**Week 1-Algorithms and Data Structures HandsOn**

**Program:**

// Exercise 1: Inventory Management System

import java.util.\*;

class Product {

int productId;

String productName;

int quantity;

double price;

Product(int productId, String productName, int quantity, double price) {

this.productId = productId;

this.productName = productName;

this.quantity = quantity;

this.price = price;

}

public String toString() {

return productId + " - " + productName + " - Qty: " + quantity + " - Rs." + price;

}

}

class Inventory {

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

public static void addProduct(Product p) {

inventory.put(p.productId, p);

}

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

if (inventory.containsKey(id)) {

Product p = inventory.get(id);

p.quantity = newQty;

p.price = newPrice;

}

}

public static void deleteProduct(int id) {

inventory.remove(id);

}

public static void displayInventory() {

for (Product p : inventory.values())

System.out.println(p);

}

}

// Exercise 2: E-commerce Platform Search Function

class EcomProduct {

int productId;

String productName;

String category;

EcomProduct(int productId, String productName, String category) {

this.productId = productId;

this.productName = productName;

this.category = category;

}

public String toString() {

return productName + " [" + category + "]";

}

}

class EcomSearch {

static EcomProduct[] products = {

new EcomProduct(1, "Laptop", "Electronics"),

new EcomProduct(2, "Shoes", "Fashion"),

new EcomProduct(3, "Phone", "Electronics"),

new EcomProduct(4, "Bag", "Travel")

};

public static EcomProduct linearSearch(String name) {

for (EcomProduct p : products) {

if (p.productName.equalsIgnoreCase(name))

return p;

}

return null;

}

public static EcomProduct binarySearch(String name) {

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

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

while (left <= right) {

int mid = (left + right) / 2;

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

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

else if (cmp < 0) right = mid - 1;

else left = mid + 1;

}

return null;

}

}

// Exercise 3: Sorting Customer Orders

class Order {

int orderId;

String customerName;

double totalPrice;

Order(int id, String name, double price) {

orderId = id;

customerName = name;

totalPrice = price;

}

public String toString() {

return customerName + " - Rs." + totalPrice;

}

}

class OrderSorting {

static void bubbleSort(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].totalPrice > arr[j + 1].totalPrice) {

Order temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

static void quickSort(Order[] arr, int low, int high) {

if (low < high) {

int pi = partition(arr, low, high);

quickSort(arr, low, pi - 1);

quickSort(arr, pi + 1, high);

}

}

static int partition(Order[] arr, int low, int high) {

double pivot = arr[high].totalPrice;

int i = (low - 1);

for (int j = low; j < high; j++) {

if (arr[j].totalPrice < pivot) {

i++;

Order temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

Order temp = arr[i + 1];

arr[i + 1] = arr[high];

arr[high] = temp;

return i + 1;

}

}

// Exercise 4: Employee Management System

class Employee {

int employeeId;

String name, position;

double salary;

Employee(int id, String name, String pos, double salary) {

this.employeeId = id;

this.name = name;

this.position = pos;

this.salary = salary;

}

public String toString() {

return name + " - " + position + " - Rs." + salary;

}

}

class EmployeeSystem {

static Employee[] employees = new Employee[10];

static int count = 0;

static void add(Employee e) {

employees[count++] = e;

}

static void delete(int id) {

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

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

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

employees[j] = employees[j + 1];

count--;

break;

}

}

}

static void search(int id) {

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

if (employees[i].employeeId == id)

System.out.println("Found: " + employees[i]);

}

static void traverse() {

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

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

}

}

// Exercise 5: Task Management System

class Task {

int taskId;

String taskName;

String status;

Task next;

Task(int id, String name, String status) {

this.taskId = id;

this.taskName = name;

this.status = status;

this.next = null;

}

}

class TaskList {

static Task head = null;

static void addTask(Task t) {

if (head == null)

head = t;

else {

Task temp = head;

while (temp.next != null)

temp = temp.next;

temp.next = t;

}

}

static void traverse() {

Task temp = head;

while (temp != null) {

System.out.println(temp.taskName + " - " + temp.status);

temp = temp.next;

}

}

static void delete(int id) {

Task temp = head, prev = null;

if (temp != null && temp.taskId == id) {

head = temp.next;

return;

}

while (temp != null && temp.taskId != id) {

prev = temp;

temp = temp.next;

}

if (temp != null)

prev.next = temp.next;

}

}

// Exercise 6: Library Management System

class Book {

int bookId;

String title, author;

Book(int id, String t, String a) {

bookId = id;

title = t;

author = a;

}

public String toString() {

return title + " by " + author;

}

}

class Library {

static Book[] books = {

new Book(1, "Java Basics", "James"),

new Book(2, "Python 101", "Guido"),

new Book(3, "C++ Primer", "Bjarne")

};

static Book linearSearch(String title) {

for (Book b : books)

if (b.title.equalsIgnoreCase(title))

return b;

return null;

}

static Book binarySearch(String title) {

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

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

while (low <= high) {

int mid = (low + high) / 2;

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

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

else if (cmp < 0) high = mid - 1;

else low = mid + 1;

}

return null;

}

}

// Exercise 7: Financial Forecasting

class Forecast {

static double predictValue(double current, double growthRate, int years) {

if (years == 0)

return current;

return predictValue(current \* (1 + growthRate), growthRate, years - 1);

}

}

public class Main {

public static void main(String[] args) {

// Exercise 1

Inventory.addProduct(new Product(101, "Mouse", 20, 299.99));

Inventory.addProduct(new Product(102, "Keyboard", 15, 499.99));

Inventory.updateProduct(101, 25, 289.99);

Inventory.deleteProduct(102);

Inventory.displayInventory();

// Exercise 2

System.out.println("Linear: " + EcomSearch.linearSearch("Phone"));

System.out.println("Binary: " + EcomSearch.binarySearch("Phone"));

// Exercise 3

Order[] orders = {

new Order(1, "Alice", 1500),

new Order(2, "Bob", 800),

new Order(3, "Carol", 1200)

};

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

for (Order o : orders)

System.out.println(o);

// Exercise 4

EmployeeSystem.add(new Employee(1, "John", "Manager", 50000));

EmployeeSystem.add(new Employee(2, "Jane", "Engineer", 40000));

EmployeeSystem.traverse();

EmployeeSystem.search(2);

EmployeeSystem.delete(1);

EmployeeSystem.traverse();

// Exercise 5

TaskList.addTask(new Task(1, "Design DB", "Pending"));

TaskList.addTask(new Task(2, "Code UI", "Ongoing"));

TaskList.traverse();

TaskList.delete(1);

TaskList.traverse();

// Exercise 6

System.out.println("Linear: " + Library.linearSearch("Python 101"));

System.out.println("Binary: " + Library.binarySearch("Python 101"));

// Exercise 7

double future = Forecast.predictValue(10000, 0.10, 3);

System.out.println("Future Value: Rs." + future);

}

}

Output:

A screenshot of a computer program

AI-generated content may be incorrect.