**Exercise 1: Inventory Management System**

**Code:**

import java.util.\*;

// Product class

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;

    }

}

// Inventory Manager

class InventoryManager {

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

    public void addProduct(Product product) {

        if (inventory.containsKey(product.productId)) {

            System.out.println("Product ID already exists.");

        } else {

            inventory.put(product.productId, product);

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

        }

    }

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

        if (inventory.containsKey(id)) {

            Product p = inventory.get(id);

            p.quantity = qty;

            p.price = price;

            System.out.println("Product updated.");

        } else {

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

        }

    }

    public void deleteProduct(int id) {

        if (inventory.containsKey(id)) {

            inventory.remove(id);

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

        } else {

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

        }

    }

    public void displayAllProducts() {

        if (inventory.isEmpty()) {

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

        } else {

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

                System.out.println(p);

            }

        }

    }

}

// Main class

public class InventorySystem {

    public static void main(String[] args) {

        InventoryManager manager = new InventoryManager();

        Scanner sc = new Scanner(System.in);

        while (true) {

            System.out.println("\nInventory Management System");

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

            System.out.println("2. Update Product");

            System.out.println("3. Delete Product");

            System.out.println("4. Display All Products");

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

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

            int choice = sc.nextInt();

            switch (choice) {

                case 1:

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

                    int id = sc.nextInt();

                    sc.nextLine();

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

                    String name = sc.nextLine();

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

                    int qty = sc.nextInt();

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

                    double price = sc.nextDouble();

                    manager.addProduct(new Product(id, name, qty, price));

                    break;

                case 2:

                    System.out.print("Enter Product ID to update: ");

                    int uid = sc.nextInt();

                    System.out.print("Enter New Quantity: ");

                    int newQty = sc.nextInt();

                    System.out.print("Enter New Price: ");

                    double newPrice = sc.nextDouble();

                    manager.updateProduct(uid, newQty, newPrice);

                    break;

                case 3:

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

                    int did = sc.nextInt();

                    manager.deleteProduct(did);

                    break;

                case 4:

                    manager.displayAllProducts();

                    break;

                case 5:

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

                    sc.close();

                    return;

                default:

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

            }

        }

    }

}

**Output:**

**A black rectangle with white dots

AI-generated content may be incorrect.**

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

**Code:**

import java.util.\*;

class Product {

    int productId;

    String productName;

    String category;

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

        this.productId = id;

        this.productName = name;

        this.category = category;

    }

    @Override

    public String toString() {

        return "[" + productId + "] " + productName + " (" + category + ")";

    }

}

public class ProductSearchSystem {

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

        for (Product p : products) {

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

                return p;

            }

        }

        return null;

    }

    // Binary search by product name (array must be sorted)

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

        int left = 0;

        int right = products.length - 1;

        while (left <= right) {

            int mid = (left + right) / 2;

            int compare = products[mid].productName.compareToIgnoreCase(targetName);

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

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

            else right = mid - 1;

        }

        return null;

    }

    public static void main(String[] args) {

        Product[] productList = {

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

            new Product(102, "Keyboard", "Electronics"),

            new Product(103, "Shirt", "Clothing"),

            new Product(104, "Laptop", "Electronics"),

            new Product(105, "Shoes", "Footwear")

        };

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter product name to search: ");

        String name = sc.nextLine();

        System.out.println("\nUsing Linear Search:");

        Product foundLinear = linearSearch(productList, name);

        if (foundLinear != null) {

            System.out.println("Found: " + foundLinear);

        } else {

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

        }

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

        System.out.println("\nUsing Binary Search:");

        Product foundBinary = binarySearch(productList, name);

        if (foundBinary != null) {

            System.out.println("Found: " + foundBinary);

        } else {

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

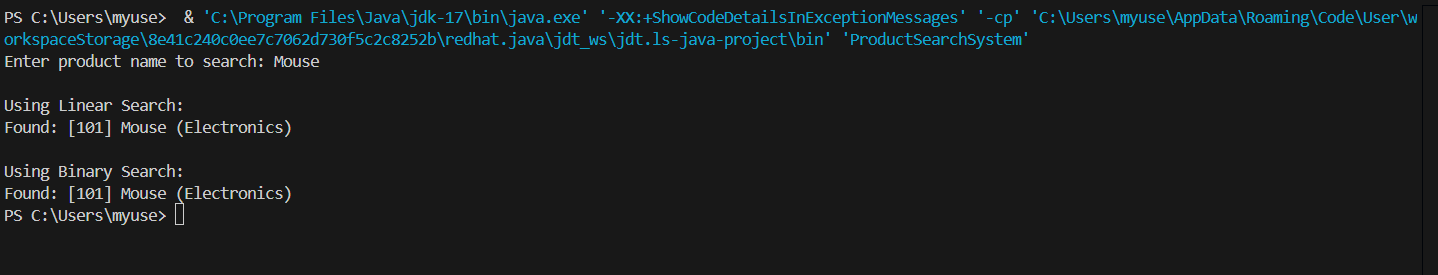
        }

        sc.close();

    }

}

**Output:**



**Exercise 3: Sorting Customer Orders**

**Code:**

import java.util.Scanner;

class Order {

    int orderId;

    String customerName;

    double totalPrice;

    public Order(int id, String name, double price) {

        this.orderId = id;

        this.customerName = name;

        this.totalPrice = price;

    }

    @Override

    public String toString() {

        return "[" + orderId + "] " + customerName + " - ₹" + totalPrice;

    }

}

public class OrderSorting {

    public static void bubbleSort(Order[] orders) {

        int n = orders.length;

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

            boolean swapped = false;

            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;

                    swapped = true;

                }

            }

            if (!swapped) break;

        }

    }

    public static void quickSort(Order[] orders, int low, int high) {

        if (low < high) {

            int pivotIndex = partition(orders, low, high);

            quickSort(orders, low, pivotIndex - 1);

            quickSort(orders, pivotIndex + 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;

    }

    public static void printOrders(Order[] orders) {

        for (Order o : orders) {

            System.out.println(o);

        }

    }

    public static void main(String[] args) {

        Order[] orders = {

            new Order(1001, "Alice", 550.0),

            new Order(1002, "Bob", 2000.0),

            new Order(1003, "Charlie", 750.0),

            new Order(1004, "Daisy", 1200.0),

            new Order(1005, "Evan", 300.0)

        };

        Scanner sc = new Scanner(System.in);

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

        printOrders(orders);

        System.out.print("\nChoose sorting algorithm (1 = Bubble Sort, 2 = Quick Sort): ");

        int choice = sc.nextInt();

        if (choice == 1) {

            bubbleSort(orders);

            System.out.println("\nSorted Orders using Bubble Sort:");

        } else {

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

            System.out.println("\nSorted Orders using Quick Sort:");

        }

        printOrders(orders);

        sc.close();

    }

}

**Output:**

**A computer screen with white text

AI-generated content may be incorrect.**

**Exercise 4: Employee Management System**

**Code:**

import java.util.Scanner;

class Employee {

    int employeeId;

    String name;

    String position;

    double salary;

    public Employee(int id, String name, String position, double salary) {

        this.employeeId = id;

        this.name = name;

        this.position = position;

        this.salary = salary;

    }

    @Override

    public String toString() {

        return "[" + employeeId + "] " + name + " - " + position + " - ₹" + salary;

    }

}

public class EmployeeManagementSystem {

    static final int MAX\_EMPLOYEES = 100;

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

    static int count = 0;

    public static void addEmployee(Employee e) {

        if (count < MAX\_EMPLOYEES) {

            employees[count++] = e;

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

        } else {

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

        }

    }

    public static Employee searchEmployee(int id) {

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

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

                return employees[i];

            }

        }

        return null;

    }

    public static void displayEmployees() {

        if (count == 0) {

            System.out.println("No employees to display.");

            return;

        }

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

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

        }

    }

    public static boolean 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] = null;

                return true;

            }

        }

        return false;

    }

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        while (true) {

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

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

            System.out.println("2. Search Employee");

            System.out.println("3. Display All Employees");

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

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

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

            int option = sc.nextInt();

            sc.nextLine(); // Clear input buffer

            switch (option) {

                case 1:

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

                    int id = sc.nextInt();

                    sc.nextLine();

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

                    String name = sc.nextLine();

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

                    String pos = sc.nextLine();

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

                    double sal = sc.nextDouble();

                    addEmployee(new Employee(id, name, pos, sal));

                    break;

                case 2:

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

                    int sid = sc.nextInt();

                    Employee found = searchEmployee(sid);

                    if (found != null)

                        System.out.println("Found: " + found);

                    else

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

                    break;

                case 3:

                    displayEmployees();

                    break;

                case 4:

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

                    int did = sc.nextInt();

                    if (deleteEmployee(did))

                        System.out.println("Deleted successfully.");

                    else

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

                    break;

                case 5:

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

                    sc.close();

                    return;

                default:

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

            }

        }

    }

}

**Output:**

**A computer screen with blue lines

AI-generated content may be incorrect.**

**Exercise 5: Task Management System**

**Code:**

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;

    }

    @Override

    public String toString() {

        return "[" + taskId + "] " + taskName + " - " + status;

    }

}

public class TaskManagementSystem {

    Task head = null;

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

        Task newTask = new Task(taskId, taskName, 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 successfully!");

    }

    public Task searchTask(int taskId) {

        Task temp = head;

        while (temp != null) {

            if (temp.taskId == taskId)

                return temp;

            temp = temp.next;

        }

        return null;

    }

    public boolean deleteTask(int taskId) {

        if (head == null) return false;

        if (head.taskId == taskId) {

            head = head.next;

            return true;

        }

        Task current = head;

        Task previous = null;

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

            previous = current;

            current = current.next;

        }

        if (current == null) return false;

        previous.next = current.next;

        return true;

    }

    public void displayTasks() {

        if (head == null) {

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

            return;

        }

        Task temp = head;

        while (temp != null) {

            System.out.println(temp);

            temp = temp.next;

        }

    }

    public static void main(String[] args) {

        TaskManagementSystem tms = new TaskManagementSystem();

        Scanner sc = new Scanner(System.in);

        while (true) {

            System.out.println("\n=== Task Management System ===");

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

            System.out.println("2. Search Task");

            System.out.println("3. Display All Tasks");

            System.out.println("4. Delete Task");

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

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

            int choice = sc.nextInt();

            sc.nextLine(); // clear buffer

            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();

                    tms.addTask(id, name, status);

                    break;

                case 2:

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

                    int searchId = sc.nextInt();

                    Task found = tms.searchTask(searchId);

                    if (found != null) {

                        System.out.println("Task Found: " + found);

                    } else {

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

                    }

                    break;

                case 3:

                    tms.displayTasks();

                    break;

                case 4:

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

                    int deleteId = sc.nextInt();

                    boolean deleted = tms.deleteTask(deleteId);

                    if (deleted) {

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

                    } else {

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

                    }

                    break;

                case 5:

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

                    sc.close();

                    return;

                default:

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

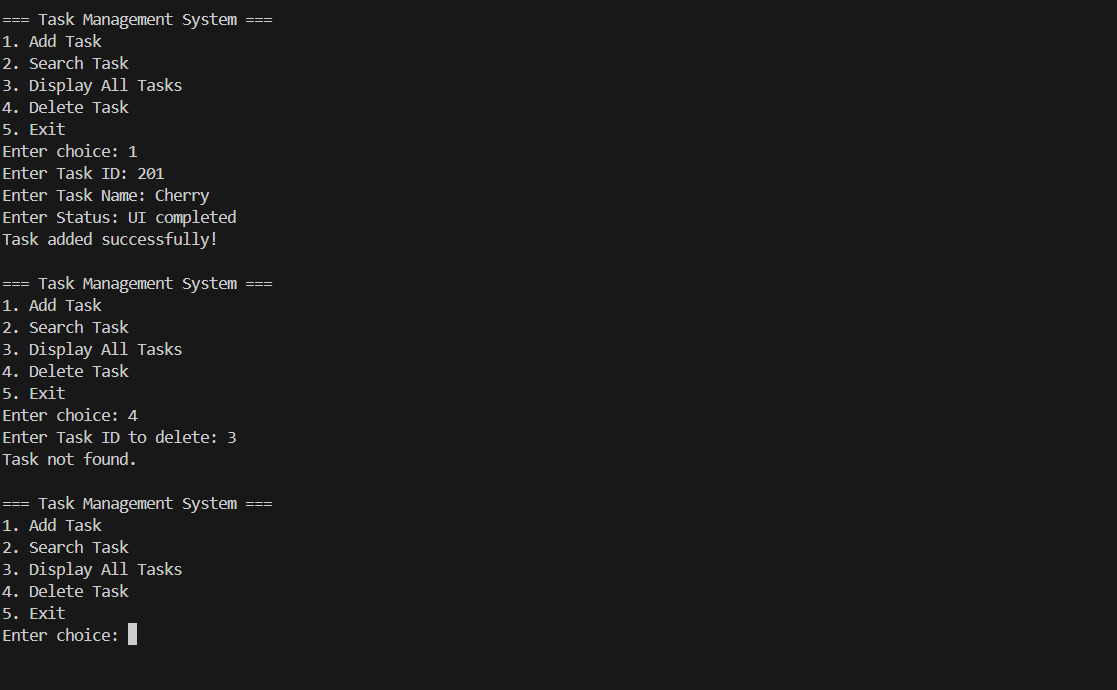
            }

        }

    }

}

**Output:**

****

**Exercise 6: Library Management System**

**Code:**

import java.util.Arrays;

import java.util.Scanner;

class Book {

    int bookId;

    String title;

    String author;

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

        this.bookId = id;

        this.title = title.toLowerCase(); // normalize case

        this.author = author;

    }

    @Override

    public String toString() {

        return "[" + bookId + "] " + title + " by " + author;

    }

}

public class LibraryManagementSystem {

    static Book[] books = new Book[100];

    static int count = 0;

    public static void addBook(Book book) {

        if (count < books.length) {

            books[count++] = book;

            System.out.println("Book added successfully!");

        } else {

            System.out.println("Library is full.");

        }

    }

    public static Book linearSearch(String title) {

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

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

                return books[i];

            }

        }

        return null;

    }

    public static Book binarySearch(String title) {

        Arrays.sort(books, 0, count, (a, b) -> a.title.compareTo(b.title));

        int left = 0, right = count - 1;

        title = title.toLowerCase();

        while (left <= right) {

            int mid = (left + right) / 2;

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

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

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

            else right = mid - 1;

        }

        return null;

    }

    public static void displayBooks() {

        if (count == 0) {

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

            return;

        }

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

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

        }

    }

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        while (true) {

            System.out.println("\n=== Library Management System ===");

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

            System.out.println("2. Search by Title (Linear Search)");

            System.out.println("3. Search by Title (Binary Search)");

            System.out.println("4. Display All Books");

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

            System.out.print("Choose an option: ");

            int choice = sc.nextInt();

            sc.nextLine();

            switch (choice) {

                case 1:

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

                    int id = sc.nextInt();

                    sc.nextLine();

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

                    String title = sc.nextLine();

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

                    String author = sc.nextLine();

                    addBook(new Book(id, title, author));

                    break;

                case 2:

                    System.out.print("Enter Title to search (Linear): ");

                    String lt = sc.nextLine();

                    Book linearResult = linearSearch(lt);

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

                    break;

                case 3:

                    System.out.print("Enter Title to search (Binary): ");

                    String bt = sc.nextLine();

                    Book binaryResult = binarySearch(bt);

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

                    break;

                case 4:

                    displayBooks();

                    break;

                case 5:

                    System.out.println("Goodbye!");

                    sc.close();

                    return;

                default:

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

            }

        }

    }

}

**Output:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Exercise 7: Financial Forecasting**

**Code:**

import java.util.Scanner;

public class FinancialForecasting {

    public static double forecastValue(double currentValue, double growthRate, int years) {

        if (years == 0) {

            return currentValue;  // base case

        }

        return forecastValue(currentValue, growthRate, years - 1) \* (1 + growthRate);

    }

    public static double forecastValueMemo(double currentValue, double growthRate, int years, double[] memo) {

        if (years == 0) return currentValue;

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

        memo[years] = forecastValueMemo(currentValue, growthRate, years - 1, memo) \* (1 + growthRate);

        return memo[years];

    }

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.println("=== Financial Forecasting Tool ===");

        System.out.print("Enter current value (e.g., initial investment): ");

        double currentValue = sc.nextDouble();

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

        double growthRate = sc.nextDouble();

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

        int years = sc.nextInt();

        double result = forecastValue(currentValue, growthRate, years);

        System.out.printf("Forecasted value after %d years: %.2f\n", years, result);

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

        double memoResult = forecastValueMemo(currentValue, growthRate, years, memo);

        System.out.printf("Forecasted value using memoization: %.2f\n", memoResult);

    }

}

**Output:**

**A computer screen with white text

AI-generated content may be incorrect.**