**WEEK 1 - ALGORITHMS AND DATA STRUCTURE**

Super Set ID: 6427473

Name: SANJAI JAIVARDHAN B

1. **Inventory Management System**

**Program:**

* Inventory.java:

**package** inventory;

**import** java.util.HashMap;

**public** **class** Inventory {

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

**public** **void** addProduct(Product product) {

**if** (products.containsKey(product.getProductId())) {

System.***out***.println("Product with ID " + product.getProductId() + " already exists.");

} **else** {

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

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

}

}

**public** **void** updateProduct(**int** productId, **int** newQuantity, **double** newPrice) {

Product p = products.get(productId);

**if** (p != **null**) {

p.setQuantity(newQuantity);

p.setPrice(newPrice);

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

} **else** {

System.***out***.println("Product ID " + productId + " not found.");

}

}

**public** **void** deleteProduct(**int** productId) {

Product removed = products.remove(productId);

**if** (removed != **null**) {

System.***out***.println("Deleted: " + removed);

} **else** {

System.***out***.println("Product ID " + productId + " not found.");

}

}

**public** **void** displayInventory() {

System.***out***.println("\nInventory:");

**if** (products.isEmpty()) {

System.***out***.println("No products available.");

} **else** {

**for** (Product p : products.values()) {

System.***out***.println(p);

}

}

}

}

* Product.java:

**package inventory;**

**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) {**

**if (quantity < 0 || price < 0) throw new IllegalArgumentException("Quantity and price must be non-negative");**

**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) {**

**if (quantity >= 0) this.quantity = quantity;**

**}**

**public void setPrice(double price) {**

**if (price >= 0) this.price = price;**

**}**

**@Override**

**public String toString() {**

**return "Product ID: " + productId + ", Name: " + productName +**

**", Qty: " + quantity + ", Price: ₹" + price;**

**}**

**}**

* TestInventory.java:

**package inventory;**

**public class TestInventory {**

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

**Inventory inventory = new Inventory();**

**Product p1 = new Product(1001, "TV", 15, 45000.0);**

**Product p2 = new Product(1002, "Fan", 40, 2000.0);**

**Product p3 = new Product(1003, "Fridge", 10, 30000.0);**

**inventory.addProduct(p1);**

**inventory.addProduct(p2);**

**inventory.addProduct(p3);**

**inventory.displayInventory();**

**inventory.addProduct(new Product(1001, "Duplicate TV", 5, 50000.0));**

**inventory.updateProduct(1002, 45, 2100.0);**

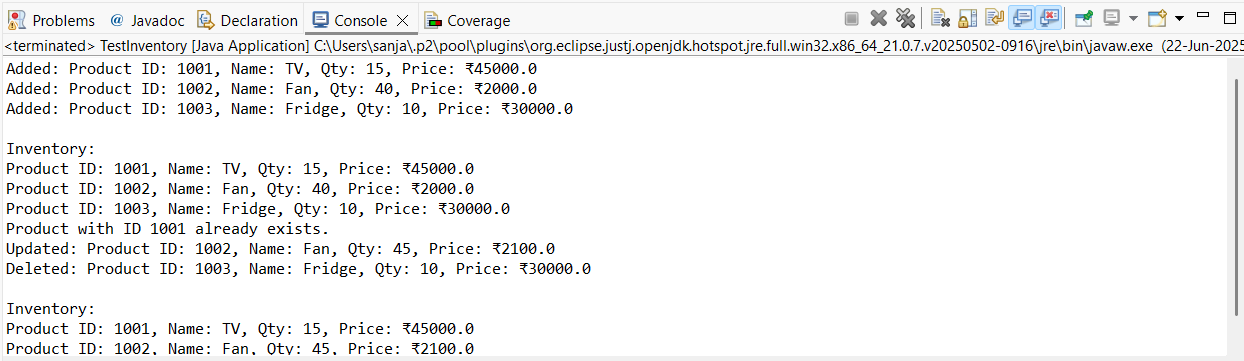
**inventory.deleteProduct(1003);**

**inventory.displayInventory();**

**}**

**}**

**Output:**

****

1. **Ecommerce Platform Search Function**

**Program:**

* Product.java:

**package** ecommerce;

**public** **class** Product {

**private** **int** productId;

**private** String productName;

**private** String category;

**public** Product(**int** productId, String productName, String category) {

**this**.productId = productId;

**this**.productName = productName;

**this**.category = category;

}

**public** String getProductName() {

**return** productName;

}

@Override

**public** String toString() {

**return** "[" + productId + "] " + productName + " - " + category;

}

}

* TestSearch.java:

**package ecommerce;**

**public class TestSearch {**

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

**Product[] products = {**

**new Product(101, "iPhone", "Electronics"),**

**new Product(102, "Samsung TV", "Electronics"),**

**new Product(103, "Nike Shoes", "Footwear"),**

**new Product(104, "Dell Laptop", "Computers"),**

**new Product(105, "Canon Camera", "Photography")**

**};**

**System.*out*.println("🔎 Linear Search for 'Nike Shoes':");**

**Product result1 = SearchEngine.*linearSearch*(products, "Nike Shoes");**

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

**System.*out*.println("\n🔍 Binary Search for 'Dell Laptop':");**

**SearchEngine.*sortByName*(products);**

**Product result2 = SearchEngine.*binarySearch*(products, "Dell Laptop");**

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

**}**

**}**

* SearchEngine.java:

**package ecommerce;**

**import java.util.Arrays;**

**import java.util.Comparator;**

**public class SearchEngine {**

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

**for (Product p : products) {**

**if (p.getProductName().equalsIgnoreCase(name)) {**

**return p;**

**}**

**}**

**return null;**

**}**

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

**int low = 0, high = products.length - 1;**

**while (low <= high) {**

**int mid = (low + high) / 2;**

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

**if (cmp == 0)**

**return products[mid];**

**else if (cmp < 0)**

**low = mid + 1;**

**else**

**high = mid - 1;**

**}**

**return null;**

**}**

**public static void sortByName(Product[] products) {**

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

**}**

**}**

**Output:**

**A screenshot of a computer

AI-generated content may be incorrect.**

1. **Sorting Customer Orders**
   * **Program:**
   * Order.java:

**package** sorting;

**public** **class** Order {

**private** **int** orderId;

**private** String customerName;

**private** **double** totalPrice;

**public** Order(**int** orderId, String customerName, **double** totalPrice) {

**this**.orderId = orderId;

**this**.customerName = customerName;

**this**.totalPrice = totalPrice;

}

**public** **double** getTotalPrice() {

**return** totalPrice;

}

@Override

**public** String toString() {

**return** "[" + orderId + "] " + customerName + " - ₹" + totalPrice;

}

}

* + Sorter.java:

**package** sorting;

**public** **class** Sorter {

**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].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** 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].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;

}

**public** **static** **void** printOrders(Order[] orders) {

**for** (Order o : orders) {

System.***out***.println(o);

}

}

* + TestSorting.java:

**package sorting;**

**public class TestSorting {**

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

**Order[] orders = {**

**new Order(301, "Arunachalam", 3500),**

**new Order(302, "Balasubramanian", 1500),**

**new Order(303, "Chidambaram", 9000),**

**new Order(304, "Dhanasekaran", 4500),**

**new Order(305, "Elangovan", 2500)**

**};**

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

**Sorter.printOrders(orders);**

**System.out.println("\n🔄 Sorted using Bubble Sort:");**

**Sorter.bubbleSort(orders);**

**Sorter.printOrders(orders);**

**orders = new Order[] {**

**new Order(301, "Arunachalam", 3500),**

**new Order(302, "Balasubramanian", 1500),**

**new Order(303, "Chidambaram", 9000),**

**new Order(304, "Dhanasekaran", 4500),**

**new Order(305, "Elangovan", 2500)**

**};**

**System.out.println("\n🔀 Sorted using Quick Sort:");**

**Sorter.quickSort(orders, 0, orders.length - 1);**

**Sorter.printOrders(orders);**

**}**

**}**

**Output:** A screenshot of a computer

AI-generated content may be incorrect.

1. **Employee Management System**

**Program:**

* StaffMember.java:

**package staffsystem;**

**public class StaffMember {**

**private int id;**

**private String name;**

**private String role;**

**private double salary;**

**public StaffMember(int id, String name, String role, double salary) {**

**this.id = id;**

**this.name = name;**

**this.role = role;**

**this.salary = salary;**

**}**

**public int getId() {**

**return id;**

**}**

**@Override**

**public String toString() {**

**return "[" + id + "] " + name + " - " + role + " - ₹" + salary;**

**}**

**}**

* StaffDirectory.java:

**package staffsystem;**

**public class StaffDirectory {**

**private StaffMember[] records;**

**private int count;**

**public StaffDirectory(int capacity) {**

**records = new StaffMember[capacity];**

**count = 0;**

**}**

**public void register(StaffMember staff) {**

**if (count < records.length) {**

**records[count++] = staff;**

**System.*out*.println("Registered: " + staff);**

**} else {**

**System.*out*.println("Directory is full. Cannot add more entries.");**

**}**

**}**

**public StaffMember locate(int id) {**

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

**if (records[i].getId() == id) {**

**return records[i];**

**}**

**}**

**return null;**

**}**

**public void listAll() {**

**System.*out*.println("Staff Listing:");**

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

**System.*out*.println(records[i]);**

**}**

**}**

**public void remove(int id) {**

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

**if (records[i].getId() == id) {**

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

**records[j] = records[j + 1];**

**}**

**records[count - 1] = null;**

**count--;**

**System.*out*.println("Removed entry for ID: " + id);**

**return;**

**}**

**}**

**System.*out*.println("ID not found.");**

**}**

**}**

* MainApplication.java:

**package staffsystem;**

**public class MainApplication {**

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

**StaffDirectory directory = new StaffDirectory(5);**

**directory.register(new StaffMember(1, "Ajay", "Supervisor", 65000));**

**directory.register(new StaffMember(2, "Kaarthick", "Technician", 55000));**

**directory.register(new StaffMember(3, "Rahul", "Coordinator", 50000));**

**System.*out*.println();**

**directory.listAll();**

**System.*out*.println("\nLooking up ID 2:");**

**StaffMember found = directory.locate(2);**

**System.*out*.println(found != null ? found : "Record not located");**

**System.*out*.println("\nRemoving ID 2:");**

**directory.remove(2);**

**System.*out*.println();**

**directory.listAll();**

**}**

**}**

**Output:**

**A screenshot of a computer

AI-generated content may be incorrect.**

1. **Task Management System**

**Program:**

* Job.java:

**package workflowmanager;**

**public class Job {**

**private int jobId;**

**private String title;**

**private String currentState;**

**public Job(int jobId, String title, String currentState) {**

**this.jobId = jobId;**

**this.title = title;**

**this.currentState = currentState;**

**}**

**public int getJobId() {**

**return jobId;**

**}**

**@Override**

**public String toString() {**

**return "[" + jobId + "] " + title + " - " + currentState;**

**}**

**}**

* JobNode.java:

**package workflowmanager;**

**public class JobNode {**

**Job job;**

**JobNode next;**

**public JobNode(Job job) {**

**this.job = job;**

**this.next = null;**

**}**

**}**

* WorkflowManager.java:

**package workflowmanager;**

**public class WorkflowManager {**

**private JobNode head;**

**public void addJob(Job job) {**

**JobNode newNode = new JobNode(job);**

**if (head == null) {**

**head = newNode;**

**} else {**

**JobNode temp = head;**

**while (temp.next != null) {**

**temp = temp.next;**

**}**

**temp.next = newNode;**

**}**

**System.*out*.println("Job added: " + job);**

**}**

**public void showAllJobs() {**

**if (head == null) {**

**System.*out*.println("No jobs to display.");**

**return;**

**}**

**System.*out*.println("Job List:");**

**JobNode current = head;**

**while (current != null) {**

**System.*out*.println(current.job);**

**current = current.next;**

**}**

**}**

**public Job findJobById(int jobId) {**

**JobNode current = head;**

**while (current != null) {**

**if (current.job.getJobId() == jobId) {**

**return current.job;**

**}**

**current = current.next;**

**}**

**return null;**

**public void removeJob(int jobId) {**

**if (head == null) {**

**System.*out*.println("No jobs available.");**

**return;**

**}**

**if (head.job.getJobId() == jobId) {**

**head = head.next;**

**System.*out*.println("Job with ID " + jobId + " removed.");**

**return;**

**}**

**JobNode prev = head;**

**JobNode current = head.next;**

**while (current != null) {**

**if (current.job.getJobId() == jobId) {**

**prev.next = current.next;**

**System.*out*.println("Job with ID " + jobId + " removed.");**

**return;**

**}**

**prev = current;**

**current = current.next;**

**}**

**System.*out*.println("Job ID " + jobId + " not found.");**

**}**

**}**

* Main.java:

**package workflowmanager;**

**public class Main {**

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

**WorkflowManager manager = new WorkflowManager();**

**manager.addJob(new Job(1, "Unit Testing", "Pending"));**

**manager.addJob(new Job(2, "Architecture Design", "In Progress"));**

**manager.addJob(new Job(3, "Deployment", "Pending"));**

**System.*out*.println();**

**manager.showAllJobs();**

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

**Job result = manager.findJobById(2);**

**System.*out*.println(result != null ? result : "Job not found.");**

**System.*out*.println("\nRemoving job ID 1:");**

**manager.removeJob(1);**

**System.*out*.println();**

**manager.showAllJobs();**

**}**

**}**

**Output:**

A screenshot of a computer

AI-generated content may be incorrect.

1. **Library Management System**

**Program:**

* Publication.java:

**package catalogue;**

**public class Publication {**

**private int id;**

**private String name;**

**private String writer;**

**public Publication(int id, String name, String writer) {**

**this.id = id;**

**this.name = name.toLowerCase();**

**this.writer = writer;**

**}**

**public String getName() {**

**return name;**

**}**

**public int getId() {**

**return id;**

**}**

**@Override**

**public String toString() {**

**return "[" + id + "] " + name + " by " + writer;**

**}**

**}**

* CatalogueManager.java:

**package catalogue;**

**import java.util.Arrays;**

**import java.util.Comparator;**

**public class CatalogueManager {**

**private Publication[] collection;**

**private int count;**

**public CatalogueManager(int capacity) {**

**collection = new Publication[capacity];**

**count = 0;**

**}**

**public void addPublication(Publication p) {**

**if (count < collection.length) {**

**collection[count++] = p;**

**} else {**

**System.*out*.println("Catalogue is full.");**

**}**

**}**

**public Publication searchLinearByTitle(String title) {**

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

**if (collection[i].getName().equalsIgnoreCase(title)) {**

**return collection[i];**

**}**

**}**

**return null;**

**}**

**public Publication searchBinaryByTitle(String title) {**

**Arrays.*sort*(collection, 0, count, Comparator.*comparing*(Publication::getName));**

**int left = 0, right = count - 1;**

**title = title.toLowerCase();**

**while (left <= right) {**

**int mid = (left + right) / 2;**

**int cmp = collection[mid].getName().compareTo(title);**

**if (cmp == 0) return collection[mid];**

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

**else right = mid - 1;**

**}**

**return null;**

**}**

**public void showAllPublications() {**

**System.*out*.println("Catalogue Contents:");**

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

**System.*out*.println(collection[i]);**

**}**

**}**

**}**

* Main.java:

**package catalogue;**

**public class Main {**

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

**CatalogueManager manager = new CatalogueManager(5);**

**manager.addPublication(new Publication(101, "Java Programming", "J. Vyas"));**

**manager.addPublication(new Publication(102, "Data Structures", "D. Sharma"));**

**manager.addPublication(new Publication(103, "Python Basics", "P. Yadav"));**

**manager.addPublication(new Publication(104, "Operating Systems", "O. Singh"));**

**manager.addPublication(new Publication(105, "Algorithms", "A. Mehta"));**

**manager.showAllPublications();**

**System.*out*.println("\nSearching (Linear) for 'Python Basics':");**

**Publication result1 = manager.searchLinearByTitle("Python Basics");**

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

**System.*out*.println("\nSearching (Binary) for 'Java Programming':");**

**Publication result2 = manager.searchBinaryByTitle("Java Programming");**

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

**}**

**}**

**Output:**

A screenshot of a computer

AI-generated content may be incorrect.

1. **Financial Forecasting**

**Program:**

* InvestmentProjection.java:

**package finance;**

**import java.util.Map;**

**import java.util.HashMap;**

**public class InvestmentProjection {**

**private Map<Integer, Double> memoTable = new HashMap<>();**

**public double computeRecursive(double baseAmount, double annualRate, int termYears) {**

**if (termYears == 0) return baseAmount;**

**return computeRecursive(baseAmount, annualRate, termYears - 1) \* (1 + annualRate);**

**}**

**public double computeMemoized(double baseAmount, double annualRate, int termYears) {**

**if (termYears == 0) return baseAmount;**

**if (memoTable.containsKey(termYears)) return memoTable.get(termYears);**

**double projected = computeMemoized(baseAmount, annualRate, termYears - 1) \* (1 + annualRate);**

**memoTable.put(termYears, projected);**

**return projected;**

**}**

**public double computeIterative(double baseAmount, double annualRate, int termYears) {**

**for (int i = 1; i <= termYears; i++) {**

**baseAmount \*= (1 + annualRate);**

**}**

**return baseAmount;**

**}**

**}**

* RunProjection.java:

**package finance;**

**public class RunProjection {**

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

**InvestmentProjection planner = new InvestmentProjection();**

**double initialInvestment = 100000;**

**double yearlyIncrease = 0.08;**

**int duration = 15;**

**System.*out*.printf("Recursive Estimate: ₹%.2f%n",**

**planner.computeRecursive(initialInvestment, yearlyIncrease, duration));**

**System.*out*.printf("Memoized Estimate: ₹%.2f%n",**

**planner.computeMemoized(initialInvestment, yearlyIncrease, duration));**

**System.*out*.printf("Iterative Estimate: ₹%.2f%n",**

**planner.computeIterative(initialInvestment, yearlyIncrease, duration));**

**}**

**}**

**Output:**

A screenshot of a computer

AI-generated content may be incorrect.