**Exercise 3: Sorting Customer Orders**

**Orders.java**

package com.sort.example;

public class Orders {

private int orderId;

private String customerName;

private double totalPrice;

public Orders(int orderId,String customerName,double totalPrice) {

this.orderId=orderId;

this.customerName=customerName;

this.totalPrice=totalPrice;

}

public double getTotalPrice() {

return totalPrice;

}

public String toString() {

return "OrderID: " + orderId + ", Name: " + customerName + ", Total: " + totalPrice;

}

}

**Bubblesort.java**

package com.sort.example;

public class BubbleSort {

public static void sort(Orders[] 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].getTotalPrice()>orders[j+1].getTotalPrice())

{

Orders temp=orders[j];

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

orders[j+1]=temp;

}

}

}

}

}

**QuickSort.java**

package com.sort.example;

public class QuickSort {

public static void sort(Orders[] orders,int low,int high) {

if(low<high) {

int pi=*partition*(orders,low,high);

*sort*(orders,low,pi-1);

*sort*(orders,pi+1,high);

}

}

private static int partition(Orders[] orders,int low,int high) {

double pivot=orders[high].getTotalPrice();

int i=low-1;

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

if(orders[j].getTotalPrice()<pivot){

i++;

Orders temp=orders[i];

orders[i]=orders[j];

orders[j]=temp;

}

}

Orders temp=orders[i+1];

orders[i+1]=orders[high];

orders[high]=temp;

return i+1;

}

}

**Main.java**

package com.sort.example;

public class Main {

public static void main(String[] args) {

Orders[] orders = {

new Orders(101,"Alice",250.0),

new Orders(102,"Bob",120.5),

new Orders(103,"Charlie",499.9),

new Orders(104,"David",75.0),

new Orders(105,"Eva",320.0)

};

System.*out*.println("Original Orders:");

*printOrders*(orders);

Orders[] bubbleSorted = orders.clone();

BubbleSort.*sort*(bubbleSorted);

System.*out*.println("\nBubble Sorted Orders:");

*printOrders*(bubbleSorted);

Orders[] quickSorted = orders.clone();

QuickSort.*sort*(quickSorted,0,quickSorted.length-1);

System.*out*.println("\nQuick Sorted Orders:");

*printOrders*(quickSorted);

}

private static void printOrders(Orders[] orders) {

for (Orders order : orders) {

System.*out*.println(order);

}

}

}

**Output:**

Original Orders:

OrderID: 101, Name: Alice, Total: 250.0

OrderID: 102, Name: Bob, Total: 120.5

OrderID: 103, Name: Charlie, Total: 499.9

OrderID: 104, Name: David, Total: 75.0

OrderID: 105, Name: Eva, Total: 320.0

Bubble Sorted Orders:

OrderID: 104, Name: David, Total: 75.0

OrderID: 102, Name: Bob, Total: 120.5

OrderID: 101, Name: Alice, Total: 250.0

OrderID: 105, Name: Eva, Total: 320.0

OrderID: 103, Name: Charlie, Total: 499.9

Quick Sorted Orders:

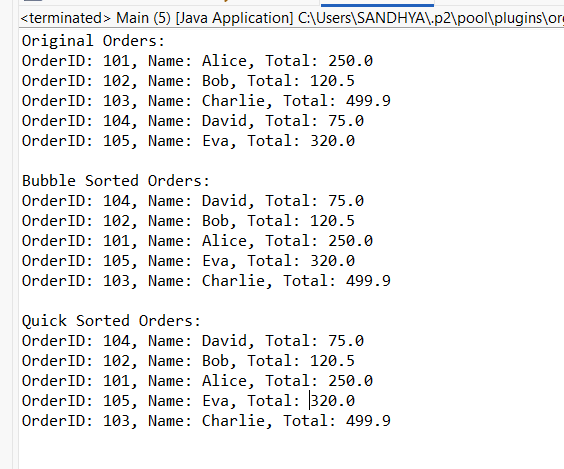
OrderID: 104, Name: David, Total: 75.0

OrderID: 102, Name: Bob, Total: 120.5

OrderID: 101, Name: Alice, Total: 250.0

OrderID: 105, Name: Eva, Total: 320.0

OrderID: 103, Name: Charlie, Total: 499.9



**Exercise 4: Employee Management System**

**Employee.java**

package com.employee.example;

public class Employee {

private int employeeId;

private String name;

private String position;

private 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 int getEmployeeId() {

return employeeId;

}

public String toString() {

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

}

}

**EmployeeManager.java**

package com.employee.example;

public class EmployeeManager {

private Employee[] employees;

private int count;

public EmployeeManager(int size) {

employees=new Employee[size];

count = 0;

}

public void addEmployee(Employee emp) {

if (count<employees.length) {

employees[count++]=emp;

} else {

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

}

}

public Employee searchById(int id) {

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

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

return employees[i];

}

}

return null;

}

public void listAllEmployees() {

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

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

}

}

public boolean deleteEmployee(int id) {

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

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

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

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

}

employees[--count]=null;

return true;

}

}

return false;

}

}

**Main.java**

package com.employee.example;

public class Main {

public static void main(String[] args) {

EmployeeManager manager=new EmployeeManager(5);

manager.addEmployee(new Employee(1,"Alice","Developer",75000));

manager.addEmployee(new Employee(2,"Bob","Manager",85000));

manager.addEmployee(new Employee(3,"Charlie","Tester",65000));

System.*out*.println("All Employees:");

manager.listAllEmployees();

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

Employee emp=manager.searchById(2);

System.*out*.println(emp!=null?emp:"Employee not found");

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

boolean deleted=manager.deleteEmployee(1);

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

System.*out*.println("\nEmployees after deletion:");

manager.listAllEmployees();

}

}

**Output:**

All Employees:

ID: 1, Name: Alice, Position: Developer, Salary: $75000.0

ID: 2, Name: Bob, Position: Manager, Salary: $85000.0

ID: 3, Name: Charlie, Position: Tester, Salary: $65000.0

Searching for Employee with ID 2:

ID: 2, Name: Bob, Position: Manager, Salary: $85000.0

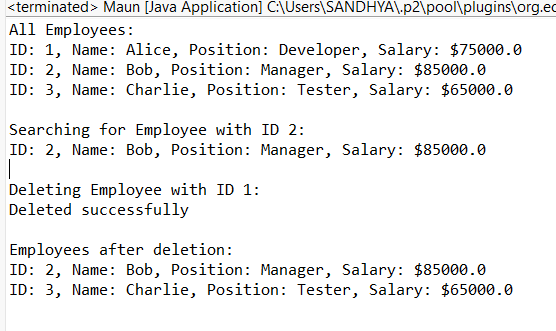
Deleting Employee with ID 1:

Deleted successfully

Employees after deletion:

ID: 2, Name: Bob, Position: Manager, Salary: $85000.0

ID: 3, Name: Charlie, Position: Tester, Salary: $65000.0



**Exercise 5: Task Management System**

**Task.java**

package com.task.example;

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

}}

**Node.java**

package com.task.example;

public class Node {

Task task;

Node next;

public Node(Task task) {

this.task = task;

this.next = null;

}

}

**NodeLinkedList.java**

package com.task.example;

public class TaskLinkedList {

private Node head;

public void addTask(Task task) {

Node newNode=new Node(task);

if (head==null) {

head=newNode;

} else {

Node temp=head;

while(temp.next!=null)

temp=temp.next;

temp.next=newNode;

}

}

public Task searchTask(int taskId) {

Node current=head;

while(current!=null) {

if(current.task.taskId==taskId)

return current.task;

current=current.next;

}

return null;

}

public void listAllTasks() {

Node temp=head;

while(temp!=null) {

System.*out*.println(temp.task);

temp=temp.next;

}

}

public boolean deleteTask(int taskId) {

if(head==null) return false;

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

head=head.next;

return true;

}

Node prev=head;

Node curr=head.next;

while(curr!=null) {

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

prev.next=curr.next;

return true;

}

prev=curr;

curr=curr.next;

}

return false;

}

}

**Main.java**

package com.task.example;

public class Main {

public static void main(String[] args) {

TaskLinkedList taskList=new TaskLinkedList();

taskList.addTask(new Task(101,"Design UI","Pending"));

taskList.addTask(new Task(102,"Develop Backend","In Progress"));

taskList.addTask(new Task(103,"Test Modules","Not Started"));

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

taskList.listAllTasks();

System.*out*.println("\nSearching for Task ID 102:");

Task found=taskList.searchTask(102);

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

System.*out*.println("\nDeleting Task ID 101:");

System.*out*.println(deleted?"Task deleted." : "Task not found.");

System.*out*.println("\nTasks after deletion:");

taskList.listAllTasks();

}

}

**Output:**

All Tasks:

TaskID: 101, Name: Design UI, Status: Pending

TaskID: 102, Name: Develop Backend, Status: In Progress

TaskID: 103, Name: Test Modules, Status: Not Started

Searching for Task ID 102:

TaskID: 102, Name: Develop Backend, Status: In Progress

Deleting Task ID 101:

Task deleted.

Tasks after deletion:

TaskID: 102, Name: Develop Backend, Status: In Progress

TaskID: 103, Name: Test Modules, Status: Not Started

