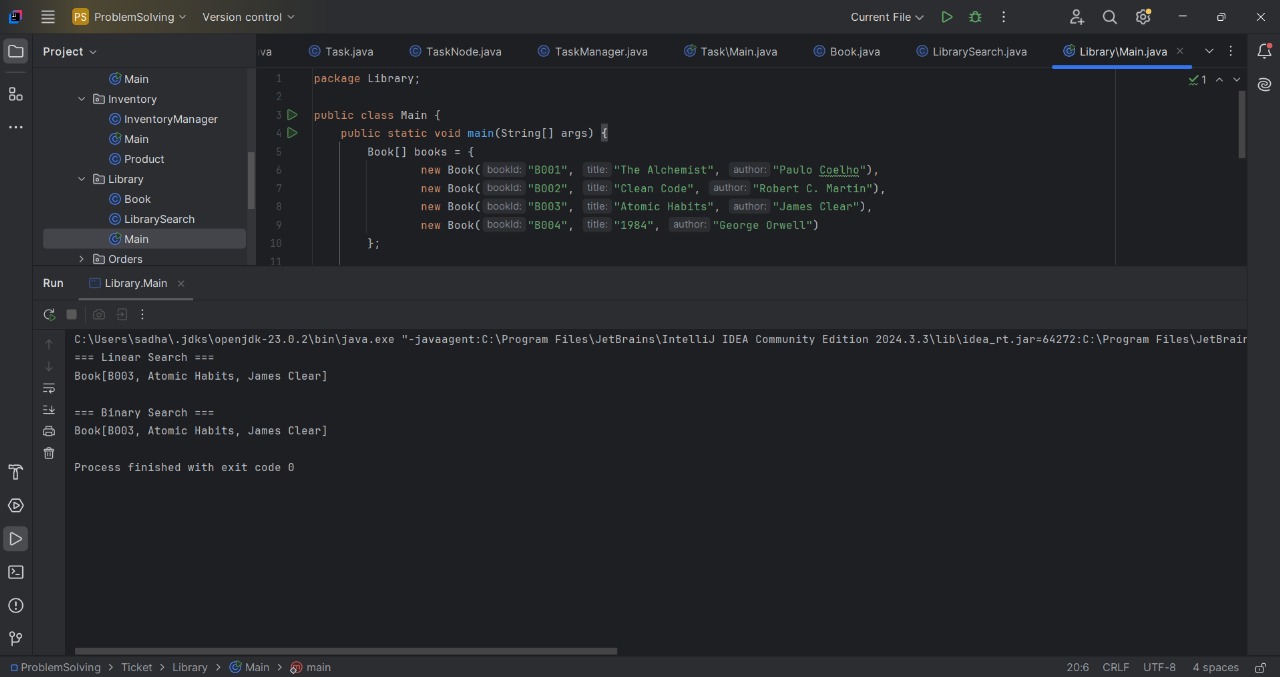
    public static void sortByTitle(Book[] books) {

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

    }

}

**OUTPUT:**

****

**Exercise 7: Financial Forecasting**

**Main.java:**

package FinancialForecasting;

public class Main {

    public static void main(String[] args) {

        double initialValue = 10000;

        double growthRate = 0.08; // 8%

        int years = 5;

        double result1 = FinancialForecaster.futureValueRecursive(initialValue, growthRate, years);

        System.out.printf("Future Value (Recursive): ₹%.2f%n", result1);

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

        double result2 = FinancialForecaster.futureValueMemo(initialValue, growthRate, years, memo);

        System.out.printf("Future Value (Memoized): ₹%.2f%n", result2);

    }

}

**Financial forecaster.java:**

package FinancialForecasting;

public class FinancialForecaster {

    public static double futureValueRecursive(double currentValue, double growthRate, int years) {

        if (years == 0) return currentValue;

        return futureValueRecursive(currentValue, growthRate, years - 1) \* (1 + growthRate);

    }

    public static double futureValueMemo(double currentValue, double growthRate, int years, double[] memo) {

        if (years == 0) return currentValue;

        if (memo[years] != 0.0) return memo[years];

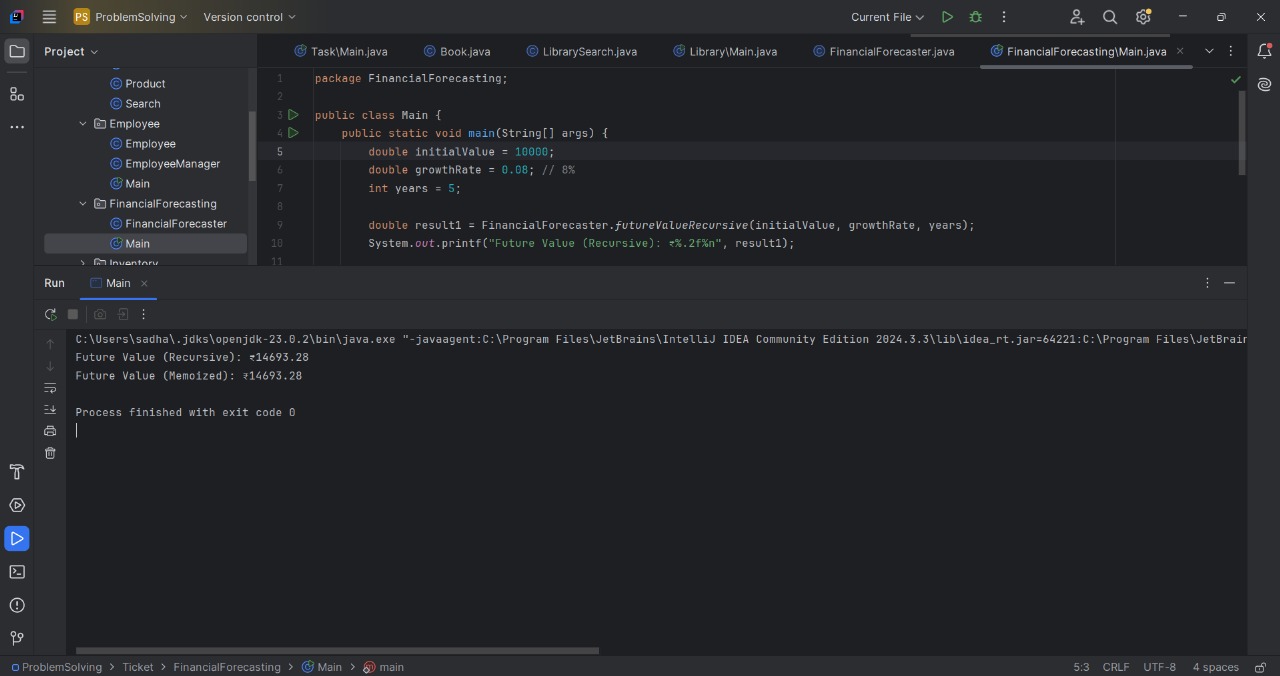
        memo[years] = futureValueMemo(currentValue, growthRate, years - 1, memo) \* (1 + growthRate);

        return memo[years];

    }

}

**OUTPUT:**

****