**Algorithms\_Data\_Structure Hands ON Exercises**

**Exercise 1: Inventory Management System**

**Scenario:**

You are developing an inventory management system for a warehouse. Efficient data storage and retrieval are crucial.

**CODE:**

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;

}

}

class InventorySystem {

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

void addProduct(Product p) {

inventory.put(p.productId, p);

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

}

void updateProduct(int id, int quantity, double price) {

Product p = inventory.get(id);

if (p != null) {

p.quantity = quantity;

p.price = price;

System.out.println("Updated product ID " + id);

} else {

System.out.println("Product not found: ID " + id);

}

}

void deleteProduct(int id) {

if (inventory.containsKey(id)) {

inventory.remove(id);

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

} else {

System.out.println("Product not found: ID " + id);

}

}

}

public class Main {

public static void main(String[] args) {

InventorySystem inventory = new InventorySystem();

inventory.addProduct(new Product(1, "Pen", 100, 5.0));

inventory.addProduct(new Product(2, "Notebook", 50, 20.0));

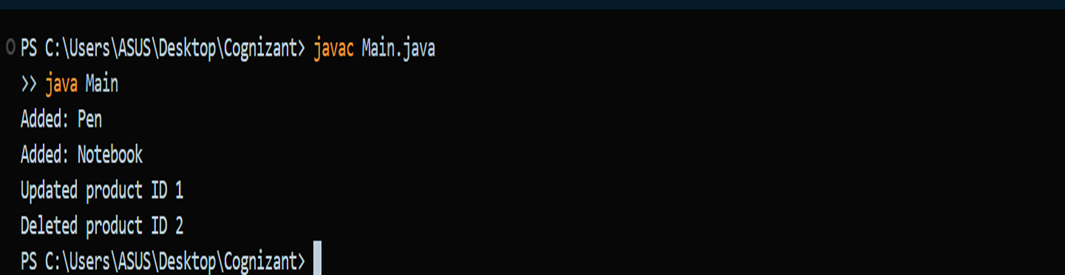
inventory.updateProduct(1, 120,4.5);

inventory.deleteProduct(2);

}

**}**

**OUTPUT:**

****

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

**Scenario:**

You are working on the search functionality of an e-commerce platform. The search needs to be optimized for fast performance.

**OUTPUT:**

import java.util.Arrays;

import java.util.Comparator;

class ProductSearch {

int id;

String productName;

String category;

ProductSearch(int id, String productName, String category) {

this.id = id;

this.productName = productName;

this.category = category;

}

static ProductSearch binarySearch(ProductSearch[] arr, String target) {

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

while (left <= right) {

int mid = (left + right) / 2;

int cmp = arr[mid].productName.compareToIgnoreCase(target);

if (cmp == 0)

return arr[mid];

else if (cmp < 0)

left = mid + 1;

else

right = mid - 1;

}

return null;

}

}

public class Main {

public static void main(String[] args) {

ProductSearch[] products = {

new ProductSearch(1, "Phone", "Electronics"),

new ProductSearch(2, "Laptop", "Electronics"),

new ProductSearch(3, "Tablet", "Electronics")

};

// Sort the array before binary search

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

ProductSearch found = ProductSearch.binarySearch(products, "Laptop");

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

}

}

**OUTPUT**: A screenshot of a computer program

AI-generated content may be incorrect.

**Exercise 3: Sorting Customer Orders**

**Scenario:**

You are tasked with sorting customer orders by their total price on an e-commerce platform. This helps in prioritizing high-value orders.

**CODE:**

class Order {

int orderId;

String customerName;

double totalPrice;

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

this.orderId = orderId;

this.customerName = customerName;

this.totalPrice = totalPrice;

}

}

class OrderSorting {

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);

}

}

private 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;

}

}

public class Main {

public static void main(String[] args) {

Order[] orders = {

new Order(101, "Alice", 500),

new Order(102, "Bob", 300),

new Order(103, "Charlie", 700)

};

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

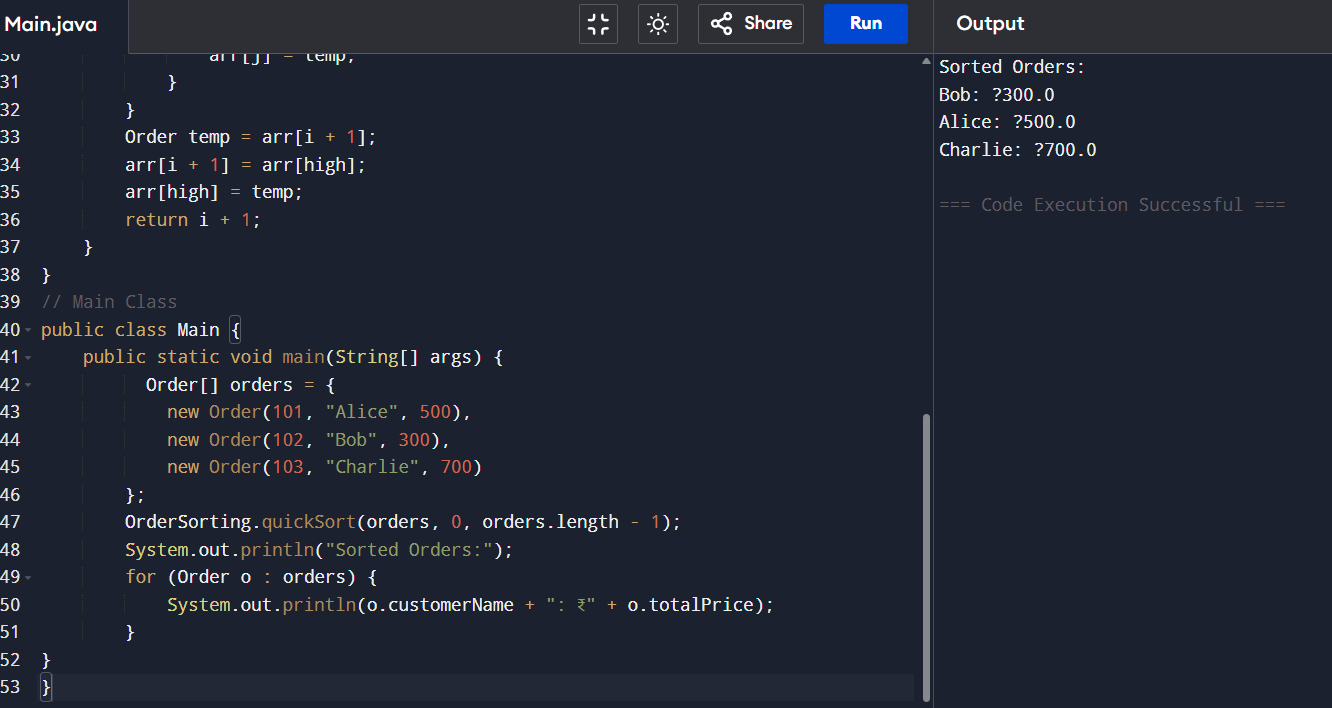
System.out.println("Sorted Orders:");

for (Order o : orders) {

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

}

**OUTPUT:**

****

**Exercise 4: Employee Management System**

**Scenario:**

**You are developing an employee management system for a company. Efficiently managing employee records is crucial.**

**CODE:**

import java.util.\*;

// Employee class

class Employee {

int id;

String name;

String designation;

double salary;

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

this.id = id;

this.name = name;

this.designation = designation;

this.salary = salary;

}

}

// EmployeeSystem class

class EmployeeSystem {

List<Employee> employees = new ArrayList<>();

void addEmployee(Employee e) {

employees.add(e);

}

void traverseEmployees() {

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

for (Employee e : employees) {

System.out.println(e.id + " " + e.name + " - " + e.designation + " : ₹" + e.salary);

}

}

}

public class Main {

public static void main(String[] args) {

// Exercise 4: Employee Management

EmployeeSystem empSys = new EmployeeSystem();

empSys.addEmployee(new Employee(1, "John", "Manager", 50000));

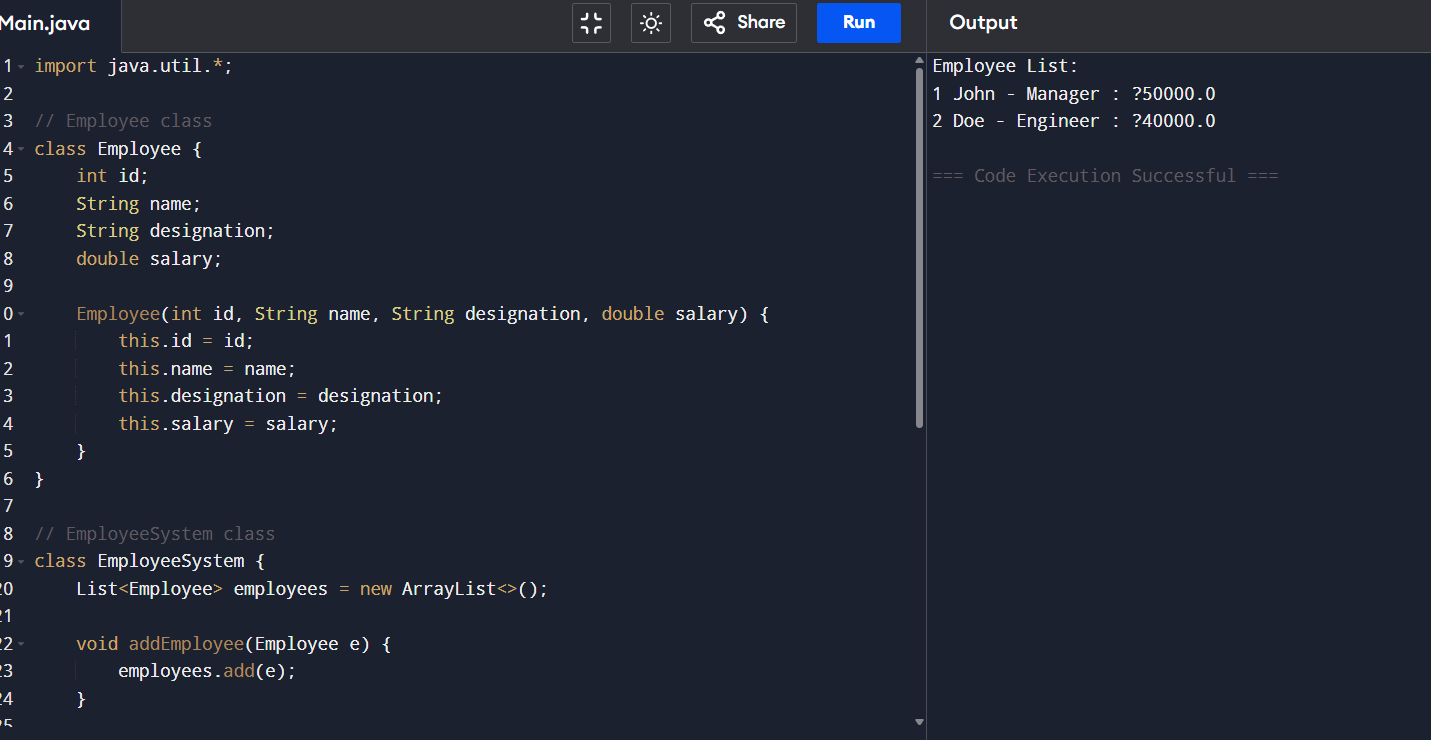
empSys.addEmployee(new Employee(2, "Doe", "Engineer", 40000));

empSys.traverseEmployees();

}

}

**OUTPUT:**

****

**Exercise 5: Task Management System**

**Scenario:**

You are developing a task management system where tasks need to be added, deleted, and traversed efficiently.

**CODE:**

import java.util.\*;

// Exercise 5: Task Management

class Task {

int id;

String title;

String status;

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

this.id = id;

this.title = title;

this.status = status;

}

}

class TaskManager {

List<Task> tasks = new ArrayList<>();

void addTask(Task t) {

tasks.add(t);

}

void traverseTasks() {

System.out.println("Tasks:");

for (Task t : tasks) {

System.out.println(t.id + ". " + t.title + " - " + t.status);

}

}

}

public class Main {

public static void main(String[] args) {

TaskManager taskMgr = new TaskManager();

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

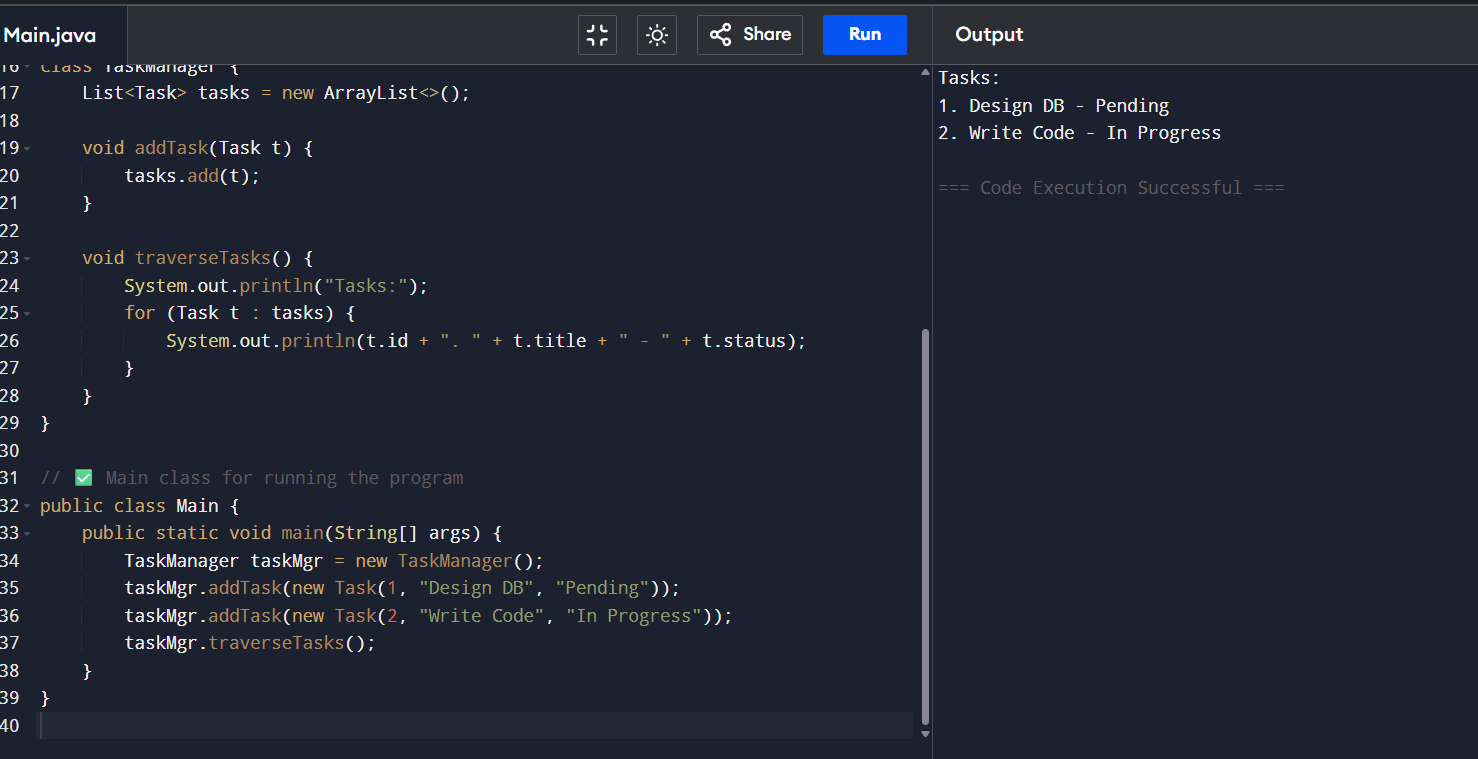
taskMgr.addTask(new Task(2, "Write Code", "In Progress"));

taskMgr.traverseTasks();

}

}

**OUTPUT:**

****

**Exercise 6: Library Management System**

**Scenario:**

You are developing a library management system where users can search for books by title or author.

**CODE:**

import java.util.\*;

class Book {

int id;

String title;

String author;

Book(int id, String title, String author) {

this.id = id;

this.title = title;

this.author = author;

}

}

class LibrarySearch {

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

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

while (left <= right) {

int mid = (left + right) / 2;

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

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

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

else right = mid - 1;

}

return null;

}

}

public class Main {

public static void main(String[] args) {

Book[] books = {

new Book(1, "Java Basics", "Author A"),

new Book(2, "Advanced Java", "Author B")

};

// Sort the array before binary search

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

// Perform binary search

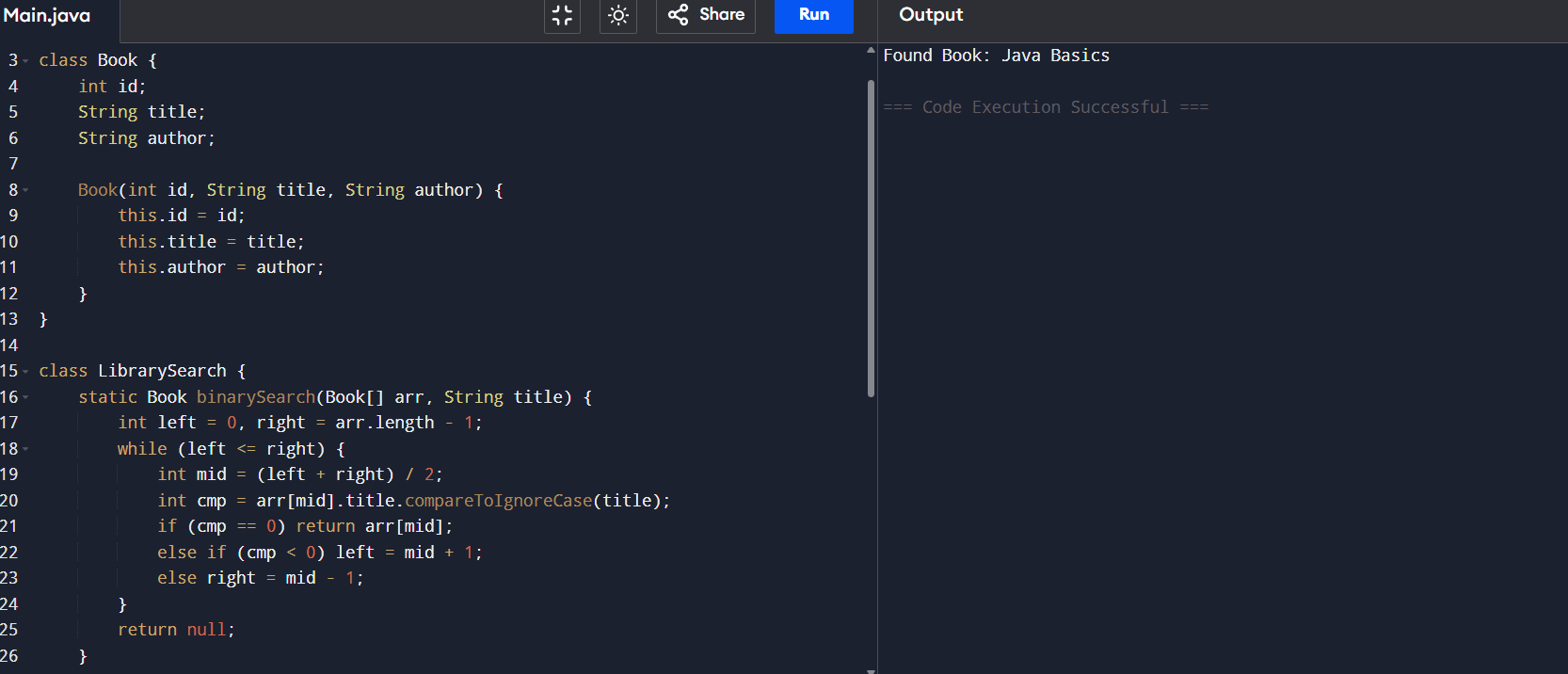
Book book = LibrarySearch.binarySearch(books, "Java Basics");

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

}

}

**OUTPUT:**

****

**Exercise 7: Financial Forecasting**

Scenario:

You are developing a financial forecasting tool that predicts future values based on past data.

**CODE:**

class Forecasting {

static double predictValueDP(double principal, double rate, int years) {

double[] dp = new double[years + 1];

dp[0] = principal;

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

dp[i] = dp[i - 1] \* (1 + rate);

}

return dp[years];

}

}

public class Main {

public static void main(String[] args) {

double future = Forecasting.predictValueDP(1000, 0.1, 5);

System.out.println("Future Value (5 years): ₹" + future);

}

}

