**Algorithms\_Data Structures**

**Hands-on -Exercise 1: Inventory Management System**

**Code:**

Main.html-

import java.util.\*;

public class Main {

public static void main(String[] args) {

Inventory inventory = new Inventory();

inventory.addProduct(new Product(101, "Keyboard", 50, 799.99));

inventory.addProduct(new Product(102, "Mouse", 100, 499.50));

inventory.addProduct(new Product(103, "Monitor", 30, 7999.00));

inventory.updateProduct(102, 80, 475.00);

inventory.deleteProduct(101);

inventory.displayAllProducts();

}

}

class Product {

int productId;

String productName;

int quantity;

double price;

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

this.productId = productId;

this.productName = productName;

this.quantity = quantity;

this.price = price;

}

}

class Inventory {

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

public void addProduct(Product product) {

products.put(product.productId, product);

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

}

public void updateProduct(int productId, int quantity, double price) {

Product p = products.get(productId);

if (p != null) {

p.quantity = quantity;

p.price = price;

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

}

}

public void deleteProduct(int productId) {

Product removed = products.remove(productId);

if (removed != null) {

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

}

}

public void displayAllProducts() {

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

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

System.out.println(p.productId + " - " + p.productName + " - Qty: " + p.quantity + " - ₹" + p.price);

}

}

}

OUTPUT:

A screenshot of a computer

AI-generated content may be incorrect.

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

**Code:**

Main.html-

import java.util.\*;

public class Main {

public static void main(String[] args) {

Product[] products = {

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

new Product(102, "Shoes", "Footwear"),

new Product(103, "Phone", "Electronics"),

new Product(104, "Shirt", "Clothing"),

new Product(105, "Watch", "Accessories")

};

System.out.println("Linear Search:");

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

System.out.println(result1 != null ? "Found: " + result1.productName : "Not found");

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

System.out.println("Binary Search:");

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

System.out.println(result2 != null ? "Found: " + result2.productName : "Not found");

}

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

for (Product p : products) {

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

return p;

}

}

return null;

}

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

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

while (left <= right) {

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

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

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

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

else right = mid - 1;

}

return null;

}

}

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;

}

}

OUTPUT:

A screenshot of a computer

AI-generated content may be incorrect.

**Hands-on - Exercise 3: Sorting Customer Orders**

**Code:**

Main.html-

public class Main {

public static void main(String[] args) {

Order[] orders1 = {

new Order(101, "Alice", 1200.50),

new Order(102, "Bob", 250.75),

new Order(103, "Charlie", 999.99),

new Order(104, "Diana", 3000.00),

new Order(105, "Eve", 750.25)

};

Order[] orders2 = orders1.clone();

System.out.println("Bubble Sort:");

bubbleSort(orders1);

printOrders(orders1);

System.out.println("Quick Sort:");

quickSort(orders2, 0, orders2.length - 1);

printOrders(orders2);

}

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) {

Order pivot = arr[high];

int i = low - 1;

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

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

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;

}

static void printOrders(Order[] arr) {

for (Order o : arr) {

System.out.println(o.orderId + " - " + o.customerName + " - ₹" + o.totalPrice);

}

}

static 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;

}

}

OUTPUT:

A screenshot of a computer

AI-generated content may be incorrect.

**Hands-on - Exercise 4: Employee Management System**

**Code:**

Main.html-

public class Main {

public static void main(String[] args) {

EmployeeManager manager = new EmployeeManager(5);

manager.addEmployee(new Employee(1, "Alice", "Manager", 60000));

manager.addEmployee(new Employee(2, "Bob", "Developer", 45000));

manager.addEmployee(new Employee(3, "Charlie", "Tester", 40000));

manager.traverseEmployees();

System.out.println("Searching for Employee ID 2:");

Employee emp = manager.searchEmployee(2);

if (emp != null) System.out.println(emp.name + " - " + emp.position);

else System.out.println("Not found");

manager.deleteEmployee(2);

manager.traverseEmployees();

}

}

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;

}

}

class EmployeeManager {

private Employee[] employees;

private int count = 0;

public EmployeeManager(int capacity) {

employees = new Employee[capacity];

}

public void addEmployee(Employee e) {

if (count < employees.length) {

employees[count++] = e;

System.out.println("Added: " + e.name);

}

}

public Employee searchEmployee(int id) {

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

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

}

return null;

}

public void deleteEmployee(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];

}

employees[--count] = null;

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

return;

}

}

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

}

public void traverseEmployees() {

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

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

System.out.println(employees[i].employeeId + " - " + employees[i].name + " - " + employees[i].position + " - ₹" + employees[i].salary);

}

}

}

OUTPUT:

A screenshot of a computer

AI-generated content may be incorrect.

**Hands-on - Exercise 5: Task Management System**

**Code:**

Main.html-

public class Main {

public static void main(String[] args) {

TaskList taskList = new TaskList();

taskList.addTask(new Task(1, "Design module", "Pending"));

taskList.addTask(new Task(2, "Write code", "In Progress"));

taskList.addTask(new Task(3, "Test module", "Pending"));

taskList.traverseTasks();

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

Task found = taskList.searchTask(2);

if (found != null) System.out.println(found.taskName + " - " + found.status);

else System.out.println("Task not found");

taskList.deleteTask(2);

taskList.traverseTasks();

}

}

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;

}

}

class TaskNode {

Task task;

TaskNode next;

public TaskNode(Task task) {

this.task = task;

this.next = null;

}

}

class TaskList {

private TaskNode head;

public void addTask(Task task) {

TaskNode newNode = new TaskNode(task);

if (head == null) head = newNode;

else {

TaskNode temp = head;

while (temp.next != null) temp = temp.next;

temp.next = newNode;

}

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

}

public void traverseTasks() {

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

TaskNode temp = head;

while (temp != null) {

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

temp = temp.next;

}

}

public Task searchTask(int id) {

TaskNode temp = head;

while (temp != null) {

if (temp.task.taskId == id) return temp.task;

temp = temp.next;

}

return null;

}

public void deleteTask(int id) {

if (head == null) return;

if (head.task.taskId == id) {

head = head.next;

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

return;

}

TaskNode prev = head;

TaskNode curr = head.next;

while (curr != null) {

if (curr.task.taskId == id) {

prev.next = curr.next;

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

return;

}

prev = curr;

curr = curr.next;

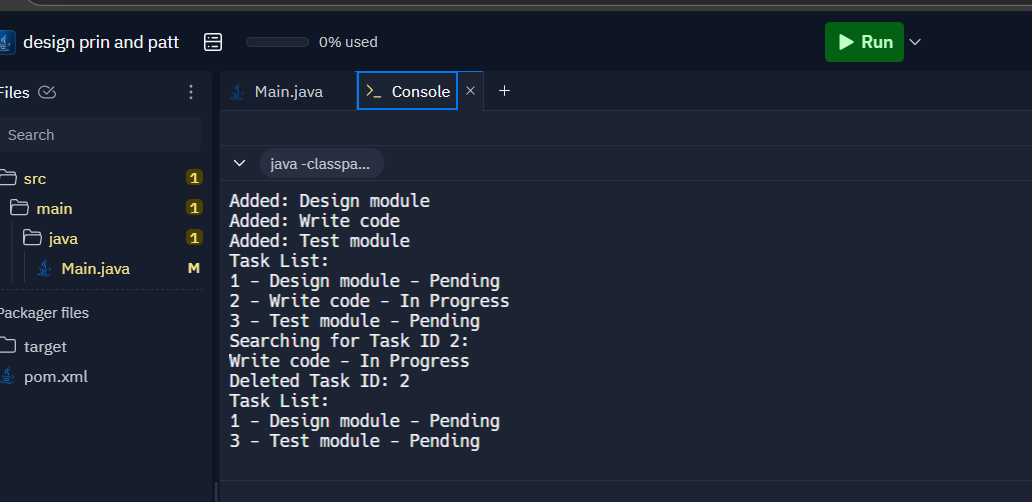
}

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

}

}

OUTPUT:



**Hands-on - Exercise 6: Library Management System**

**Code:**

Main.html-

import java.util.\*;

public class Main {

public static void main(String[] args) {

Book[] books = {

new Book(101, "Data Structures", "Mark Allen"),

new Book(102, "Java Programming", "James Gosling"),

new Book(103, "Algorithms", "Robert Sedgewick"),

new Book(104, "Computer Networks", "Andrew Tanenbaum")

};

System.out.println("Linear Search:");

Book b1 = linearSearch(books, "Java Programming");

System.out.println(b1 != null ? "Found: " + b1.title : "Not Found");

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

System.out.println("Binary Search:");

Book b2 = binarySearch(books, "Java Programming");

System.out.println(b2 != null ? "Found: " + b2.title : "Not Found");

}

static Book linearSearch(Book[] books, String title) {

for (Book b : books) {

if (b.title.equalsIgnoreCase(title)) return b;

}

return null;

}

static Book binarySearch(Book[] books, String title) {

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

while (left <= right) {

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

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

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

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

else right = mid - 1;

}

return null;

}

static 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;

}

}

}

OUTPUT:

A screenshot of a computer

AI-generated content may be incorrect.

**Hands-on - Exercise 7: Financial Forecasting**

**Code:**

Main.html-

public class Main {

public static void main(String[] args) {

double initialAmount = 10000;

double growthRate = 0.08;

int years = 5;

double future = calculateFutureValue(initialAmount, growthRate, years);

System.out.println("Future Value after " + years + " years: ₹" + future);

}

public static double calculateFutureValue(double amount, double rate, int years) {

if (years == 0) return amount;

return calculateFutureValue(amount \* (1 + rate), rate, years - 1);

}

}

OUTPUT:

A screenshot of a computer

AI-generated content may be incorrect.