Week 1 task : Algorithm_Data_Structures

Exercise 1: Inventory Management System

Code :

InventorySystem.java > ...

```java
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;
    }

    @Override
    public String toString() {
        return "Product ID: " + productId +
               ", Name: " + productName +
               ", Quantity: " + quantity +
               ", Price: ₹" + price;
    }
}

class InventoryManager {
    private HashMap<String, Product> inventory;

    public InventoryManager() {
        inventory = new HashMap<>();
    }

    public void addProduct(Product product) {
        if (inventory.containsKey(product.productId)) {
            System.out.println(x:"Product ID already exists.");
        } else {
```

```
36          } else {
37              inventory.put(product.productId, product);
38              System.out.println(x:"Product added successfully.");
39          }
40      }
41
42      public void updateProduct(String productId, int newQuantity, double newPrice) {
43          if (inventory.containsKey(productId)) {
44              Product product = inventory.get(productId);
45              product.quantity = newQuantity;
46              product.price = newPrice;
47              System.out.println(x:"Product updated successfully.");
48          } else {
49              System.out.println(x:"Product ID not found.");
50          }
51      }
52
53      public void deleteProduct(String productId) {
54          if (inventory.remove(productId) != null) {
55              System.out.println(x:"Product deleted successfully.");
56          } else {
57              System.out.println(x:"Product ID not found.");
58          }
59      }
60
61      public void displayInventory() {
62          if (inventory.isEmpty()) {
63              System.out.println(x:"Inventory is empty.");
64          } else {
65              for (Product product : inventory.values()) {
66                  System.out.println(product);
67              }
68          }
69      }
```

```java
}

public class InventorySystem {
    public static void main(String[] args) {
        InventoryManager manager = new InventoryManager();
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println(x:"\nInventory Menu:");
            System.out.println(x:"1. Add Product");
            System.out.println(x:"2. Update Product");
            System.out.println(x:"3. Delete Product");
            System.out.println(x:"4. Display Inventory");
            System.out.println(x:"5. Exit");
            System.out.print(s:"Choose option: ");

            int choice = scanner.nextInt();
            scanner.nextLine();

            String productId, productName;
            int quantity;
            double price;

            switch (choice) {
                case 1:
                    System.out.print(s:"Enter Product ID: ");
                    productId = scanner.nextLine();
                    System.out.print(s:"Enter Product Name: ");
                    productName = scanner.nextLine();
                    System.out.print(s:"Enter Quantity: ");
                    quantity = scanner.nextInt();
                    System.out.print(s:"Enter Price: ");
                    price = scanner.nextDouble();
                    manager.addProduct(new Product(productId, productName, quantity, price));
                    break;
```

```
                case 2:
                    System.out.print(s:"Enter Product ID to Update: ");
                    productId = scanner.nextLine();
                    System.out.print(s:"Enter New Quantity: ");
                    quantity = scanner.nextInt();
                    System.out.print(s:"Enter New Price: ");
                    price = scanner.nextDouble();
                    manager.updateProduct(productId, quantity, price);
                    break;
                case 3:
                    System.out.print(s:"Enter Product ID to Delete: ");
                    productId = scanner.nextLine();
                    manager.deleteProduct(productId);
                    break;
                case 4:
                    manager.displayInventory();
                    break;
                case 5:
                    System.out.println(x:"Exiting... Bye!");
                    scanner.close();
                    return;
                default:
                    System.out.println(x:"Invalid choice. Try again.");
            }
        }
    }
}

```

Output :

```
PS C:\Users\Admin\Desktop\JAVA QS 2> javac InventorySystem.java
PS C:\Users\Admin\Desktop\JAVA QS 2> java InventorySystem

  Inventory Menu:
  1. Add Product
  2. Update Product
  3. Delete Product
  4. Display Inventory
  5. Exit
  Choose option: 1
  Enter Product ID: 10
  Enter Product Name: Dabur
  Enter Quantity: 20
  Enter Price: 50
  Product added successfully.

  Inventory Menu:
  1. Add Product
  2. Update Product
  3. Delete Product
  4. Display Inventory
  5. Exit
  Choose option: 5
  Exiting... Bye!
PS C:\Users\Admin\Desktop\JAVA QS 2>
```

Exercise 2: E-commerce Platform Search Function

Code :

```java
import java.util.Arrays;
import java.util.Comparator;

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;
    }
}

public class ECommerceSearch {

    public static Product linearSearch(Product[] products, String targetName) {
        for (Product product : products) {
            if (product.productName.equalsIgnoreCase(targetName)) {
                return product;
            }
        }
        return null;
    }

    public static Product binarySearch(Product[] products, String targetName) {
        int left = 0, right = products.length - 1;
        while (left <= right) {
            int mid = left + (right - left) / 2;
            int result = products[mid].productName.compareToIgnoreCase(targetName);
```

```
            int result = products[mid].productName.compareToIgnoreCase(targetName);
            if (result == 0) {
                return products[mid];
            } else if (result < 0) {
                left = mid + 1;
            } else {
                right = mid - 1;
            }
        }
        return null;
    }

    public static void main(String[] args) {
        Product[] productList = {
            new Product(productId:101, productName:"Laptop", category:"Electronics"),
            new Product(productId:102, productName:"Shoes", category:"Fashion"),
            new Product(productId:103, productName:"Book", category:"Education"),
            new Product(productId:104, productName:"Watch", category:"Accessories"),
            new Product(productId:105, productName:"Headphones", category:"Electronics")
        };

        Product linearResult = linearSearch(productList, targetName:"Book");
        System.out.println("Linear Search Result: " + (linearResult != null ? linearResult : "Product not found"));

        Arrays.sort(productList, Comparator.comparing(p -> p.productName.toLowerCase()));
        Product binaryResult = binarySearch(productList, targetName:"Book");
        System.out.println("Binary Search Result: " + (binaryResult != null ? binaryResult : "Product not found"));
    }
}
```

Output :

```
PS C:\Users\Admin\Desktop\JAVA QS 2> javac ECommerceSearch.java
PS C:\Users\Admin\Desktop\JAVA QS 2> java ECommerceSearch
Linear Search Result: Product ID: 103, Name: Book, Category: Education
Binary Search Result: Product ID: 103, Name: Book, Category: Education
PS C:\Users\Admin\Desktop\JAVA QS 2>
```

Exercise 3: Sorting Customer Orders

Code :

```java
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 String toString() {
        return "[" + orderId + ", " + customerName + ", Total: " + totalPrice + "]";
    }
}

public class OrderSorting {
    static void bubbleSort(Order[] orders) {
        for (int i = 0; i < orders.length - 1; i++) {
            for (int j = 0; j < orders.length - i - 1; j++) {
                if (orders[j].totalPrice > orders[j + 1].totalPrice) {
                    Order temp = orders[j];
                    orders[j] = orders[j + 1];
                    orders[j + 1] = temp;
                }
            }
        }
    }

    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);
        }
    }
```

```java
    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;
    }

    Run main | Debug main | Run | Debug
    public static void main(String[] args) {
        Order[] orders = {
            new Order(orderId:1, customerName:"Alice", totalPrice:2500.0),
            new Order(orderId:2, customerName:"Bob", totalPrice:1500.0),
            new Order(orderId:3, customerName:"Charlie", totalPrice:4000.0)
        };

        System.out.println(x:"Bubble Sort:");
        bubbleSort(orders);
        for (Order o : orders) System.out.println(o);

        orders = new Order[]{
            new Order(orderId:1, customerName:"Alice", totalPrice:2500.0),
            new Order(orderId:2, customerName:"Bob", totalPrice:1500.0),
            new Order(orderId:3, customerName:"Charlie", totalPrice:4000.0)
        };

        System.out.println(x:"Quick Sort:");
        quickSort(orders, low:0, orders.length - 1);
        for (Order o : orders) System.out.println(o);
    }
}
```

Output :

```
PS C:\Users\Admin\Desktop\JAVA QS 2>  & 'C:\Java\jdk-23\bin\java.exe' '-XX:+ShowCodeDetai
1bb8b2db0a4c0c4e1ab28ea22ab48b8\redhat.java\jdt_ws\JAVA QS 2_bd9184e0\bin' 'OrderSorting'
d9184e0\x5cbin' 'OrderSorting' ;02e3a7ca-414e-45bd-9741-354fad3f22d5Bubble Sort:
[2, Bob, Total: 1500.0]
[1, Alice, Total: 2500.0]
[3, Charlie, Total: 4000.0]
Quick Sort:
[2, Bob, Total: 1500.0]
[1, Alice, Total: 2500.0]
[3, Charlie, Total: 4000.0]
PS C:\Users\Admin\Desktop\JAVA QS 2>
```

Exercise 4 : Employee Management System

Code :

EmployeeManagement.java > ...

```java
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 String toString() {
        return "[" + employeeId + ", " + name + ", " + position + ", Salary: " + salary + "]";
    }
}

public class EmployeeManagement {
    static final int MAX = 100;
    static Employee[] employees = new Employee[MAX];
    static int count = 0;

    static void addEmployee(Employee emp) {
        if (count < MAX) {
            employees[count++] = emp;
        }
    }

    static void traverseEmployees() {
        for (int i = 0; i < count; i++) {
            System.out.println(employees[i]);
        }
    }

    public static void main(String[] args) {
        addEmployee(new Employee(employeeId:1, name:"John", position:"Manager", salary:60000.0));
        addEmployee(new Employee(employeeId:2, name:"Jane", position:"Developer", salary:50000.0));
        addEmployee(new Employee(employeeId:1, name:"John", position:"Manager", salary:60000.0));
        addEmployee(new Employee(employeeId:1, name:"John", position:"Manager", salary:60000.0));

        traverseEmployees();
    }
}
```

Output :

```
PS C:\Users\Admin\Desktop\JAVA QS 2> javac EmployeeManagement.java
PS C:\Users\Admin\Desktop\JAVA QS 2> java EmployeeManagement.java
[1, John, Manager, Salary: 60000.0]
[2, Jane, Developer, Salary: 50000.0]
[1, John, Manager, Salary: 60000.0]
[1, John, Manager, Salary: 60000.0]
PS C:\Users\Admin\Desktop\JAVA QS 2>
```

Exercise 5 : Task management system

Code :

```java
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;
    }

    public String toString() {
        return "[" + taskId + ", " + taskName + ", " + status + "]";
    }
}

public class TaskManagement {
    static Task head = null;

    static void addTask(int id, String name, String status) {
        Task newTask = new Task(id, name, status);
        newTask.next = null;
        if (head == null) head = newTask;
        else {
            Task temp = head;
            while (temp.next != null) temp = temp.next;
            temp.next = newTask;
        }
    }

    static Task searchTask(int id) {
        Task curr = head;
        while (curr != null) {
            if (curr.taskId == id) return curr;
            curr = curr.next;
        }
        return null;
    }

    static void deleteTask(int id) {
        if (head == null) return;
        if (head.taskId == id) {
            head = head.next;
```

```java
    static void deleteTask(int id) {
        if (head == null) return;
        if (head.taskId == id) {
            head = head.next;
            return;
        }
        Task prev = head, curr = head.next;
        while (curr != null) {
            if (curr.taskId == id) {
                prev.next = curr.next;
                return;
            }
            prev = curr;
            curr = curr.next;
        }
    }

    static void traverseTasks() {
        Task curr = head;
        while (curr != null) {
            System.out.println(curr);
            curr = curr.next;
        }
    }

    Run main | Debug main | Run | Debug
    public static void main(String[] args) {
        addTask(id:1, name:"Design", status:"Pending");
        addTask(id:2, name:"Coding", status:"In Progress");
        addTask(id:3, name:"Testing", status:"Pending");

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

        Task found = searchTask(id:2);
        if (found != null) System.out.println("Found: " + found);

        deleteTask(id:1);

        System.out.println(x:"After Deletion:");
        traverseTasks();
    }
}
```

Output :

```
PS C:\Users\Admin\Desktop\JAVA QS 2> javac TaskManagement.java
PS C:\Users\Admin\Desktop\JAVA QS 2> java TaskManagement.java
All Tasks:
[1, Design, Pending]
[2, Coding, In Progress]
[3, Testing, Pending]
Found: [2, Coding, In Progress]
After Deletion:
[2, Coding, In Progress]
[3, Testing, Pending]
PS C:\Users\Admin\Desktop\JAVA QS 2>
```

Exercise 6 : Library Management System

Code :

```java
import java.util.Arrays;
import java.util.Comparator;

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;
    }

    public String toString() {
        return title + " by " + author;
    }
}

public class LibrarySearch {

    // Linear search by title
    static Book linearSearch(Book[] books, String title) {
        for (Book book : books) {
            if (book.title.equalsIgnoreCase(title)) {
                return book;
            }
        }
        return null;
    }

    // Binary search by title
    static Book binarySearch(Book[] books, String title) {
        int low = 0, high = books.length - 1;
        while (low <= high) {
            int mid = (low + high) / 2;
```

```
            int cmp = books[mid].title.compareToIgnoreCase(title);
            if (cmp == 0) return books[mid];
            else if (cmp < 0) low = mid + 1;
            else high = mid - 1;
        }
        return null;
    }

    Run main | Debug main | Run | Debug
    public static void main(String[] args) {
        Book[] books = {
            new Book(bookId:1, title:"Data Structures", author:"Author A"),
            new Book(bookId:2, title:"Operating Systems", author:"Author B"),
            new Book(bookId:3, title:"Algorithms", author:"Author C")
        };

        System.out.println(x:"Linear Search:");
        Book linearResult = linearSearch(books, title:"Algorithms");
        if (linearResult != null)
            System.out.println("Found: " + linearResult);

        // Sort books by title for binary search
        Arrays.sort(books, Comparator.comparing(b -> b.title.toLowerCase()));

        System.out.println(x:"Binary Search:");
        Book binaryResult = binarySearch(books, title:"Data Structures");
        if (binaryResult != null)
            System.out.println("Found: " + binaryResult.title);
    }
}
```

Output:

```
PS C:\Users\Admin\Desktop\JAVA QS 2> javac LibrarySearch.java
PS C:\Users\Admin\Desktop\JAVA QS 2> java LibrarySearch.java
Linear Search:
Found: Algorithms by Author C
Binary Search:
Found: Data Structures
PS C:\Users\Admin\Desktop\JAVA QS 2>
```

Exercise 7 : Financial Forecasting

Code :

```java
public class FinancialForecast {

    static double predictRecursive(double presentValue, double growthRate, int years) {
        if (years == 0)
            return presentValue;
        return predictRecursive(presentValue, growthRate, years - 1) * (1 + growthRate);
    }

    static double predictIterative(double presentValue, double growthRate, int years) {
        for (int i = 0; i < years; i++) {
            presentValue *= (1 + growthRate);
        }
        return presentValue;
    }

    Run main | Debug main | Run | Debug
    public static void main(String[] args) {
        double presentValue = 10000;    // Base value
        double growthRate = 0.10;       // 10% growth
        int years = 3;

        double futureRecursive = predictRecursive(presentValue, growthRate, years);
        double futureIterative = predictIterative(presentValue, growthRate, years);

        System.out.printf(format:"Recursive Prediction after %d years: %.2f\n", years, futureRecursive);
        System.out.printf(format:"Iterative Prediction after %d years: %.2f\n", years, futureIterative);
    }
}
```

Output :

```
PS C:\Users\Admin\Desktop\JAVA QS 2> javac FinancialForecast.java
PS C:\Users\Admin\Desktop\JAVA QS 2> java FinancialForecast.java
Recursive Prediction after 3 years: 13310.00
Iterative Prediction after 3 years: 13310.00
PS C:\Users\Admin\Desktop\JAVA QS 2>
```