**Exercise 1: Inventory Management System**

Code:

import java.util.\*;

class Product {

private int productId;

private String productName;

private int quantity;

private double price;

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

this.productId = productId;

this.productName = productName;

this.quantity = quantity;

this.price = price;

}

public int getProductId() { return productId; }

public void setProductName(String name) { this.productName = name; }

public void setQuantity(int quantity) { this.quantity = quantity; }

public void setPrice(double price) { this.price = price; }

public String toString() {

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

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

}

}

class InventoryManager {

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

public void addProduct(Product product) {

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

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

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

Product p = inventory.get(id);

if (p != null) {

p.setProductName(name);

p.setQuantity(quantity);

p.setPrice(price);

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

} else {

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

}

}

public void deleteProduct(int id) {

Product removed = inventory.remove(id);

if (removed != null)

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

else

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

}

public void listInventory() {

if (inventory.isEmpty()) {

System.out.println("Inventory is empty.");

} else {

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

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

System.out.println(p);

}

}

}}

public class Main {

public static void main(String[] args) {

InventoryManager manager = new InventoryManager();

Product p1 = new Product(101, "Mouse", 50, 499.0);

Product p2 = new Product(102, "Keyboard", 30, 799.0);

Product p3 = new Product(103, "Monitor", 20, 5499.0);

manager.addProduct(p1);

manager.addProduct(p2);

manager.addProduct(p3);

manager.listInventory();

manager.updateProduct(102, "Mechanical Keyboard", 25, 1099.0);

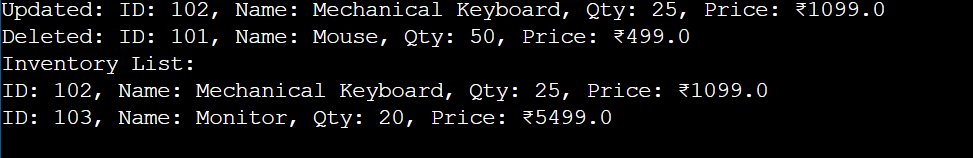
manager.deleteProduct(101);

manager.listInventory();

}

}

Output:



**Exercise 3: Sorting Customer Orders**

Code:

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;

}

public String toString() {

return "OrderID: " + orderId + ", Customer: " + customerName + ", Total: ₹" + totalPrice;

}

}

public class Main {

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);

}

}

public 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 printOrders(Order[] orders) {

for (Order o : orders) {

System.out.println(o);

}

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

Order[] orders = {

new Order(101, "user1", 2999.0),

new Order(102, "user2", 4599.5),

new Order(103, "user3", 1200.0),

new Order(104, "user4", 6200.0),

new Order(105, "user5", 3000.0)

};

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

Order[] bubbleSorted = orders.clone();

bubbleSort(bubbleSorted);

printOrders(bubbleSorted);

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

Order[] quickSorted = orders.clone();

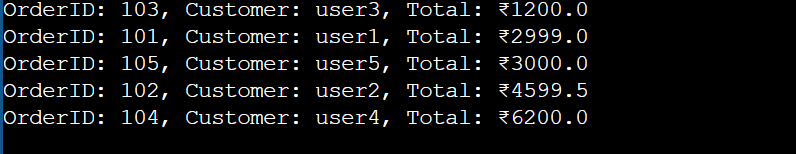
quickSort(quickSorted, 0, quickSorted.length - 1);

printOrders(quickSorted);

}

}

Output:



**Exercise 4: Employee Management System**

Code:

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;

}

public String toString() {

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

}

}

class EmployeeManagementSystem {

private Employee[] employees;

private int size;

public EmployeeManagementSystem(int capacity) {

employees = new Employee[capacity];

size = 0;

}

public void addEmployee(Employee employee) {

if (size < employees.length) {

employees[size++] = employee;

System.out.println("Added: " + employee);

} 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 deleteById(int id) {

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

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

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

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

}

employees[--size] = null;

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

return;

}

}

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

}

public void listEmployees() {

if (size == 0) {

System.out.println("No employees found.");

return;

}

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

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

}

}

}

public class Main {

public static void main(String[] args) {

EmployeeManagementSystem system = new EmployeeManagementSystem(10);

system.addEmployee(new Employee(1, "user1", "Manager", 85000));

system.addEmployee(new Employee(2, "user2", "Developer", 55000));

system.addEmployee(new Employee(3, "user3", "Analyst", 60000));

System.out.println("\nAll Employees:");

system.listEmployees();

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

Employee result = system.searchById(2);

if (result != null) System.out.println(result);

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

System.out.println("\nDeleting ID 1:");

system.deleteById(1);

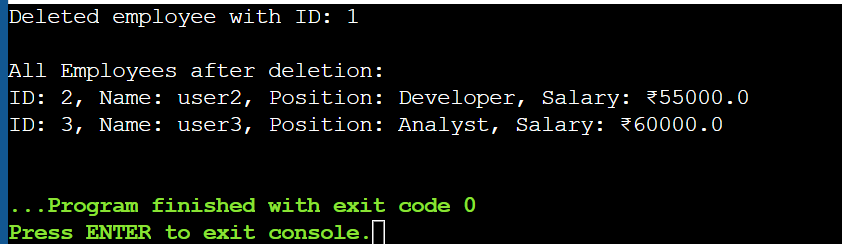
System.out.println("\nAll Employees after deletion:");

system.listEmployees();

}

}

Output:



**Exercise 5: Task Management System**

Code:

class Task {

int taskId;

String taskName;

String status;

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

this.taskId = taskId;

this.taskName = taskName;

this.status = status;

}

public String toString() {

return "TaskID: " + taskId + ", Name: " + taskName + ", Status: " + status;

}

}

class TaskNode {

Task task;

TaskNode next;

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 current = head;

while (current.next != null) {

current = current.next;

}

current.next = newNode;

}

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

public void deleteTask(int taskId) {

if (head == null) return;

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

System.out.println("Deleted: " + head.task);

head = head.next;

return;

}

TaskNode current = head;

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

current = current.next;

}

if (current.next != null) {

System.out.println("Deleted: " + current.next.task);

current.next = current.next.next;

} else {

System.out.println("Task ID " + taskId + " not found.");

}

}

public Task searchTask(int taskId) {

TaskNode current = head;

while (current != null) {

if (current.task.taskId == taskId) return current.task;

current = current.next;

}

return null;

}

public void displayTasks() {

if (head == null) {

System.out.println("No tasks.");

return;

}

TaskNode current = head;

while (current != null) {

System.out.println(current.task);

current = current.next;

}

}

}

public class Main {

public static void main(String[] args) {

TaskList tasks = new TaskList();

tasks.addTask(new Task(1, "Write report", "Pending"));

tasks.addTask(new Task(2, "Fix bug", "In Progress"));

tasks.addTask(new Task(3, "Call client", "Completed"));

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

tasks.displayTasks();

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

Task found = tasks.searchTask(2);

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

System.out.println("\nDeleting Task ID 1:");

tasks.deleteTask(1);

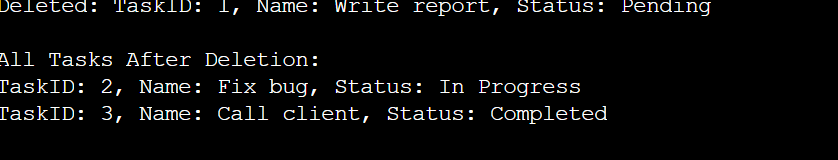
System.out.println("\nAll Tasks After Deletion:");

tasks.displayTasks();

}

}

Output:



**Exercise 6: Library Management System**

Code:

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;

this.author = author;

}

public String toString() {

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

}

}

public class Main {

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

for (Book b : books) {

if (b.title.equalsIgnoreCase(title)) {

return b;

}

} return null;

}

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

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

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

while (left <= right) {

int mid = (left + right) / 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;

}

public static void main(String[] args) {

Book[] books = {

new Book(101, "Atomic Habits", "James Clear"),

new Book(102, "The Alchemist", "Paulo Coelho"),

new Book(103, "Rich Dad Poor Dad", "Robert Kiyosaki"),

new Book(104, "Sapiens", "Yuval Noah Harari"),

new Book(105, "1984", "George Orwell")

};

System.out.println("Linear Search for 'Sapiens':");

Book result1 = linearSearch(books, "Sapiens");

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

System.out.println("\n Binary Search for 'The Alchemist':");

Book result2 = binarySearch(books, "The Alchemist");

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

}}

Output:

