**Exercise 1: Inventory Management System**

Inventory.java:

import java.util.HashMap;

public class Inventory {

    private HashMap<Integer, Product> products;

    public Inventory() {

        products = new HashMap<>();

    }

    public void addProduct(Product product) {

        if (products.containsKey(product.getProductId())) {

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

        } else {

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

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

        }

    }

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

        Product product = products.get(productId);

        if (product != null) {

            product.setProductName(name);

            product.setQuantity(quantity);

            product.setPrice(price);

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

        } else {

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

        }

    }

    public void deleteProduct(int productId) {

        if (products.remove(productId) != null) {

            System.out.println("Product removed successfully.");

        } else {

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

        }

    }

    public void printInventory() {

        for (Product product : products.values()) {

            System.out.println(product);

        }

    }

}

Product.java:

public 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 String getProductName() { return productName; }

    public int getQuantity() { return quantity; }

    public double getPrice() { return price; }

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

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

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

   @Override

public String toString() {

        return "Product{" + "ID=" + productId + ", Name='" + productName + '\'' +

               ", Quantity=" + quantity + ", Price=" + price + '}';

    } }

Main.java:

public class Main {

    public static void main(String[] args) {

        Inventory inventory = new Inventory();

        Product p1 = new Product(101, "Mouse", 50, 299.99);

        Product p2 = new Product(102, "Keyboard", 30, 499.99);

        Product p3 = new Product(103, "Monitor", 20, 9999.99);

        inventory.addProduct(p1);

        inventory.addProduct(p2);

        inventory.addProduct(p3);

        inventory.printInventory();

        inventory.updateProduct(102, "Mechanical Keyboard", 40, 799.99);

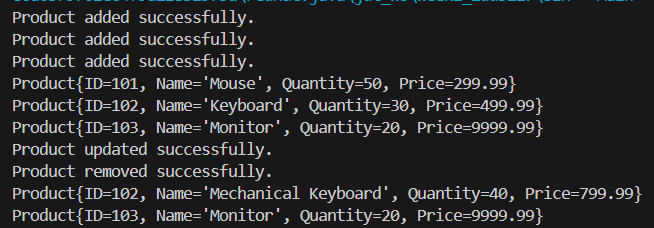
        inventory.deleteProduct(101);

      inventory.printInventory();

    }

}

OUTPUT:



**Exercise 2: E-commerce Platform Search Function(Mandatory)**

Product.java:

package Mandatory;

class Product {

    int productId;

    String productName;

    String category;

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

        this.productId = productId;

        this.productName = productName;

        this.category = category;

    }

    @Override

    public String toString() {

        return "Product [ID=" + productId + ", Name=" + productName + ", Category=" + category + "]";

    }

}

ProductSearch.java:

package Mandatory;

public class ProductSearch {

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

        for (Product product : products) {

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

                return product;

            }

        }

        return null;

    }

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

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

        while (left <= right) {

            int mid = left + (right - left) / 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;

    }

}

Main.java:

package Mandatory;

import java.util.Arrays;

public class Main {

    public static void main(String[] args) {

        Product[] products = {

            new Product(1, "Shoes", "Footwear"),

            new Product(2, "Laptop", "Electronics"),

            new Product(3, "Watch", "Accessories"),

            new Product(4, "Mobile", "Electronics")

        };

        Arrays.sort(products, (a, b) -> a.productName.compareToIgnoreCase(b.productName));

        Product foundLinear = ProductSearch.linearSearch(products, "Watch");

        System.out.println("Linear Search Result: " + foundLinear);

        Product foundBinary = ProductSearch.binarySearch(products, "Watch");

        System.out.println("Binary Search Result: " + foundBinary);

    }}

Output:



**Exercise 3: Sorting Customer Orders**

Order.java:

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;

    }

    @Override

    public String toString() {

        return "Order{" + "ID=" + orderId + ", Name='" + customerName + '\'' + ", Price=" + totalPrice + '}';

    }

}

SortOrders.java:

public class SortOrders {

    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) {

                    // Swap

                    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 pi = partition(orders, low, high);

            quickSort(orders, low, pi - 1);

            quickSort(orders, pi + 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;

    }

}

Main.java:

package exercise1;

public class Main {

    public static void main(String[] args) {

        Order[] orders = {

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

            new Order(102, "Bob", 150.0),

            new Order(103, "Charlie", 300.0)

        };

        SortOrders.bubbleSort(orders);

        for (Order o : orders) {

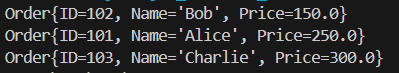
            System.out.println(o);

        }

    }

}

Output:



**Exercise 4: Employee Management System**

Employee.java:

public 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;

    }

    @Override

    public String toString() {

        return "Employee{" +

               "ID=" + employeeId +

               ", Name='" + name + '\'' +

               ", Position='" + position + '\'' +

               ", Salary=" + salary + '}';

    }

}

EmployeeManagement.java:

public class EmployeeManagement {

    private Employee[] employees;

    private int count;

    public EmployeeManagement(int size) {

        employees = new Employee[size];

        count = 0;

    }

    public void addEmployee(Employee e) {

        if (count < employees.length) {

            employees[count++] = e;

        } else {

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

        }

    }

    public Employee searchEmployee(int id) {

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

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

                return employees[i];

            }

        }

        return null;

    }

    public void displayEmployees() {

        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].employeeId == id) {

                // Shift elements left

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

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

                }

                employees[--count] = null;

                return true;

            }

        }

        return false;

    }

}

Main.java:

public class Main {

    public static void main(String[] args) {

        EmployeeManagement system = new EmployeeManagement(5);

        system.addEmployee(new Employee(1, "Alice", "Manager", 80000));

        system.addEmployee(new Employee(2, "Bob", "Engineer", 60000));

        system.addEmployee(new Employee(3, "Charlie", "HR", 50000));

        System.out.println("All employees:");

        system.displayEmployees();

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

        Employee found = system.searchEmployee(2);

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

        System.out.println("\nDeleting employee ID 2:");

        boolean deleted = system.deleteEmployee(2);

        System.out.println(deleted ? "Deleted successfully" : "Delete failed");

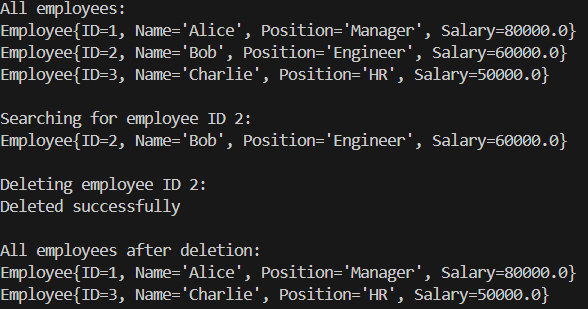
        System.out.println("\nAll employees after deletion:");

        system.displayEmployees();

    }

}

Output:



**Exercise 5: Task Management System**

Task.java:

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 "Task{" +

               "ID=" + taskId +

               ", Name='" + taskName + '\'' +

               ", Status='" + status + '\'' +

               '}';

    }

}

TaskNode.java:

class TaskNode {

    Task task;

    TaskNode next;

    public TaskNode(Task task) {

        this.task = task;

        this.next = null;

    }

}

TaskManager.java:

public class TaskManager {

    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;

        }

    }

    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() {

        TaskNode current = head;

        while (current != null) {

            System.out.println(current.task);

            current = current.next;

        }

    }

    public boolean deleteTask(int taskId) {

        if (head == null) return false;

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

            head = head.next;

            return true;

        }

        TaskNode current = head;

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

            current = current.next;

        }

        if (current.next == null) return false;

        current.next = current.next.next;

        return true;

    }

}

Main.java:

public class Main {

    public static void main(String[] args) {

        TaskManager manager = new TaskManager();

        manager.addTask(new Task(1, "Design UI", "Pending"));

        manager.addTask(new Task(2, "Develop Backend", "In Progress"));

        manager.addTask(new Task(3, "Testing", "Pending"));

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

        manager.displayTasks();

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

        Task found = manager.searchTask(2);

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

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

        boolean deleted = manager.deleteTask(2);

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

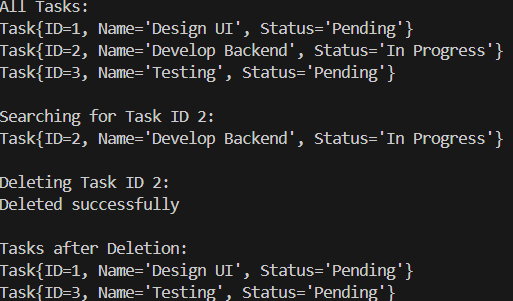
        System.out.println("\nTasks after Deletion:");

        manager.displayTasks();

    }

}

Output:



**Exercise 6: Library Management System**

Book.java:

public 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;

    }

    @Override

    public String toString() {

        return "Book{" +

               "ID=" + bookId +

               ", Title='" + title + '\'' +

               ", Author='" + author + '\'' +

               '}';

    }

}

Library.java:

import java.util.Arrays;

import java.util.Comparator;

public class Library {

    Book[] books;

    public Library(Book[] books) {

        this.books = books;

    }

    public Book linearSearchByTitle(String title) {

        for (Book book : books) {

            if (book.title.equalsIgnoreCase(title)) {

                return book;

            }

        }

        return null;

    }

    public Book binarySearchByTitle(String title) {

        int low = 0, high = books.length - 1;

        while (low <= high) {

            int mid = (low + high) / 2;

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

            if (compare == 0) {

                return books[mid];

            } else if (compare < 0) {

                low = mid + 1;

            } else {

                high = mid - 1;

            }

        }

        return null;

    }

    public void sortBooksByTitle() {

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

    }

}

Main.java:

public class Main {

    public static void main(String[] args) {

        Book[] books = {

            new Book(1, "The Alchemist", "Paulo Coelho"),

            new Book(2, "To Kill a Mockingbird", "Harper Lee"),

            new Book(3, "1984", "George Orwell")

        };

        Library library = new Library(books);

        System.out.println("Linear Search for '1984':");

        Book result1 = library.linearSearchByTitle("1984");

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

        library.sortBooksByTitle();

        System.out.println("\nBinary Search for '1984':");

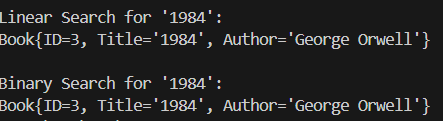
        Book result2 = library.binarySearchByTitle("1984");

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

    }

}

Output:



**Exercise 7: Financial Forecasting**

FinancialForecasting.java:

public class FinancialForecast {

    public static double predictFutureValue(double initialAmount, double rate, int years) {

        if (years == 0) {

            return initialAmount;

        }

        return predictFutureValue(initialAmount, rate, years - 1) \* (1 + rate);

    }

    public static double predictWithMemo(double initialAmount, double rate, int years, Double[] memo) {

        if (years == 0) {

            return initialAmount;

        }

        if (memo[years] != null) {

            return memo[years];

        }

        memo[years] = predictWithMemo(initialAmount, rate, years - 1, memo) \* (1 + rate);

        return memo[years];

    }

}

Main.java:

public class Main {

    public static void main(String[] args) {

        double initialAmount = 1000.0;

        double growthRate = 0.05;

        int years = 5;

        System.out.println("Recursive Forecast (without memoization):");

        double futureValue = FinancialForecast.predictFutureValue(initialAmount, growthRate, years);

        System.out.printf("Predicted Value after %d years: %.2f\n", years, futureValue);

        System.out.println("\nRecursive Forecast (with memoization):");

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

        double memoizedValue = FinancialForecast.predictWithMemo(initialAmount, growthRate, years, memo);

        System.out.printf("Predicted Value after %d years: %.2f\n", years, memoizedValue);

    }

}

Output:

