**Exercise 1: Inventory Management System**

import java.util.HashMap;

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 void display(){

System.out.println(productId + " | " + productName + " | " + quantity + " $ |" + price);

}

}

class InventoryManager {

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

public void addProduct(Product p){

inventory.put(p.productId,p);

}

public void updateProduct(int id,int qty, double price){

Product p = inventory.get(id);

if(p!=null){

p.quantity = qty;

p.price = price;

}

}

public void deleteProduct(int id){

inventory.remove(id);

}

public void showInventory(){

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

p.display();

}

}

}

public class Main {

public static void main(String[] args) {

InventoryManager manager = new InventoryManager();

manager.addProduct(new Product(1, "Keyboard", 10, 999.99));

manager.addProduct(new Product(2, "Mouse", 25, 499.50));

manager.updateProduct(1, 15, 950.00);

manager.deleteProduct(2);

manager.showInventory();

}

}

**Output**



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

import java.util.Arrays;

import java.util.Comparator;

class Product {

int productId;

String productName;

String category;

public Product(int id,String productName, String category){

this.productId = id;

this.productName = productName;

this.category = category;

}

public void display(){

System.out.println(productId + " | " + productName + " | " + category);

}

}

public class Main{

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

for(Product p: products){

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

return p;

}

}

return null;

}

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 = products[mid].productName.compareToIgnoreCase(name);

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

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

else right = mid - 1;

}

return null;

}

public static void main(String[] args){

Product[] catalog = {

new Product(101,"Laptop","Electronics"),

new Product(102,"Sneakers","Footwear"),

new Product(103,"Mouse","Accessories"),

new Product(104,"Backpack","Bags")

};

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

Product found1 = linearSearch(catalog,"Mouse");

if(found1!=null) found1.display();

System.out.println("\nBinary Search Results: ");

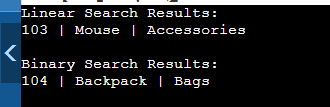
Product found2 = binarySearch(catalog,"Backpack");

if(found2!=null) found2.display();

}

}

**Output**



**Exercise 3: Sorting Customer Orders**

**Quick sort**

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 + " " + customerName + " $ " + totalPrice;

}

}

public class Main{

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);

}

}

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 tmp = orders[i];

orders[i] = orders[j];

orders[j] = tmp;

}

}

Order tmp = orders[i+1];

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

orders[high] = tmp;

return i+1;

}

public static void main(String[] args) {

Order[] orders = {

new Order(101, "Alice", 99.99),

new Order(102, "Bob", 149.49),

new Order(103, "Charlie", 79.75)

};

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

for (Order o : orders) {

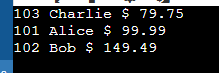
System.out.println(o);

}

}

}

**Output**



**Bubble Sort**

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 + " " + customerName + " $ " + totalPrice;

}

}

// Bubble Sort

public class Main{

static void bubbleSort(Order[] orders){

int len = orders.length;

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

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

if(orders[j].totalPrice > orders[j+1].totalPrice){

Order tmp = orders[j];

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

orders[j+1] = tmp;

}

}

}

}

public static void main(String[] args){

Order[] orders = {

new Order(101,"Alice",99.99),

new Order(102,"Prabha",149.99),

new Order(103,"Karthick",79.75)

};

bubbleSort(orders);

for(Order o: orders){

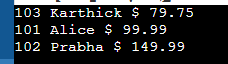
System.out.println(o);

}

}

}

**Output**



**Exercise 4: Employee Management System**

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 "[ " + employeeId + " ] " + name + " - " + position + " - Rs." + salary;

}

}

class EmployeeManagementSystem{

private Employee[] employees;

private int size;

public EmployeeManagementSystem(int capacity){

employees = new Employee[capacity];

size = 0;

}

public void addEmployee(Employee emp){

if(size < employees.length){

employees[size++] = emp;

System.out.println("Employee added!");

}

else{

System.out.println("Array is full. Cannot add more employees");

}

}

public void viewAll(){

if(size==0){

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

return;

}

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

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

}

}

public Employee search(int empId){

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

if(employees[j].employeeId == empId) return employees[j];

}

return null;

}

public boolean delete(int empId){

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

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

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

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

}

employees[--size] = null;

return true;

}

}

return false;

}

}

public class Main{

public static void main(String[] args){

Scanner in = new Scanner(System.in);

EmployeeManagementSystem ems=new EmployeeManagementSystem(100);

while(true){

System.out.println("\n--- Employee Management System ---");

System.out.println("1. Add Employee");

System.out.println("2. View All Employees");

System.out.println("3. Search Employee by ID");

System.out.println("4. Delete Employee by ID");

System.out.println("5. Exit");

System.out.print("Enter your choice: ");

int ch = in.nextInt();

switch(ch){

case 1:

System.out.println("Enter ID: ");

int id = in.nextInt();

in.nextLine();

System.out.print("Enter Name: ");

String name = in.nextLine();

System.out.print("Enter position: ");

String position = in.nextLine();

System.out.print("Enter Salary: ");

double salary = in.nextDouble();

ems.addEmployee(new Employee(id, name, position, salary));

break;

case 2:

ems.viewAll();

break;

case 3:

System.out.print("Enter Employee ID to search: ");

Employee found = ems.search(in.nextInt());

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

break;

case 4:

System.out.print("Enter Employee ID to delete: ");

System.out.println(ems.delete(in.nextInt()) ? "Deleted." : "Not found.");

break;

case 5:

System.out.println("Exiting.");

return;

default:

System.out.println("Invalid choice.");

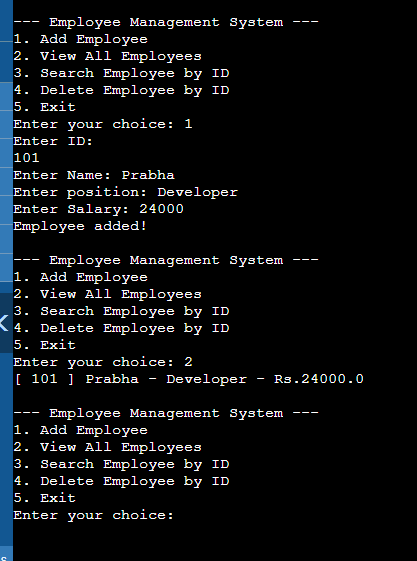
}

}

}

}

**Output**



**Exercise 5: Task Management System**

import java.util.Scanner;

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;

}

}

class TaskManager {

Task head = null;

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

Task newTask = new Task(id, name, status);

if (head == null) {

head = newTask;

} else {

Task current = head;

while (current.next != null) {

current = current.next;

}

current.next = newTask;

}

}

public Task searchTask(int id) {

Task current = head;

while (current != null) {

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

current = current.next;

}

return null;

}

public void deleteTask(int id) {

if (head == null) return;

if (head.taskId == id) {

head = head.next;

return;

}

Task prev = null, current = head;

while (current != null && current.taskId != id) {

prev = current;

current = current.next;

}

if (current != null) prev.next = current.next;

}

public void displayTasks() {

Task current = head;

if (current == null) {

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

return;

}

while (current != null) {

System.out.println("Task ID: " + current.taskId + ", Name: " + current.taskName + ", Status: " + current.status);

current = current.next;

}

}

}

public class Main {

public static void main(String[] args) {

TaskManager tm = new TaskManager();

Scanner sc = new Scanner(System.in);

while (true) {

System.out.println("\n1. Add Task\n2. Delete Task\n3. Search Task\n4. Display Tasks\n5. Exit");

int choice = sc.nextInt();

switch (choice) {

case 1:

System.out.print("Enter Task ID: ");

int id = sc.nextInt();

sc.nextLine();

System.out.print("Enter Task Name: ");

String name = sc.nextLine();

System.out.print("Enter Status: ");

String status = sc.nextLine();

tm.addTask(id, name, status);

break;

case 2:

System.out.print("Enter Task ID to delete: ");

int deleteId = sc.nextInt();

tm.deleteTask(deleteId);

break;

case 3:

System.out.print("Enter Task ID to search: ");

int searchId = sc.nextInt();

Task t = tm.searchTask(searchId);

if (t != null) {

System.out.println("Found - Task ID: " + t.taskId + ", Name: " + t.taskName + ", Status: " + t.status);

} else {

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

}

break;

case 4:

tm.displayTasks();

break;

case 5:

System.out.println("Exiting...");

sc.close();

return;

default:

System.out.println("Invalid option.");

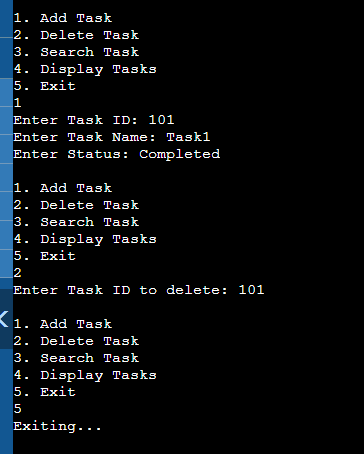
}

}

}

}

**Output**



**Exercise 6: Library Management System**

import java.util.\*;

class Book {

int bookId;

String title;

String author;

public Book(int bookId, String title, String author) {

this.bookId = bookId;

this.title = title.toLowerCase();

this.author = author.toLowerCase();

}

public void print() {

System.out.println("[" + bookId + "] " + title + " by " + author);

}

}

public class Main {

public static List<Book> linearSearch(List<Book> books, String query) {

List<Book> result = new ArrayList<>();

for (Book book : books) {

if (book.title.contains(query.toLowerCase())) {

result.add(book);

}

}

return result;

}

public static Book binarySearch(List<Book> sortedBooks, String query) {

int left = 0, right = sortedBooks.size() - 1;

query = query.toLowerCase();

while (left <= right) {

int mid = (left + right) / 2;

int cmp = sortedBooks.get(mid).title.compareTo(query);

if (cmp == 0) return sortedBooks.get(mid);

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

else right = mid - 1;

}

return null;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

List<Book> books = new ArrayList<>();

books.add(new Book(1, "Thirukkural", "Tiruvalluvar"));

books.add(new Book(2, "Athisudi", "Avviyar"));

books.add(new Book(3, "Wings of Fire", "Abdul Kalam"));

books.add(new Book(4, "Harry Potter", "J. K. Rowling"));

System.out.println("Search by title (Linear): ");

String keyword = sc.nextLine();

List<Book> linearResults = linearSearch(books, keyword);

if (linearResults.isEmpty()) System.out.println("No match found.");

else linearResults.forEach(Book::print);

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

System.out.println("\nBinary search (Exact title match): ");

String exactTitle = sc.nextLine();

Book result = binarySearch(books, exactTitle);

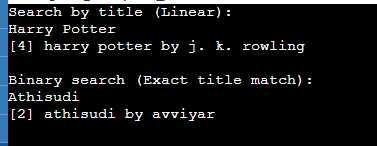
if (result == null) System.out.println("Book not found.");

else result.print();

}

}

**Output**



**Exercise 7: Financial Forecasting**

import java.util.Scanner;

public class Main {

public static double forecast(double presentValue, double rate, int years) {

if (years == 0) return presentValue;

return forecast(presentValue, rate, years - 1) \* (1 + rate);

}

public static double forecastMemo(double presentValue, double rate, int years, double[] memo) {

if (years == 0) return presentValue;

if (memo[years] != 0) return memo[years];

memo[years] = forecastMemo(presentValue, rate, years - 1, memo) \* (1 + rate);

return memo[years];

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter present value: ");

double pv = sc.nextDouble();

System.out.println("Enter annual growth rate (e.g. 0.1 for 10%): ");

double r = sc.nextDouble();

System.out.println("Enter number of years to forecast: ");

int n = sc.nextInt();

System.out.println("\nRecursive Forecast: " + forecast(pv, r, n));

double[] memo = new double[n + 1];

System.out.println("Memoized Forecast: " + forecastMemo(pv, r, n, memo));

}

}

**Output**

