**Data Structures And Algorithms**

**Exercise 1: Inventory Management System**

import java.util.HashMap;

import java.util.Map;

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;

}

@Override

public String toString() {

return "[ID: " + productId + ", Name: " + productName + ", Qty: " + quantity + ", Price: ₹" + price + "]";

}

}

class Inventory {

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

public void addProduct(Product p) {

inventory.put(p.productId, p);

System.out.println("✅ Added: " + p);

}

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

if (inventory.containsKey(productId)) {

Product p = inventory.get(productId);

p.quantity = quantity;

p.price = price;

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

} else {

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

}

}

public void deleteProduct(int productId) {

if (inventory.remove(productId) != null) {

System.out.println("🗑️ Deleted product ID: " + productId);

} else {

System.out.println("❌ Product not found for deletion");

}

}

public void showAllProducts() {

System.out.println("\n📦 Inventory Snapshot:");

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

System.out.println(p);

}

}

}

public class InventoryDemo {

public static void main(String[] args) {

Inventory warehouse = new Inventory();

Product p1 = new Product(101, "Wireless Mouse", 50, 899.99);

Product p2 = new Product(102, "Mechanical Keyboard", 30, 1999.50);

Product p3 = new Product(103, "HD Monitor", 20, 7499.00);

warehouse.addProduct(p1);

warehouse.addProduct(p2);

warehouse.addProduct(p3);

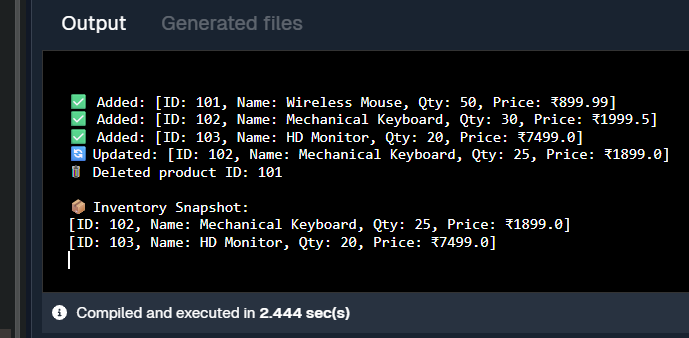
warehouse.updateProduct(102, 25, 1899.00); // Update quantity and price

warehouse.deleteProduct(101); // Delete a product

warehouse.showAllProducts(); // View current inventory

}

}



Exercise 2: E-commerce Platform Search Function

//Code

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

public class ProductSearch {

private static List<String> getProductList() {

List<String> products = new ArrayList<>();

products.add("iPhone 15 Pro");

products.add("Samsung Galaxy S24");

products.add("MacBook Air M2");

products.add("Dell XPS 13");

products.add("Sony WH-1000XM5 Headphones");

products.add("Apple Watch Series 9");

products.add("HP Pavilion Laptop");

return products;

}

private static List<String> searchProducts(String keyword, List<String> products) {

List<String> results = new ArrayList<>();

for (String product : products) {

if (product.toLowerCase().contains(keyword.toLowerCase())) {

results.add(product);

}

}

return results;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

List<String> productList = getProductList();

System.out.println("🛒 Welcome to the E-commerce Search Platform");

System.out.print("🔍 Enter a keyword to search for products: ");

String keyword = scanner.nextLine();

List<String> searchResults = searchProducts(keyword, productList);

System.out.println("\n🔎 Search Results:");

if (searchResults.isEmpty()) {

System.out.println("❌ No products found matching \"" + keyword + "\".");

} else {

for (String product : searchResults) {

System.out.println("✅ " + product);

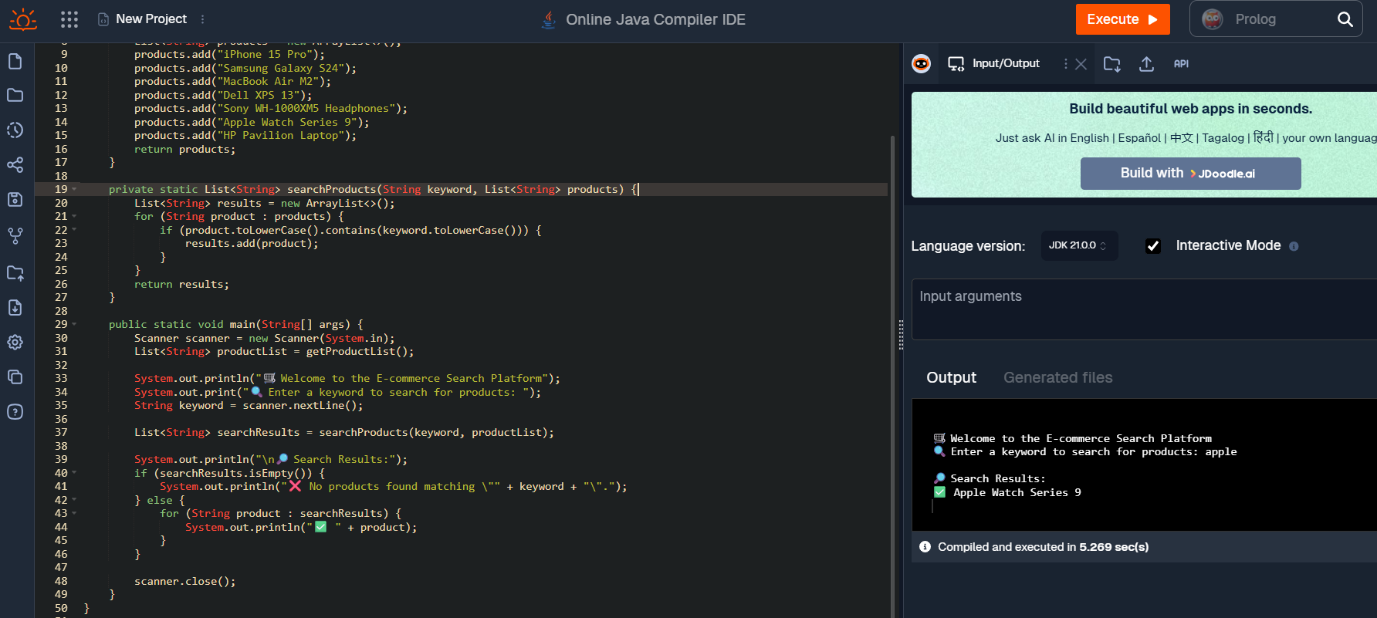
}

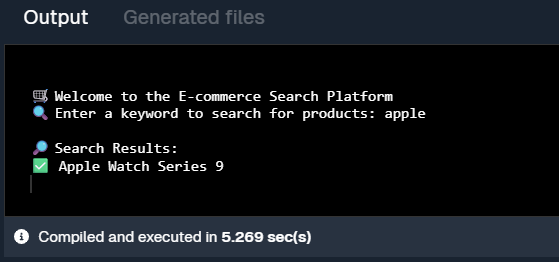
}

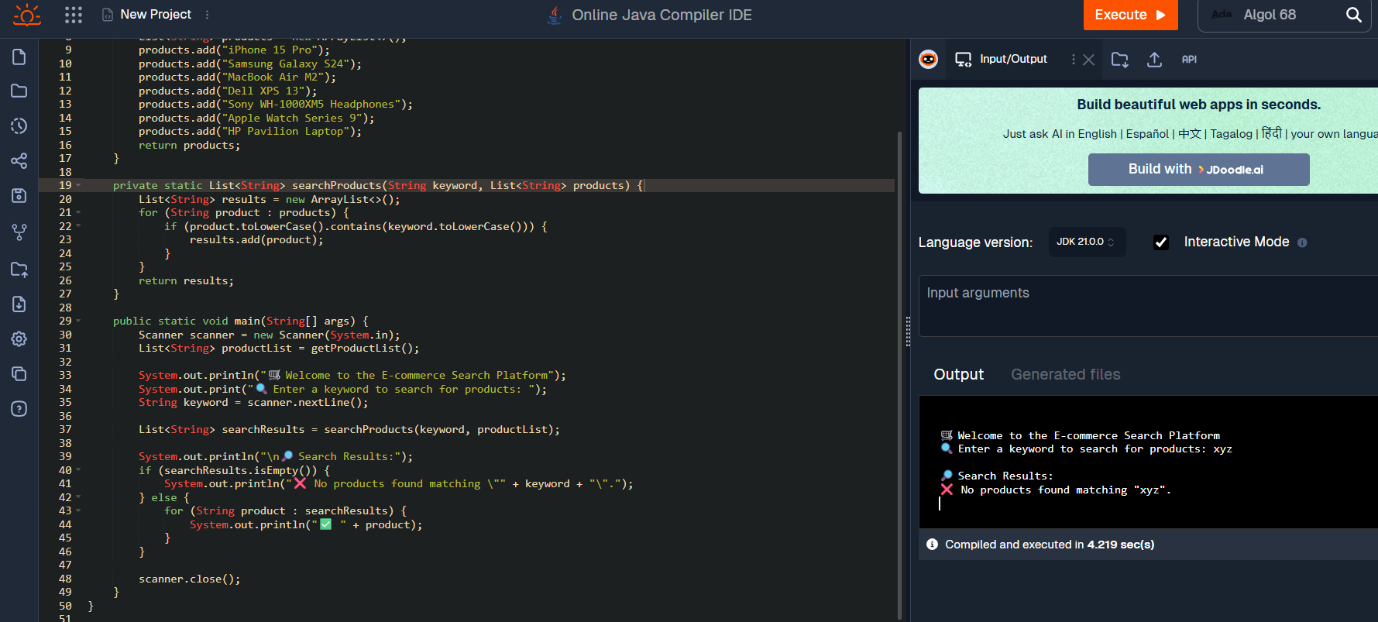
scanner.close();

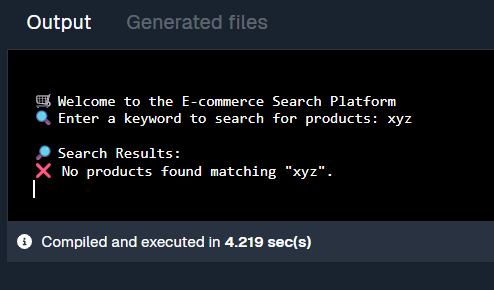
}

}









Exercise 3: Sorting Customer Orders

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;

}

@Override

public String toString() {

return "[OrderID: " + orderId + ", Name: " + customerName + ", ₹" + totalPrice + "]";

}

}

public class CustomerOrderSorter {

public static void bubbleSort(Order[] orders) {

int n = orders.length;

for (int i = 0; i < n-1; i++) {

for (int j = 0; j < n-i-1; j++) {

if (orders[j].totalPrice > orders[j+1].totalPrice) {

Order temp = orders[j];

orders[j] = orders[j+1];

orders[j+1] = temp;

}

}

}

}

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].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;

}

public static void displayOrders(String title, Order[] orders) {

System.out.println("\n🔽 " + title);

for (Order o : orders) {

System.out.println(o);

}

}

public static void main(String[] args) {

Order[] orders1 = {

new Order(101, "Riya", 899.99),

new Order(102, "Arjun", 1200.50),

new Order(103, "Zoya", 499.00),

new Order(104, "Karan", 3000.75),

new Order(105, "Meera", 1500.10)

};

Order[] orders2 = orders1.clone();

bubbleSort(orders1);

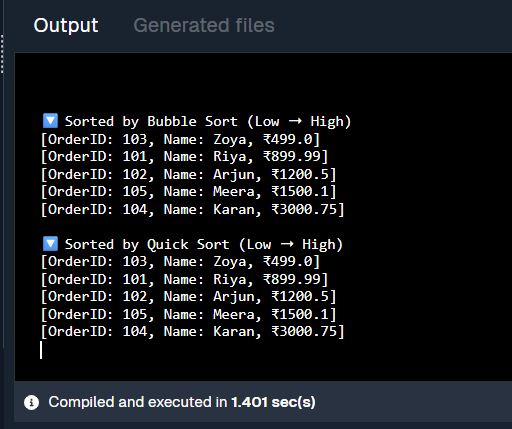
displayOrders("Sorted by Bubble Sort (Low ➝ High)", orders1);

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

displayOrders("Sorted by Quick Sort (Low ➝ High)", orders2);

}

}



Exercise 4: Employee Management System

import java.util.Scanner;

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;

}

@Override

public String toString() {

return "[ID: " + employeeId + ", Name: " + name + ", Position: " + position + ", Salary: ₹" + salary + "]";

}

}

public class EmployeeManager {

private static final int MAX = 100;

private Employee[] employees = new Employee[MAX];

private int size = 0;

public void addEmployee(Employee emp) {

if (size < MAX) {

employees[size++] = emp;

System.out.println("✅ Employee added: " + emp.name);

} else {

System.out.println("❌ Employee list is full!");

}

}

public Employee searchById(int id) {

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

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

return employees[i];

}

}

return null;

}

public void showAllEmployees() {

System.out.println("\n👩‍💼 All Employees:");

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

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

}

}

public boolean deleteById(int id) {

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

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

// Shift elements

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

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

}

employees[--size] = null;

return true;

}

}

return false;

}

public static void main(String[] args) {

EmployeeManager manager = new EmployeeManager();

manager.addEmployee(new Employee(1, "Riya", "Developer", 60000));

manager.addEmployee(new Employee(2, "Amit", "Designer", 55000));

manager.addEmployee(new Employee(3, "Zara", "Manager", 75000));

manager.showAllEmployees();

System.out.println("\n🔍 Searching for employee with ID 2...");

Employee emp = manager.searchById(2);

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

System.out.println("\n🗑️ Deleting employee with ID 1...");

boolean deleted = manager.deleteById(1);

System.out.println(deleted ? "Deleted successfully!" : "Employee not found.");

manager.showAllEmployees();

}

}



Exercise 5: Task Management System

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;

}

@Override

public String toString() {

return "[Task ID: " + taskId + ", Name: " + taskName + ", Status: " + status + "]";

}

}

class TaskNode {

Task data;

TaskNode next;

public TaskNode(Task data) {

this.data = data;

this.next = null;

}

}

public class TaskManager {

TaskNode head = null;

public void addTask(Task task) {

TaskNode newNode = new TaskNode(task);

if (head == null) {

head = newNode;

} else {

TaskNode current = head;

while (current.next != null) {

current = current.next;

}

current.next = newNode;

}

System.out.println("✅ Task added: " + task.taskName);

}

public Task searchTask(int taskId) {

TaskNode current = head;

while (current != null) {

if (current.data.taskId == taskId) {

return current.data;

}

current = current.next;

}

return null;

}

public void showAllTasks() {

System.out.println("\n📋 Task List:");

TaskNode current = head;

while (current != null) {

System.out.println(current.data);

current = current.next;

}

}

public boolean deleteTask(int taskId) {

if (head == null) return false;

if (head.data.taskId == taskId) {

head = head.next;

return true;

}

TaskNode current = head;

while (current.next != null) {

if (current.next.data.taskId == taskId) {

current.next = current.next.next;

return true;

}

current = current.next;

}

return false;

}

public static void main(String[] args) {

TaskManager manager = new TaskManager();

manager.addTask(new Task(1, "Design UI", "Pending"));

manager.addTask(new Task(2, "Build Backend", "In Progress"));

manager.addTask(new Task(3, "Write Tests", "Pending"));

manager.showAllTasks();

System.out.println("\n🔍 Searching for task ID 2...");

Task result = manager.searchTask(2);

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

System.out.println("\n🗑️ Deleting task ID 1...");

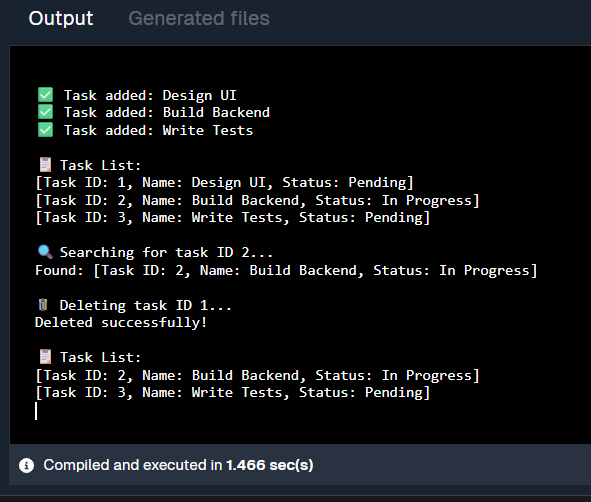
boolean deleted = manager.deleteTask(1);

System.out.println(deleted ? "Deleted successfully!" : "Task not found.");

manager.showAllTasks();

}

}



Exercise 6: Library Management System

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.toLowerCase(); // normalize for consistent search

this.author = author;

}

@Override

public String toString() {

return "[Book ID: " + bookId + ", Title: " + title + ", Author: " + author + "]";

}

}

public class LibrarySearchSystem {

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

for (Book book : books) {

if (book.title.equalsIgnoreCase(title)) {

return book;

}

}

return null;

}

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

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

title = title.toLowerCase();

while (low <= high) {

int mid = (low + high) / 2;

int comparison = books[mid].title.compareTo(title);

if (comparison == 0) {

return books[mid];

} else if (comparison < 0) {

low = mid + 1;

} else {

high = mid - 1;

}

}

return null;

}

public static void main(String[] args) {

Book[] books = {

new Book(101, "Java Programming", "Ramesh Kumar"),

new Book(102, "Data Structures", "Anita Mehta"),

new Book(103, "Operating Systems", "Suresh Iyer"),

new Book(104, "Computer Networks", "Preeti Singh"),

new Book(105, "Web Development", "Sanjay Verma")

};

System.out.println("\n🔎 Linear Search for 'Operating Systems':");

Book found = linearSearch(books, "Operating Systems");

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

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

System.out.println("\n🔎 Binary Search for 'Web Development':");

Book binaryResult = binarySearch(books, "Web Development");

System.out.println(binaryResult != null ? "✅ Found: " + binaryResult : "❌ Not Found");

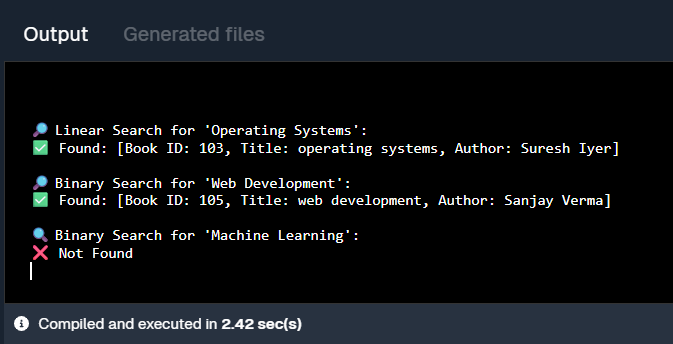
System.out.println("\n🔍 Binary Search for 'Machine Learning':");

Book fail = binarySearch(books, "Machine Learning");

System.out.println(fail != null ? "✅ Found: " + fail : "❌ Not Found");

}

}



Exercise 7: Financial Forecasting

//Code

import java.util.Scanner;

public class FinancialForecast {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("📊 Financial Forecast Tool");

System.out.print("💼 Enter Initial Investment Amount: ₹");

double initialInvestment = sc.nextDouble();

System.out.print("💰 Enter Monthly Income: ₹");

double monthlyIncome = sc.nextDouble();

System.out.print("💸 Enter Monthly Expenses: ₹");

double monthlyExpenses = sc.nextDouble();

System.out.print("📆 Enter Number of Months to Forecast: ");

int months = sc.nextInt();

System.out.println("\n📈 Forecast Report:");

double currentBalance = initialInvestment;

for (int i = 1; i <= months; i++) {

currentBalance += (monthlyIncome - monthlyExpenses);

System.out.printf("Month %02d: ₹%.2f\n", i, currentBalance);

}

System.out.printf("\n📌 Final Forecasted Balance after %d months: ₹%.2f\n", months, currentBalance);

sc.close();

}

}

