**Exercise 1: Inventory Management System**

**Code:**

import java.util.HashMap;

class Product {

String productId;

String productName;

int quantity;

double price;

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

this.productId = productId;

this.productName = productName;

this.quantity = quantity;

this.price = price;

}

@Override

public String toString() {

return productId + " - " + productName + " | Qty: " + quantity + " | Price: " + price;

}

}

public class InventorySystem {

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

public void addProduct(Product p) {

inventory.put(p.productId, p);

}

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

Product p = inventory.get(id);

if (p != null) {

p.quantity = quantity;

p.price = price;

}

}

public void deleteProduct(String id) {

inventory.remove(id);

}

public void displayInventory() {

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

System.out.println(p);

}

}

public static void main(String[] args) {

InventorySystem system = new InventorySystem();

system.addProduct(new Product("P001", "Laptop", 10, 75000));

system.addProduct(new Product("P002", "Mouse", 50, 500));

system.displayInventory();

system.updateProduct("P001", 8, 72000);

system.deleteProduct("P002");

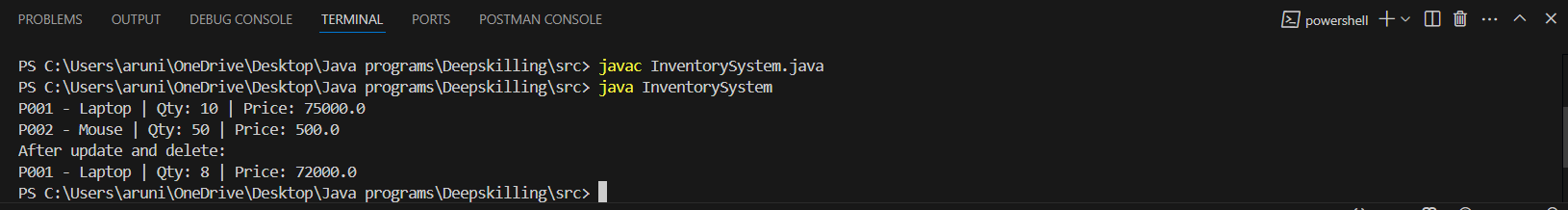
System.out.println("After update and delete:");

system.displayInventory();

}

}

**Output:**



**Exercise 2: Search Function in E-Commerce**

**Code:**

**Product.java**

import SearchFunction;

public class Product {

String productId;

String productName;

String category;

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

this.productId = productId;

this.productName = productName;

this.category = category;

}

@Override

public String toString() {

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

}

}

**SearchDemo.java**

import SearchFunction;

import java.util.Arrays;

import java.util.Comparator;

public class SearchDemo {

// Linear Search

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

for (Product p : products) {

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

return p;

}

}

return null;

}

// Binary Search

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

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

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;

}

// Main Method

public static void main(String[] args) {

Product[] products = {

new Product("P001", "Laptop", "Electronics"),

new Product("P002", "Keyboard", "Accessories"),

new Product("P003", "Mouse", "Accessories"),

new Product("P004", "Monitor", "Electronics")

};

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

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

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

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

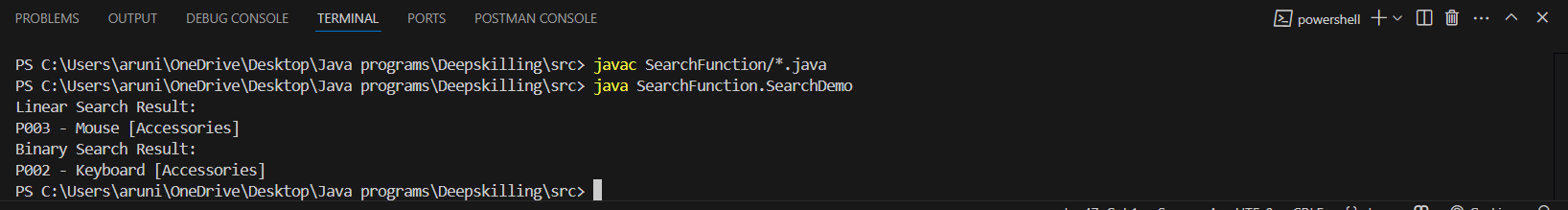
Product result2 = binarySearch(products, "Keyboard");

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

}

}

**Output:**

****

**Exercise 3: Sorting Customer Orders**

**Code:**

**Order.java**

package SortingOrder;

public class Order {

String orderId;

String customerName;

double totalPrice;

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

this.orderId = orderId;

this.customerName = customerName;

this.totalPrice = totalPrice;

}

@Override

public String toString() {

return orderId + " | " + customerName + " | (Rupees)" + totalPrice;

}

}

**SortDemo.java**

package SortingOrder;

public class SortDemo {

// Bubble Sort

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;

}

}

}

}

// Quick Sort

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;

}

// Utility: Print orders

public static void printOrders(Order[] orders) {

for (Order order : orders) {

System.out.println(order);

}

}

public static void main(String[] args) {

Order[] orders = {

new Order("O001", "Aru", 4500.00),

new Order("O002", "Riya", 2300.00),

new Order("O003", "Jay", 7800.00),

new Order("O004", "Tina", 1200.00)

};

System.out.println("Bubble Sort by Total Price:");

bubbleSort(orders);

printOrders(orders);

System.out.println("\n Quick Sort by Total Price:");

// re-initializing array since bubble sort changed the order

orders = new Order[]{

new Order("O001", "Aru", 4500.00),

new Order("O002", "Riya", 2300.00),

new Order("O003", "Jay", 7800.00),

new Order("O004", "Tina", 1200.00)

};

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

printOrders(orders);

}

}

**Output:**

****

**Exercise 4: Employee Management System**

**Code:**

**Employee.java**

package EmployeeSystem;

public class Employee {

String employeeId;

String name;

String position;

double salary;

public Employee(String employeeId, String name, String position, double salary) {

this.employeeId = employeeId;

this.name = name;

this.position = position;

this.salary = salary;

}

@Override

public String toString() {

return employeeId + " | " + name + " | " + position + " | (Rupees)" + salary;

}

}

**EmployeManager.java**

import java.util.Scanner;

public class EmployeeManager {

private static final int MAX\_EMPLOYEES = 100;

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

private int size = 0;

public void addEmployee(Employee e) {

if (size < MAX\_EMPLOYEES) {

employees[size++] = e;

} else {

System.out.println("Employee limit reached!");

}

}

public void displayEmployees() {

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

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

}

}

public Employee searchById(String id) {

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

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

return employees[i];

}

}

return null;

}

public void deleteById(String id) {

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

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

// Shift left

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

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

}

employees[size - 1] = null;

size--;

System.out.println("Employee deleted.");

return;

}

}

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

}

public static void main(String[] args) {

EmployeeManager manager = new EmployeeManager();

manager.addEmployee(new Employee("E001", "Aru", "Software Engineer", 70000));

manager.addEmployee(new Employee("E002", "Ravi", "QA Analyst", 55000));

manager.addEmployee(new Employee("E003", "Tina", "HR", 48000));

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

manager.displayEmployees();

System.out.println("\n Searching for E002:");

Employee e = manager.searchById("E002");

System.out.println(e != null ? e : "Not found");

System.out.println("\n Deleting E001:");

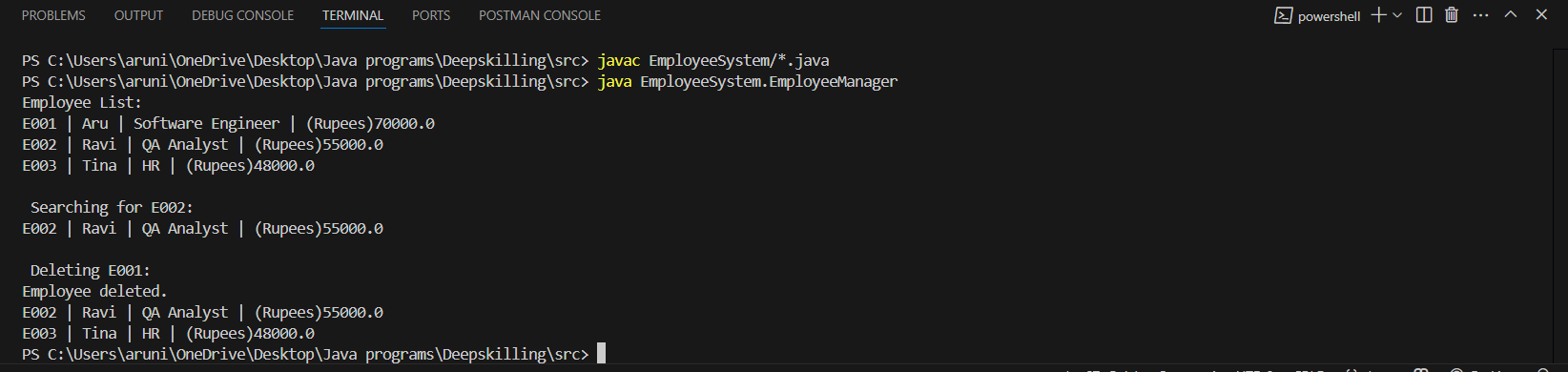
manager.deleteById("E001");

manager.displayEmployees();

}

}

**Output:**

****

**Exercise 5: Task Management System**

**Code:**

**Task.java**

package TaskSystem;

public class Task {

String taskId;

String taskName;

String status;

Task next; // Link to next task

public Task(String taskId, String taskName, String status) {

this.taskId = taskId;

this.taskName = taskName;

this.status = status;

this.next = null;

}

@Override

public String toString() {

return taskId + " | " + taskName + " | Status: " + status;

}

}

**TaskManager.java**

package TaskSystem;

public class TaskManager {

private Task head = null;

// Add task at the end

public void addTask(String taskId, String taskName, String status) {

Task newTask = new Task(taskId, taskName, status);

if (head == null) {

head = newTask;

} else {

Task current = head;

while (current.next != null) {

current = current.next;

}

current.next = newTask;

}

}

// Search task by ID

public Task searchTask(String taskId) {

Task current = head;

while (current != null) {

if (current.taskId.equals(taskId)) {

return current;

}

current = current.next;

}

return null;

}

// Traverse tasks

public void displayTasks() {

Task current = head;

while (current != null) {

System.out.println(current);

current = current.next;

}

}

// Delete task by ID

public void deleteTask(String taskId) {

if (head == null) {

System.out.println("Task list is empty.");

return;

}

if (head.taskId.equals(taskId)) {

head = head.next;

System.out.println("Task deleted.");

return;

}

Task current = head;

while (current.next != null && !current.next.taskId.equals(taskId)) {

current = current.next;

}

if (current.next != null) {

current.next = current.next.next;

System.out.println("Task deleted.");

} else {

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

}

}

public static void main(String[] args) {

TaskManager manager = new TaskManager();

manager.addTask("T001", "Finish Assignment", "Pending");

manager.addTask("T002", "Fix Bug #42", "In Progress");

manager.addTask("T003", "Attend Meeting", "Done");

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

manager.displayTasks();

System.out.println("\n Searching for T002:");

Task t = manager.searchTask("T002");

System.out.println(t != null ? t : "Not found");

System.out.println("\n Deleting T001:");

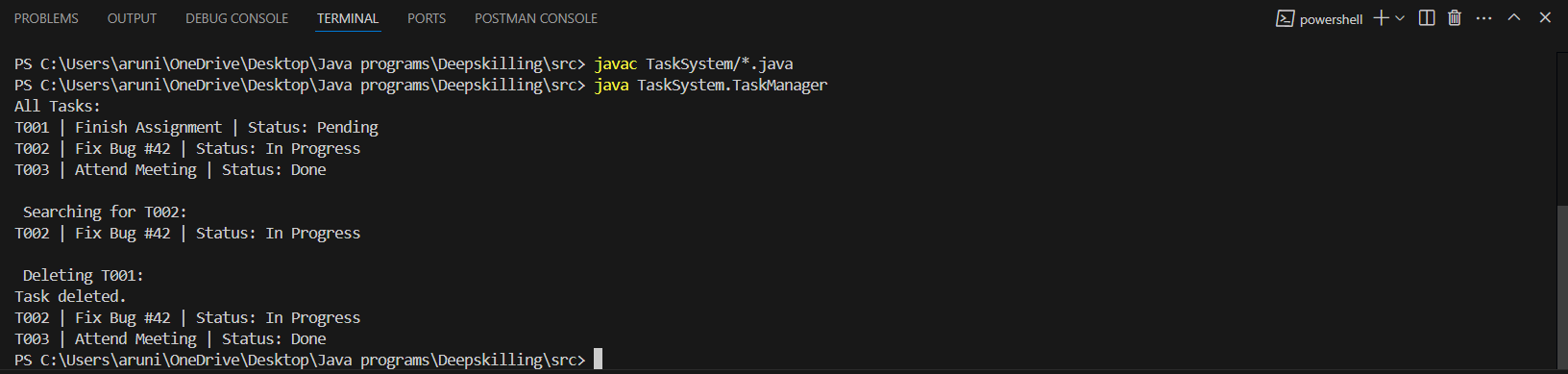
manager.deleteTask("T001");

manager.displayTasks();

}

}

**Output:**

****