Digital Nurture 4.0

Deep Skilling Handbook

Java FSE -Solution

WEEK - 1

Data Structures And Algorithms

Exercise 1: Inventory Management System

Program:

```
package DataStructureAndAlgorthms;
import java.util.*;
public class InventoryManagement {
    static class GroceryItem {
        String itemId;
        String itemName;
        int quantity;
        double price;

        GroceryItem(String itemId, String itemName, int quantity, double price) {
            this.itemId = itemId;
            this.itemName = itemName;
            this.quantity = quantity;
            this.price = price;
        }

        public String toString() {
            return itemId + " | " + itemName + " | " + quantity + " | " + price;
        }
    }
    static class Inventory {
        Map<String, GroceryItem> items = new HashMap<>();
        void addItem(GroceryItem item) {
            items.put(item.itemId, item);
        }
```

```
    void updateItem(String itemId, int quantity, double price) {
        if (items.containsKey(itemId)) {
            GroceryItem item = items.get(itemId);
            item.quantity = quantity;
            item.price = price;
        }
    }

    void deleteItem(String itemId) {
        items.remove(itemId);
    }

    void printInventory() {
        for (GroceryItem item : items.values()) {
            System.out.println(item);
        }
    }
}

public static void main(String[] args) {
    Inventory shop = new Inventory();
    GroceryItem i1 = new GroceryItem("G001", "Sugar", 50, 45.0);
    GroceryItem i2 = new GroceryItem("G002", "Milk", 30, 25.5);
    GroceryItem i3 = new GroceryItem("G003", "Rice", 100, 60.0);
    shop.addItem(i1);
    shop.addItem(i2);
    shop.addItem(i3);
```

```
        shop.printInventory();

        shop.updateItem("G002", 40, 27.0);
        shop.deleteItem("G001");

        System.out.println("After update and delete:");
        shop.printInventory();
    }
}
```

Output:

```
 1 package DataStructureAndAlgorthms;
 2
 3 import java.util.*;
 4
 5 public class InventoryManagement {
 6
 7     static class GroceryItem {
 8         String itemId;
 9         String itemName;
10         int quantity;
11         double price;
12
13         GroceryItem(String itemId, String itemName, int quantity, double price) {
14             this.itemId = itemId;
15             this.itemName = itemName;
16             this.quantity = quantity;
17             this.price = price;
18         }
19
20         public String toString() {
21             return itemId + " | " + itemName + " | " + quantity + " | " + price;
22         }
23     }
24
25     static class Inventory {
26         Map<String, GroceryItem> items = new HashMap<>();
27
28         void addItem(GroceryItem item) {
29             items.put(item.itemId, item);
30         }
31
```

Problems Javadoc Declaration Console × Debug

```
<terminated> InventoryManagement [Java Application] C:\Users\Fyzal\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.2.v20240123-084
G002 | Milk | 30 | 25.5
G001 | Sugar | 50 | 45.0
G003 | Rice | 100 | 60.0
After update and delete:
G002 | Milk | 40 | 27.0
G003 | Rice | 100 | 60.0
```

Exercise 2: E-commerce Platform Search Function

PROGRAM:

```
package DataStructureAndAlgorthms;
import java.util.*;

public class EcommerceSearch {
    static class Product {
        String productId;
        String productName;
        String category;

        Product(String productId, String productName, String category) {
            this.productId = productId;
            this.productName = productName;
            this.category = category;
        }
        public String toString() {
            return productId + " | " + productName + " | " + category;
        }
    }
    static class SearchEngine {
        Product[] products;
        SearchEngine(Product[] products) {
            this.products = products;
        }

        Product linearSearch(String name) {
            for (Product p : products) {
```

```
            if (p.productName.equalsIgnoreCase(name)) {
                return p;
            }
        }
        return null;
    }

    Product binarySearch(String name) {
        Arrays.sort(products, Comparator.comparing(p -> p.productName.toLowerCase()));
        int left = 0, right = products.length - 1;
        while (left <= right) {
            int mid = (left + right) / 2;
            int cmp = products[mid].productName.compareToIgnoreCase(name);
            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[] productList = {
        new Product("Item 1", "Laptop", "Electronics"),
        new Product("Item 2", "Shampoo", "Personal Care"),
        new Product("Item 3", "T-Shirt", "Clothing"),
        new Product("Item 4", "Book", "Stationery"),
        new Product("Item 5", "Phone", "Electronics")
    };
```

```
        SearchEngine searchEngine = new SearchEngine(productList);

        Product result1 = searchEngine.linearSearch("T-Shirt");
        System.out.println("Linear Search Result: " + (result1 != null ? result1 : "Not Found"));

        Product result2 = searchEngine.binarySearch("Phone");
        System.out.println("Binary Search Result: " + (result2 != null ? result2 : "Not Found"));
    }
}
```

OUTPUT:

```
 1 package DataStructureAndAlgorthms;
 2
 3 import java.util.*;
 4
 5 public class EcommerceSearch {
 6
 7     static class Product {
 8         String productId;
 9         String productName;
10         String category;
11
12         Product(String productId, String productName, String category) {
13             this.productId = productId;
14             this.productName = productName;
15             this.category = category;
16         }
17
18         public String toString() {
19             return productId + " | " + productName + " | " + category;
20         }
21     }
22
23     static class SearchEngine {
24         Product[] products;
25
26         SearchEngine(Product[] products) {
27             this.products = products;
28         }
29
30         Product linearSearch(String name) {
31             for (Product p : products) {
```

```
Problems  Javadoc  Declaration  Console ×  Debug
<terminated> EcommerceSearch [Java Application] C:\Users\Fyzal\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.2.v20240123-08
Linear Search Result: Item 3 | T-Shirt | Clothing
Binary Search Result: Item 5 | Phone | Electronics
```

Exercise 3: Sorting Customer Orders

PROGRAM:

```
package DataStructureAndAlgorthms;

class Order {
    String id;
    String name;
    double price;

    Order(String id, String name, double price) {
        this.id = id;
        this.name = name;
        this.price = price;
    }

    void show() {
        System.out.println(id + " - " + name + " - " + price);
    }
}

public class CustomerOrderSorting {

    static void bubble(Order[] arr) {
        int n = arr.length;
        for (int i = 0; i < n - 1; i++) {
            for (int j = 0; j < n - i - 1; j++) {
                if (arr[j].price < arr[j + 1].price) {
                    Order temp = arr[j];
```

```
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}

static void quick(Order[] arr, int l, int h) {
    if (l < h) {
        int p = part(arr, l, h);
        quick(arr, l, p - 1);
        quick(arr, p + 1, h);
    }
}

static int part(Order[] arr, int l, int h) {
    double piv = arr[h].price;
    int i = l - 1;
    for (int j = l; j < h; j++) {
        if (arr[j].price < piv) {
            i++;
            Order tmp = arr[i];
            arr[i] = arr[j];
            arr[j] = tmp;
        }
    }
    Order tmp = arr[i + 1];
    arr[i + 1] = arr[h];
    arr[h] = tmp;
    return i + 1;
```

```
    }

    static void showList(Order[] arr) {
        for (Order o : arr) o.show();
    }

    public static void main(String[] args) {
        Order[] original = {
            new Order("O1", "Fyzal", 2000),
            new Order("O2", "Kiran", 1800),
            new Order("O3", "Fazil", 2500),
            new Order("O4", "Arun", 1600),
            new Order("O5", "Yokes", 2200)
        };

        Order[] bubbleList = original.clone();
        Order[] quickList = original.clone();

        System.out.println("Bubble Sort (Descending):");
        bubble(bubbleList);
        showList(bubbleList);

        System.out.println("\nQuick Sort (Ascending):");
        quick(quickList, 0, quickList.length - 1);
        showList(quickList);
    }
}
```

OUTPUT:

```java
package DataStructureAndAlgorthms;
class Order {
    String id;
    String name;
    double price;

    Order(String id, String name, double price) {
        this.id = id;
        this.name = name;
        this.price = price;
    }

    void show() {
        System.out.println(id + " - " + name + " - " + price);
    }
}

public class CustomerOrderSorting {

    static void bubble(Order[] arr) {
```

Problems Javadoc Declaration Console × Debug

```
<terminated> CustomerOrderSorting [Java Application] C:\Users\Fyzal\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32
Bubble Sort (Descending):
O3 - Fazil - 2500.0
O5 - Yokes - 2200.0
O1 - Fyzal - 2000.0
O2 - Kiran - 1800.0
O4 - Arun - 1600.0

Quick Sort (Ascending):
O4 - Arun - 1600.0
O2 - Kiran - 1800.0
O1 - Fyzal - 2000.0
O5 - Yokes - 2200.0
O3 - Fazil - 2500.0
```

Exercise 4: Employee Management System

PROGRAM:

```
package DataStructureAndAlgorthms;

class Employee {
    String employeeId;
    String name;
    String position;
    double salary;

    Employee(String employeeId, String name, String position, double salary) {
        this.employeeId = employeeId;
        this.name = name;
        this.position = position;
        this.salary = salary;
    }

    void display() {
        System.out.println(employeeId + " - " + name + " - " + position + " - Rs." + salary);
    }
}

public class EmployeeManagementSystem {
    static Employee[] employees = new Employee[10];
    static int count = 0;

    static void addEmployee(Employee e) {
        if (count < employees.length) {
            employees[count++] = e;
```

```
    }
}

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

static void searchEmployee(String id) {
    for (int i = 0; i < count; i++) {
        if (employees[i].employeeId.equals(id)) {
            employees[i].display();
            return;
        }
    }
    System.out.println("Employee not found");
}

static void deleteEmployee(String id) {
    for (int i = 0; i < count; i++) {
        if (employees[i].employeeId.equals(id)) {
            for (int j = i; j < count - 1; j++) {
                employees[j] = employees[j + 1];
            }
            employees[--count] = null;
            return;
        }
    }
    System.out.println("Employee not found to delete");
```

```
    }

    public static void main(String[] args) {
        addEmployee(new Employee("E001", "Fyzal", "Developer", 50000));
        addEmployee(new Employee("E002", "Kiran", "Tester", 42000));
        addEmployee(new Employee("E003", "Fazil", "Support", 38000));
        addEmployee(new Employee("E004", "Arun", "HR", 46000));
        addEmployee(new Employee("E005", "Yokes", "Designer", 49000));

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

        System.out.println("\nSearching for E003:");
        searchEmployee("E003");

        System.out.println("\nDeleting E002:");
        deleteEmployee("E002");

        System.out.println("\nEmployees after deletion:");
        traverseEmployees();
    }
}
```

OUTPUT:

```java
package DataStructureAndAlgorthms;

class Employee {
    String employeeId;
    String name;
    String position;
    double salary;

    Employee(String employeeId, String name, String position, double salary) {
        this.employeeId = employeeId;
        this.name = name;
        this.position = position;
        this.salary = salary;
    }

    void display() {
        System.out.println(employeeId + " - " + name + " - " + position + " - Rs." + salary);
    }
}
```

Problems Javadoc Declaration Console × Debug

```
<terminated> EmployeeManagementSystem [Java Application] C:\Users\Fyzal\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.2.v2024
All Employees:
E001 - Fyzal - Developer - Rs.50000.0
E002 - Kiran - Tester - Rs.42000.0
E003 - Fazil - Support - Rs.38000.0
E004 - Arun - HR - Rs.46000.0
E005 - Yokes - Designer - Rs.49000.0

Searching for E003:
E003 - Fazil - Support - Rs.38000.0

Deleting E002:

Employees after deletion:
E001 - Fyzal - Developer - Rs.50000.0
E003 - Fazil - Support - Rs.38000.0
E004 - Arun - HR - Rs.46000.0
E005 - Yokes - Designer - Rs.49000.0
```

Exercise 5: Task Management System

PROGRAM:

```
package DataStructureAndAlgorthms;

class Task {
    String id;
    String name;
    String status;
    Task next;

    Task(String id, String name, String status) {
        this.id = id;
        this.name = name;
        this.status = status;
        this.next = null;
    }

    void show() {
        System.out.println(id + " | " + name + " | " + status);
    }
}

public class TaskList {
    Task head = null;

    void add(String id, String name, String status) {
        Task t = new Task(id, name, status);
        if (head == null) {
            head = t;
```

```
    } else {
        Task curr = head;
        while (curr.next != null) {
            curr = curr.next;
        }
        curr.next = t;
    }
}

void viewAll() {
    Task curr = head;
    while (curr != null) {
        curr.show();
        curr = curr.next;
    }
}

void find(String id) {
    Task curr = head;
    while (curr != null) {
        if (curr.id.equals(id)) {
            curr.show();
            return;
        }
        curr = curr.next;
    }
    System.out.println("Task not found");
}

void remove(String id) {
```

```
    if (head == null) return;

    if (head.id.equals(id)) {
        head = head.next;
        return;
    }

    Task curr = head;
    while (curr.next != null) {
        if (curr.next.id.equals(id)) {
            curr.next = curr.next.next;
            return;
        }
        curr = curr.next;
    }
    System.out.println("Task not found to remove");
}

public static void main(String[] args) {
    TaskList tasks = new TaskList();

    tasks.add("Task 1", "Submit DSA Assignment", "Pending");
    tasks.add("Task 2", "Complete Java Project", "In Progress");
    tasks.add("Task 3", "Database PPT Prep", "Pending");
    tasks.add("Task 4", "Cloud Lab Writeup", "Completed");
    tasks.add("Task 5", "AI Paper Submission", "Pending");

    System.out.println("All Tasks:");
    tasks.viewAll();
```

```
        System.out.println("\nSearch for Item 3:");
        tasks.find("Task 3");

        System.out.println("\nRemoving Item 2:");
        tasks.remove("Task 2");

        System.out.println("\nUpdated Task List:");
        tasks.viewAll();
    }
}
```

OUTPUT:

```
package DataStructureAndAlgorthms;

class Task {
    String id;
    String name;
    String status;
    Task next;

    Task(String id, String name, String status) {
        this.id = id;
        this.name = name;
        this.status = status;
        this.next = null;
    }

    void show() {
        System.out.println(id + " | " + name + " | " + status);
    }
}
```

Problems Javadoc Declaration Console × Debug

```
<terminated> TaskList [Java Application] C:\Users\Fyzal\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.2.v20240123-08
All Tasks:
Task 1 | Submit DSA Assignment | Pending
Task 2 | Complete Java Project | In Progress
Task 3 | Database PPT Prep | Pending
Task 4 | Cloud Lab Writeup | Completed
Task 5 | AI Paper Submission | Pending

Search for Item 3:
Task 3 | Database PPT Prep | Pending

Removing Item 2:

Updated Task List:
Task 1 | Submit DSA Assignment | Pending
Task 3 | Database PPT Prep | Pending
Task 4 | Cloud Lab Writeup | Completed
Task 5 | AI Paper Submission | Pending
```

Exercise 6: Library Management System

PROGRAM:

```
package DataStructureAndAlgorthms;

import java.util.*;

class Book {
    String bookId;
    String title;
    String author;

    public Book(String bookId, String title, String author) {
        this.bookId = bookId;
        this.title = title;
        this.author = author;
    }

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

public class LibraryManagementSystem {
    static List<Book> books = new ArrayList<>();

    public static void linearSearch(String title) {
        boolean found = false;
        for (Book b : books) {
            if (b.title.equalsIgnoreCase(title)) {
```

```
            System.out.println("Linear: " + b);
            found = true;
            break;
        }
    }
    if (!found) {
        System.out.println("Not Found (Linear)");
    }
}

public static void binarySearch(String title) {
    books.sort(Comparator.comparing(b -> b.title.toLowerCase()));
    int low = 0, high = books.size() - 1;
    while (low <= high) {
        int mid = (low + high) / 2;
        Book b = books.get(mid);
        int cmp = b.title.compareToIgnoreCase(title);
        if (cmp == 0) {
            System.out.println("Binary: " + b);
            return;
        } else if (cmp < 0) {
            low = mid + 1;
        } else {
            high = mid - 1;
        }
    }
    System.out.println("Not Found (Binary)");
}

public static void main(String[] args) {
```

```
        books.add(new Book("Book 1", "Harry Potter", "J.K. Rowling"));
        books.add(new Book("Book 2", "Ponniyin Selvan", "Kalki Krishnamurthy"));
        books.add(new Book("Book 3", "Thirukkural", "Thiruvalluvar"));
        books.add(new Book("Book 4", "Parthiban Kanavu", "Kalki Krishnamurthy"));
        books.add(new Book("Book 5", "Life of Pi", "Yann Martel"));

        linearSearch("Harry Potter");
        binarySearch("Thirukkural");
    }
}
```

OUTPUT:

```
package DataStructureAndAlgorthms;

import java.util.*;

class Book {
    String bookId;
    String title;
    String author;

    public Book(String bookId, String title, String author) {
        this.bookId = bookId;
        this.title = title;
        this.author = author;
    }

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

public class LibraryManagementSystem {
    static List<Book> books = new ArrayList<>();

    public static void linearSearch(String title) {
        boolean found = false;
        for (Book b : books) {
            if (b.title.equalsIgnoreCase(title)) {
                System.out.println("Linear: " + b);
```

Problems Javadoc Declaration Console × Debug

```
<terminated> LibraryManagementSystem [Java Application] C:\Users\Fyzal\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.w
Linear: Book 1 - Harry Potter - J.K. Rowling
Binary: Book 3 - Thirukkural - Thiruvalluvar
```

Exercise 7: Financial Forecasting

PROGRAM:

```
package DataStructureAndAlgorthms;

public class FinancialForecast {

    public static double forecast(double amount, double rate, int year) {
        if (year == 0) {
            return amount;
        } else {
            double updated = amount * (1 + rate);
            return forecast(updated, rate, year - 1);
        }
    }

    public static void main(String[] args) {
        double value = 10000;
        double growth = 0.3;
        int years = 7;

        double result = forecast(value, growth, years);
        System.out.println("After " + years + " years: Rs." + (int)result);
    }
}
```

OUTPUT:

```java
package DataStructureAndAlgorthms;

public class FinancialForecast {

    public static double forecast(double amount, double rate, int year) {
        if (year == 0) {
            return amount;
        } else {
            double updated = amount * (1 + rate);
            return forecast(updated, rate, year - 1);
        }
    }

    public static void main(String[] args) {
        double value = 10000;
        double growth = 0.3;
        int years = 7;

        double result = forecast(value, growth, years);
        System.out.println("After " + years + " years: Rs." + (int)result);
    }
}
```

Problems Javadoc Declaration Console Debug

<terminated> FinancialForecast [Java Application] C:\Users\Fyzal\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.2.v202

```
After 7 years: Rs.62748
```