**DATA STRUCTURES WEEK 1`**

**Exercise 1: Inventory Management System**

* **CODE FOR Product.java**

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 "Product ID: " + productId +

            ", Name: " + productName +

            ", Quantity: " + quantity +

            ", Price: Rs." + price;

    }

}

* **CODE FOR Inventory.java**

import java.util.HashMap;

public class Inventory {

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

    public void addProduct(Product product) {

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

        System.out.println("Added: " + product.getProductName());

    }

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

        if (inventory.containsKey(productId)) {

            Product p = inventory.get(productId);

            p.setQuantity(newQty);

            p.setPrice(newPrice);

            System.out.println("Updated: " + p);

        } else {

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

        }

    }

    public void deleteProduct(int productId) {

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

            System.out.println("Deleted product ID: " + productId);

        } else {

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

        }

    }

    public void displayInventory() {

        if (inventory.isEmpty()) {

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

        } else {

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

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

                System.out.println(p);

            }

        }

    }

}

* **CODE FOR Main.java**

public class Main {

    public static void main(String[] args) {

        Inventory manager = new Inventory();

        Product p1 = new Product(01, "Book", 10, 500);

        Product p2 = new Product(02, "Pen", 50, 10);

        Product p3 = new Product(03, "Laptop", 30, 72000);

        manager.addProduct(p1);

        manager.addProduct(p2);

        manager.addProduct(p3);

        manager.displayInventory();

        manager.updateProduct(02, 60, 20);

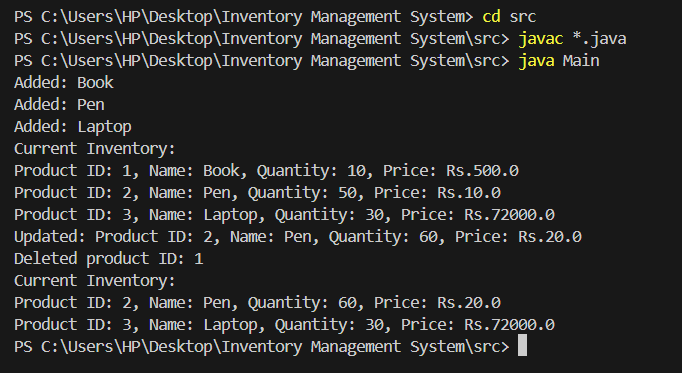
        manager.deleteProduct(01);

        manager.displayInventory();

    }

}

* **OUTPUT**



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

* **CODE FOR Product.java**

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;

    }

    @Override

    public String toString() {

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

    }

}

* **CODE FOR ProductSearch.java**

import java.util.Arrays;

import java.util.Comparator;

public class ProductSearch {

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

        for (Product p : products) {

            if (p.productName.equalsIgnoreCase(targetName)) {

                return p;

            }

        }

        return null;

    }

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

        int left = 0;

        int right = products.length - 1;

        while (left <= right) {

            int mid = (left + right) / 2;

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

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

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

            else right = mid - 1;

        }

        return null;

    }

    public static void main(String[] args) {

        Product[] products = {

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

            new Product(102, "Shampoo", "Personal Care"),

            new Product(103, "Notebook", "Stationery"),

            new Product(104, "Smartphone", "Electronics"),

            new Product(105, "Tablet", "Electronics")

        };

        // Sort for binary search

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

        // Perform Linear Search

        Product result1 = linearSearch(products, "Notebook");

        System.out.println("Linear Search Result: " + (result1 != null ? result1 : "Not Found"));

        // Perform Binary Search

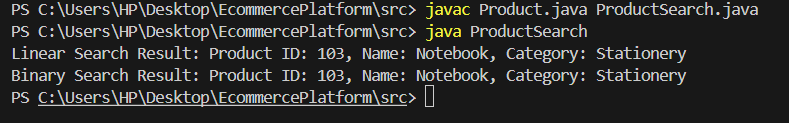
        Product result2 = binarySearch(products, "Notebook");

        System.out.println("Binary Search Result: " + (result2 != null ? result2 : "Not Found"));

    }

}

* **OUTPUT**

****

**Exercise 7: Financial Forecasting**

* CODE FOR Financial Forecasting

public class FinancialForecast {

    public static double futureValue(double presentValue, double growthRate, int years) {

        if (years == 0) {

            return presentValue;

        } else {

            return futureValue(presentValue \* (1 + growthRate), growthRate, years - 1);

        }

    }

    public static void main(String[] args) {

        double presentValue = 10000;

        double growthRate = 0.07;

        int years = 5;

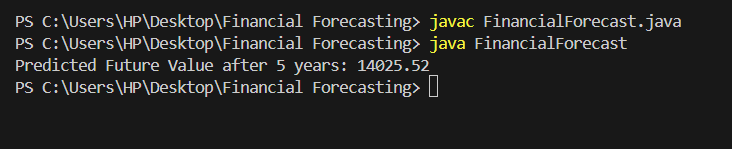
        double predictedValue = futureValue(presentValue, growthRate, years);

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

    }

}

**OUTPUT**

****

**Exercise 3: Sorting Customer Orders**

* CODE FOR 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;

    }

    @Override

    public String toString() {

        return "Order ID: " + orderId + ", Customer: " + customerName + ", Total: Rs." + totalPrice;

    }

}

* CODE FOR OrderSort.java

public class OrderSort {

    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 - 1 - i; j++) {

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

                    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 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].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 displayOrders(Order[] orders, String title) {

        System.out.println("=== " + title + " ===");

        for (Order o : orders) {

            System.out.println(o);

        }

        System.out.println();

    }

    public static void main(String[] args) {

        Order[] orders = {

            new Order(101, "Rahul", 500.0),

            new Order(102, "Rajesh", 400.0),

            new Order(103, "Rohan", 700.0),

            new Order(104, "Raj", 600.0),

        };

        Order[] bubbleSorted = orders.clone();

        bubbleSort(bubbleSorted);

        displayOrders(bubbleSorted, "Sorted by Bubble Sort");

        Order[] quickSorted = orders.clone();

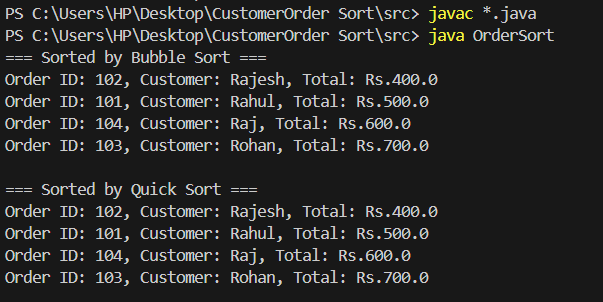
        quickSort(quickSorted, 0, quickSorted.length - 1);

        displayOrders(quickSorted, "Sorted by Quick Sort");

    }

}

* OUTPUT



**Exercise 4: Employee Management System**

* CODE FOR Employees.java

public class Employees {

    int Emp\_Id;

    String Name;

    String Position;

    double Salary;

    public Employees(int Emp\_Id, String Name, String Position, double Salary) {

        this.Emp\_Id = Emp\_Id;

        this.Name = Name;

        this.Position = Position;

        this.Salary = Salary;

    }

    @Override

    public String toString() {

        return "Employee ID: " + Emp\_Id + ", Name: " + Name +

            ", Position: " + Position + ", Salary: Rs." + Salary;

    }

}

* CODE FOR EmployeeManager.java

public class EmployeeManager {

    private Employees[] employees;

    private int size;

    public EmployeeManager(int capacity) {

        employees = new Employees[capacity];

        size = 0;

    }

    public void addEmployee(Employees emp) {

        if (size < employees.length) {

            employees[size++] = emp;

            System.out.println("Added: " + emp.Name);

        } else {

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

        }

    }

    public Employees searchEmployee(int id) {

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

            if (employees[i].Emp\_Id == id) {

                return employees[i];

            }

        }

        return null;

    }

    public void listEmployees() {

        if (size == 0) {

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

            return;

        }

        System.out.println("Employee List:");

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

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

        }

    }

    public void deleteEmployee(int id) {

        int index = -1;

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

            if (employees[i].Emp\_Id == id) {

                index = i;

                break;

            }

        }

        if (index == -1) {

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

        } else {

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

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

            }

            employees[--size] = null;

            System.out.println("Deleted employee with ID: " + id);

        }

    }

}

* CODE FOR Main.java

public class Main {

    public static void main(String[] args) {

        EmployeeManager manager = new EmployeeManager(5);

        Employees e1 = new Employees(10, "Rahul", "Manager", 70000);

        Employees e2 = new Employees(11, "Rajesh", "Software Engineer", 60000);

        Employees e3 = new Employees(12, "Rohan", "HR", 65000);

        manager.addEmployee(e1);

        manager.addEmployee(e2);

        manager.addEmployee(e3);

        manager.listEmployees();

        System.out.println("\nSearching for employee ID 11:");

        Employees found = manager.searchEmployee(11);

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

        System.out.println("\nDeleting employee ID 12:");

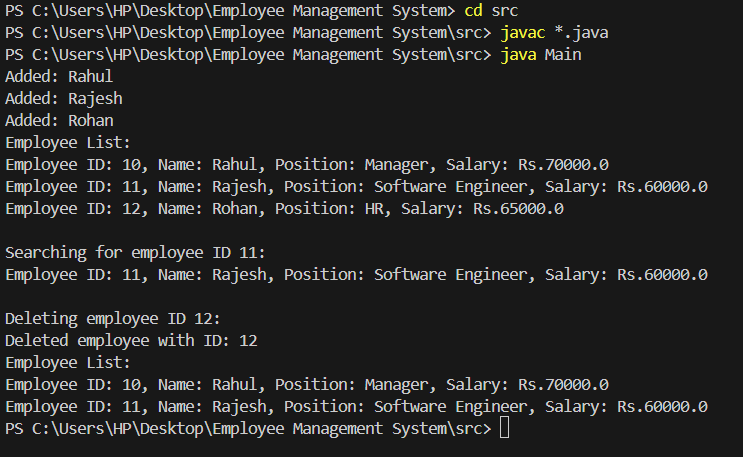
        manager.deleteEmployee(12);

        manager.listEmployees();

    }

}

* OUTPUT



**Exercise 5: Task Management System**

* CODE FOR Task.java

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;

    }

}

* CODE FOR Task\_node.java

public class Task\_node {

    Task task;

    Task\_node next;

    public Task\_node(Task task) {

        this.task = task;

        this.next = null;

    }

}

* CODE FOR Task\_manager.java

public class Task\_manager {

    private Task\_node head;

    public void addTask(Task task) {

        Task\_node newNode = new Task\_node(task);

        if (head == null) {

            head = newNode;

        } else {

            Task\_node current = head;

            while (current.next != null) {

                current = current.next;

            }

            current.next = newNode;

        }

        System.out.println("Added: " + task.taskName);

    }

    public Task searchTask(int taskId) {

        Task\_node current = head;

        while (current != null) {

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

                return current.task;

            }

            current = current.next;

        }

        return null;

    }

* CODE FOR Main.java

public class Main {

    public static void main(String[] args) {

        Task\_manager manager = new Task\_manager();

        Task t1 = new Task(01, "Build Backend", "Completed");

        Task t2 = new Task(02, "Develop Frontend", "In Progress");

        Task t3 = new Task(03, "Testing", "Pending");

        manager.addTask(t1);

        manager.addTask(t2);

        manager.addTask(t3);

        manager.listTasks();

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

        Task found = manager.searchTask(2);

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

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

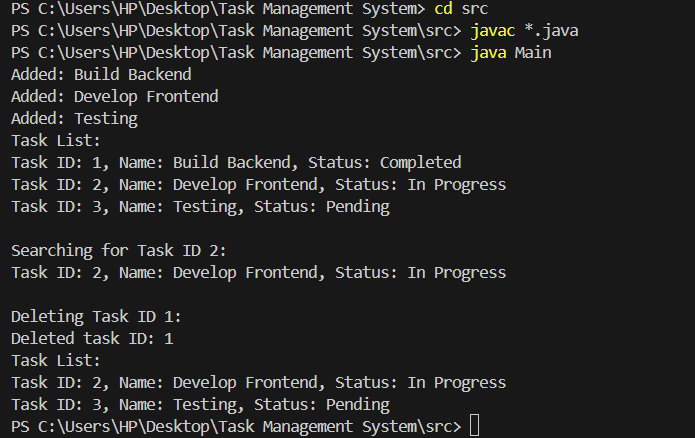
        manager.deleteTask(1);

        manager.listTasks();

    }

}

* OUTPUT



**Exercise 6: Library Management System**

* CODE FOR Book.java

public class Book {

    int bookId;

    String title;

    String author;

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

        this.bookId = bookId;

        this.title = title;

        this.author = author;

    }

    @Override

    public String toString() {

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

    }

}

* CODE FOR Library\_management.java

import java.util.Arrays;

import java.util.Comparator;

public class Library\_management {

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

        for (Book book : books) {

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

                return book;

            }

        }

        return null;

    }

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

        int low = 0, high = books.length - 1;

        while (low <= high) {

            int mid = (low + high) / 2;

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

            if (comparison == 0) {

                return books[mid];

            } else if (comparison < 0) {

                low = mid + 1;

            } else {

                high = mid - 1;

            }

        }

        return null;

    }

    public static void displayBooks(Book[] books) {

        System.out.println("Books in Library:");

        for (Book book : books) {

            System.out.println(book);

        }

    }

    public static void main(String[] args) {

        Book[] books = {

            new Book(1, "ABC", "Alice"),

            new Book(2, "DEF", "Bob"),

            new Book(3, "HIJ", "Charlie"),

            new Book(4, "PQR", "David"),

        };

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

        displayBooks(books);

        String searchTitle = "ABC";

        System.out.println("\nSearching using Linear Search:");

        Book result1 = linearSearch(books, searchTitle);

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

        System.out.println("\nSearching using Binary Search:");

        Book result2 = binarySearch(books, searchTitle);

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

    }

}

* OUTPUT

