**DATA STRUCTURES AND ALGORITHMS (Additional Problem)**

**Exercise 1: Inventory Management System**

**CODE:**

import java.util.HashMap;

public class InventoryManagementSystem {

static 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;

}

public String toString() {

return "id: " + productid + ", name: " + productname + ", quantity: " + quantity + ", price: " + price;

}

}

static class inventory {

HashMap<Integer, product> store;

public inventory() {

store = new HashMap<>();

}

public void addproduct(product p) {

if (store.containsKey(p.productid)) {

System.out.println("product with id " + p.productid + " already exists.");

} else {

store.put(p.productid, p);

System.out.println("product added: " + p.productname);

}

}

public void updateproduct(int id, int newqty, double newprice) {

product p = store.get(id);

if (p != null) {

p.quantity = newqty;

p.price = newprice;

System.out.println("product updated: " + p.productname);

} else {

System.out.println("product not found with id: " + id);

}

}

public void deleteproduct(int id) {

product removed = store.remove(id);

if (removed != null) {

System.out.println("deleted product: " + removed.productname);

} else {

System.out.println("product not found with id: " + id);

}

}

public void displayinventory() {

if (store.isEmpty()) {

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

return;

}

System.out.println("current inventory:");

for (product p : store.values()) {

System.out.println(p);

}

}

}

public static void main(String[] args) {

inventory inv = new inventory();

inv.addproduct(new product(101, "keyboard", 10, 799.50));

inv.addproduct(new product(102, "mouse", 20, 499.99));

inv.addproduct(new product(103, "monitor", 5, 7500.00));

System.out.println("\n--- inventory after addition ---");

inv.displayinventory();

inv.updateproduct(102, 25, 479.99);

System.out.println("\n--- inventory after update ---");

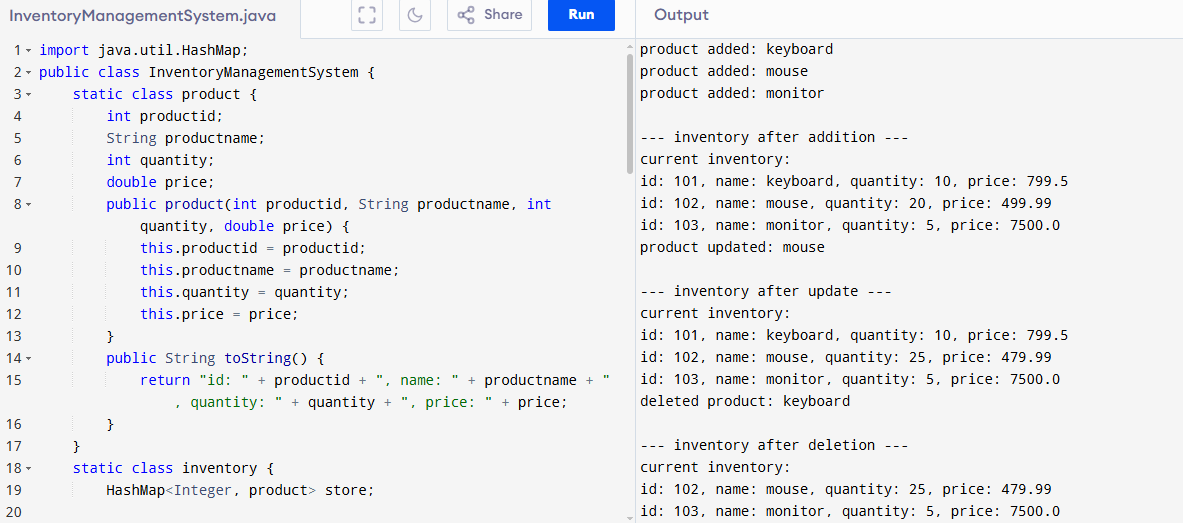
inv.displayinventory();

inv.deleteproduct(101);

System.out.println("\n--- inventory after deletion ---");

inv.displayinventory();}}

**OUTPUT:**

****

**Exercise 3: Sorting Customer Orders**

**CODE:**

import java.util.Scanner;

public class ordersort {

static 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 "id: " + orderid + ", name: " + customername + ", price: " + totalprice;

}

}

public static void bubblesort(order[] arr) {

int n = arr.length;

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

boolean swapped = false;

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

if (arr[j].totalprice > arr[j + 1].totalprice) {

order temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

swapped = true;

}

}

if (!swapped) break;

}

}

public 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);

}

}

public 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 static void printorders(order[] arr) {

for (order o : arr) {

System.out.println(o);

}

}

public static void main(String[] args) {

order[] arr = {

new order(301, "alice", 2500.0),

new order(302, "bob", 1899.5),

new order(303, "carol", 3299.9),

new order(304, "david", 1200.0),

new order(305, "emma", 4599.0)

};

System.out.println("original orders:");

printorders(arr);

System.out.println("\norders after bubble sort:");

order[] bubblearr = arr.clone();

bubblesort(bubblearr);

printorders(bubblearr);

System.out.println("\norders after quick sort:");

order[] quickarr = arr.clone();

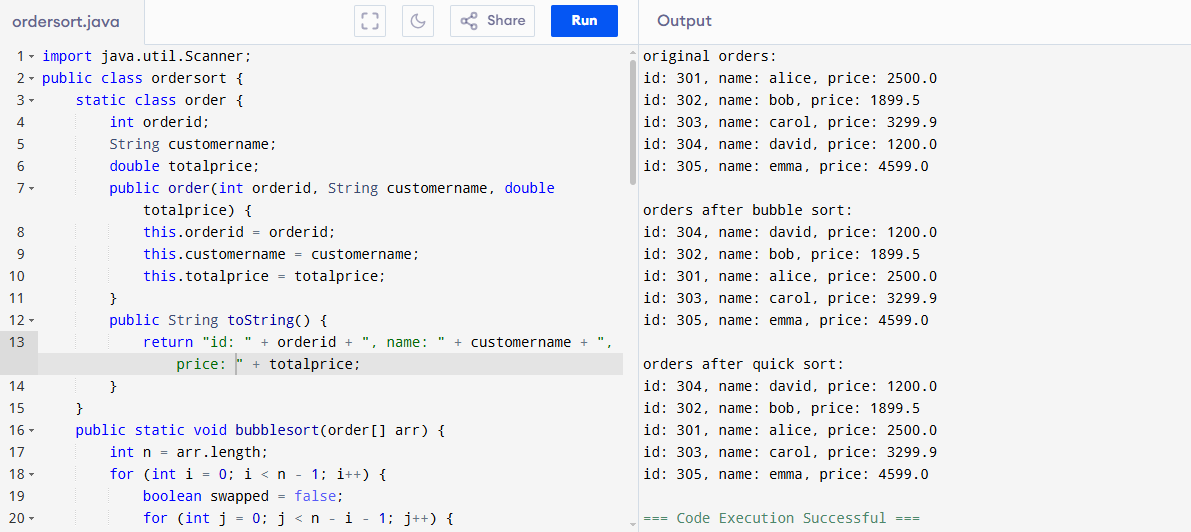
quicksort(quickarr, 0, quickarr.length - 1);

printorders(quickarr);

}

}

**OUTPUT:**

****

**Exercise 4: Employee Management System**

**CODE:**

import java.util.Scanner;

public class employeemanagement {

static 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;

}

}

static class system {

employee[] employees;

int count;

public system(int size) {

employees = new employee[size];

count = 0;

}

public void addemployee(employee e) {

if (count < employees.length) {

employees[count++] = e;

System.out.println("employee added.");

} else {

System.out.println("array full. cannot add.");

}

}

public employee searchemployee(int id) {

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

if (employees[i].employeeid == id) return employees[i];

}

return null;

}

public void deleteemployee(int id) {

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

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

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

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

}

employees[count - 1] = null;

count--;

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

return;

}

}

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

}

public void displayemployees() {

if (count == 0) {

System.out.println("no employees.");

return;

}

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

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

}

}

}

public static void main(String[] args) {

system s = new system(10);

s.addemployee(new employee(1, "alice", "developer", 50000));

s.addemployee(new employee(2, "bob", "manager", 70000));

s.addemployee(new employee(3, "carol", "designer", 45000));

System.out.println("\nall employees:");

s.displayemployees();

System.out.println("\nsearch employee id 2:");

employee e = s.searchemployee(2);

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

System.out.println("\ndelete employee id 1:");

s.deleteemployee(1);

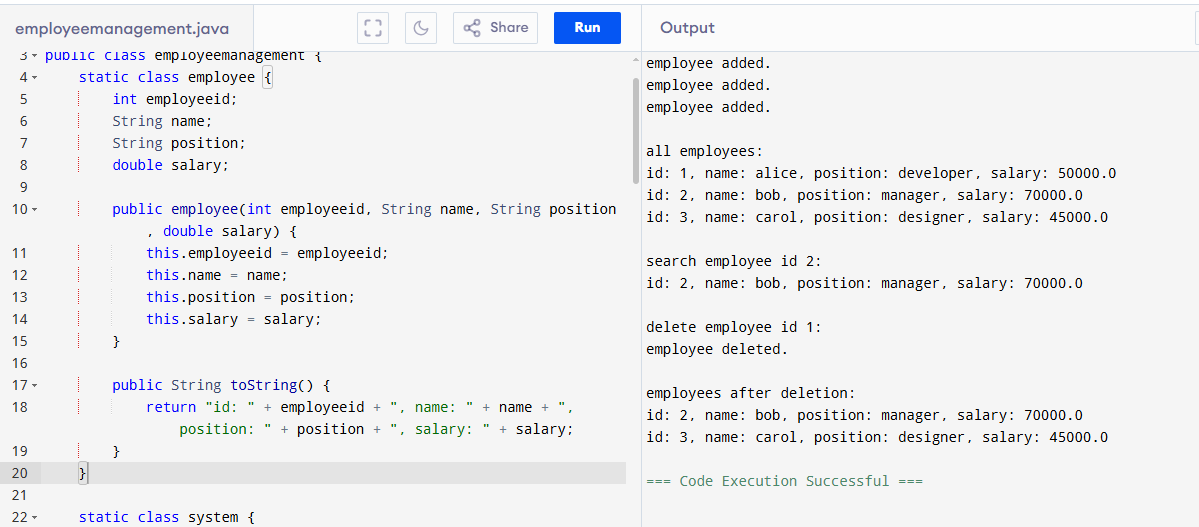
System.out.println("\nemployees after deletion:");

s.displayemployees();

}

}

**OUTPUT:**

****

**Exercise 5: Task Management System**

**CODE:**

public class taskmanager {

static class task {

int taskid;

String taskname;

String status;

task next;

public task(int taskid, String taskname, String status) {

this.taskid = taskid;

this.taskname = taskname;

this.status = status;

this.next = null;

}

public String toString() {

return "id: " + taskid + ", name: " + taskname + ", status: " + status;

}

}

static class tasklist {

task head;

public void addtask(int id, String name, String status) {

task newtask = new task(id, name, status);

if (head == null) {

head = newtask;

} else {

task temp = head;

while (temp.next != null) temp = temp.next;

temp.next = newtask;

}

System.out.println("task added.");

}

public task searchtask(int id) {

task temp = head;

while (temp != null) {

if (temp.taskid == id) return temp;

temp = temp.next;

}

return null;

}

public void deletetask(int id) {

if (head == null) {

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

return;

}

if (head.taskid == id) {

head = head.next;

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

return;

}

task temp = head;

while (temp.next != null && temp.next.taskid != id) {

temp = temp.next;

}

if (temp.next != null) {

temp.next = temp.next.next;

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

} else {

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

}

}

public void traversetasks() {

if (head == null) {

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

return;

}

task temp = head;

while (temp != null) {

System.out.println(temp);

temp = temp.next;

}

}

}

public static void main(String[] args) {

tasklist list = new tasklist();

list.addtask(1, "design module", "pending");

list.addtask(2, "write code", "in progress");

list.addtask(3, "test system", "completed");

System.out.println("\nall tasks:");

list.traversetasks();

System.out.println("\nsearch task id 2:");

task t = list.searchtask(2);

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

System.out.println("\ndelete task id 1:");

list.deletetask(1);

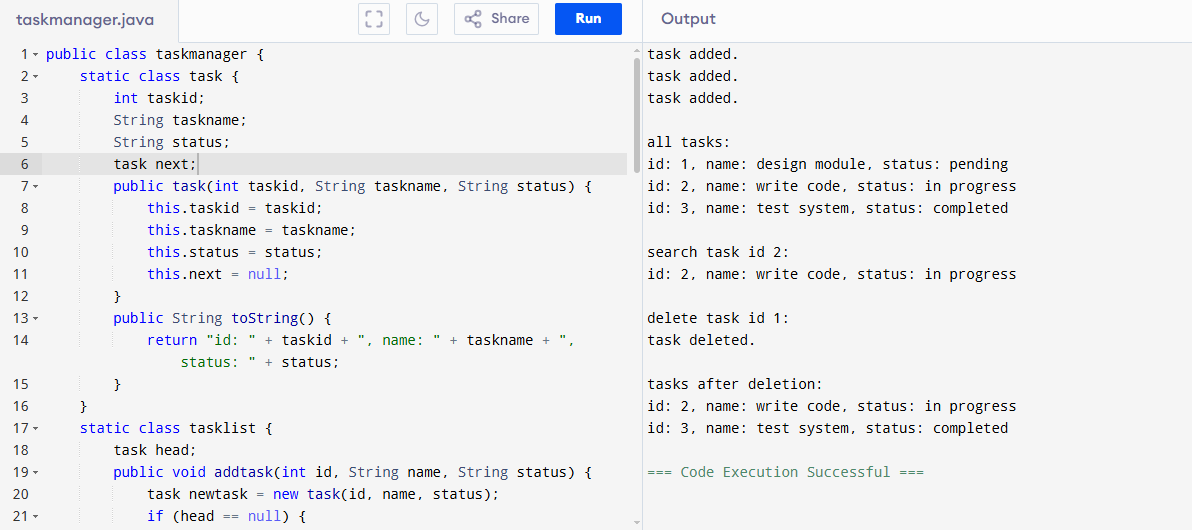
System.out.println("\ntasks after deletion:");

list.traversetasks();

}

}

**OUTPUT:**

****

**Exercise 6: Library Management System**

**CODE:**

import java.util.Arrays;

public class librarysystem {

static class book implements Comparable<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 int compareTo(book other) {

return this.title.compareToIgnoreCase(other.title);

}

}

static class library {

book[] books;

int count;

public library(int size) {

books = new book[size];

count = 0;

}

public void addbook(book b) {

if (count < books.length) {

books[count++] = b;

System.out.println("book added.");

} else {

System.out.println("library full.");

}

}

public void linearsearch(String title) {

boolean found = false;

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

if (books[i].title.equalsIgnoreCase(title)) {

System.out.println("found: " + books[i]);

found = true;

}

}

if (!found) System.out.println("book not found.");

}

public void binarysearch(String title) {

Arrays.sort(books, 0, count);

int low = 0, high = count - 1;

while (low <= high) {

int mid = (low + high) / 2;

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

if (cmp == 0) {

System.out.println("found: " + books[mid]);

return;

} else if (cmp < 0) {

low = mid + 1;

} else {

high = mid - 1;

}

}

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

}

public void displaybooks() {

if (count == 0) {

System.out.println("no books.");

return;

}

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

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

}

}

}

public static void main(String[] args) {

library lib = new library(10);

lib.addbook(new book(1, "java programming", "james gosling"));

lib.addbook(new book(2, "data structures", "narsimha"));

lib.addbook(new book(3, "algorithms", "robert sedgewick"));

System.out.println("\nall books:");

lib.displaybooks();

System.out.println("\nlinear search for 'data structures':");

lib.linearsearch("data structures");

System.out.println("\nbinary search for 'java programming':");

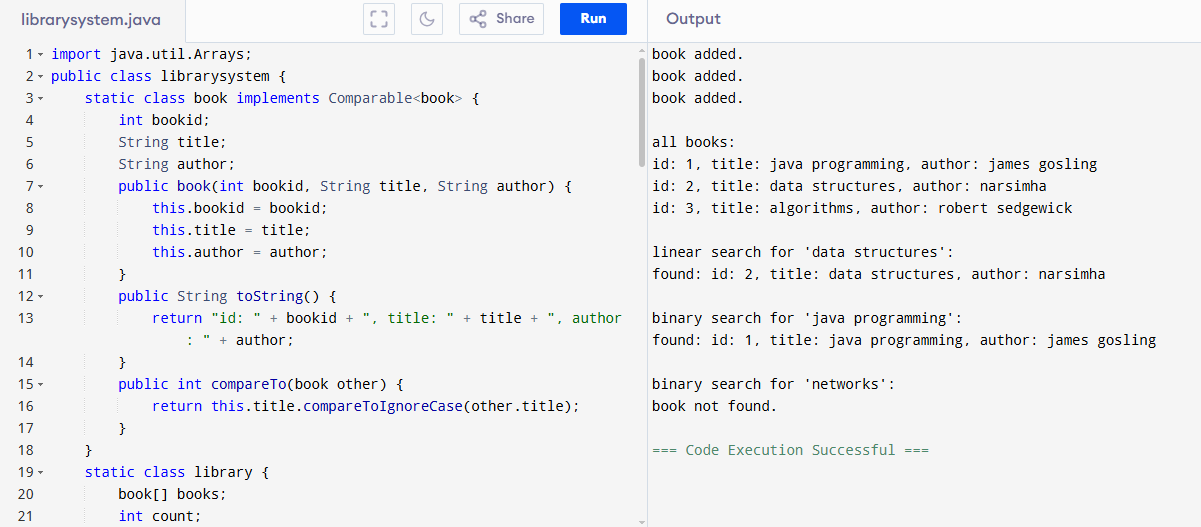
lib.binarysearch("java programming");

System.out.println("\nbinary search for 'networks':");

lib.binarysearch("networks");

}}

**OUTPUT:**

****