**1.E-commerce Platform Search Function**

**Main.java**

package search;

import java.util.Arrays;

import java.util.Comparator;

public class Main {

    public static void main(String[] args) {

        Product[] products = {

            new Product(3, "Laptop", "Electronics"),

            new Product(1, "Shirt", "Clothing"),

            new Product(5, "Book", "Education"),

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

            new Product(4, "Shoes", "Footwear")

        };

System.out.println("Linear Search for ID 4:");

        Product foundLinear = SearchAlgorithms.linearSearch(products, 4);

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

System.out.println("\nSorting products by productId for Binary Search...");

        Arrays.sort(products, Comparator.comparingInt(p -> p.productId));

System.out.println("Binary Search for ID 4:");

        Product foundBinary = SearchAlgorithms.binarySearch(products, 4);

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

    }}

**Product.java**

package search;

public class Product {

    public int productId;

    public String productName;

    public String category;

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

        this.productId = productId;

        this.productName = productName;

        this.category = category;

 }

public String toString() {

        return "[" + productId + "] " + productName + " - " + category;

    }}

**Searchalgorithm.java**

package search;

public class SearchAlgorithms {

public static Product linearSearch(Product[] products, int id) {

        for (Product p : products) {

            if (p.productId == id) {

                return p;

            } }

        return null;

    }

  public static Product binarySearch(Product[] products, int id) {

        int low = 0;

        int high = products.length - 1;

while (low <= high) {

            int mid = (low + high) / 2;

            if (products[mid].productId == id) {

                return products[mid];

            } else if (products[mid].productId < id) {

                low = mid + 1;

            } else {

                high = mid - 1;

            }

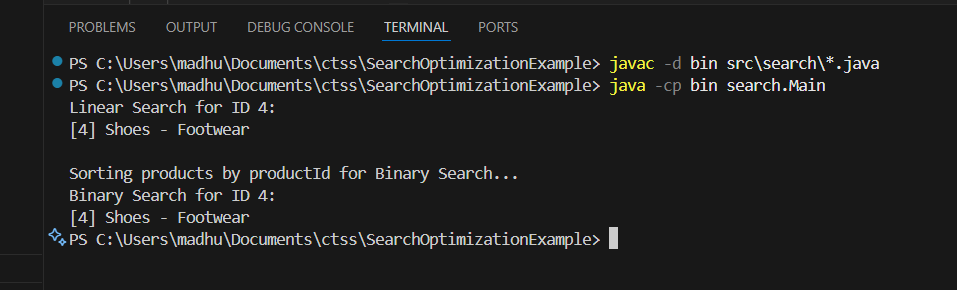
        }

        return null;

    }

}

**Output:**



**2.Financial Forecasting**

public class FinancialForecast {

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

        if (years == 0) {

            return currentValue;

        } else {

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

        }

    }

public static void main(String[] args) {

        double currentValue = 10000;

        double rate = 0.05;

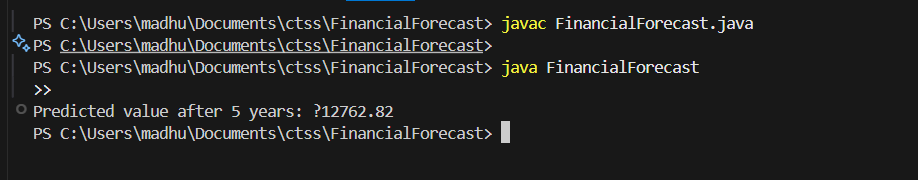
        int years = 5;

double result = forecast(years, currentValue, rate);

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

    }

}



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

**BinarySearch.java**

import java.util.Arrays;

import java.util.Comparator;

public class BinarySearch {

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

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

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

        while (low <= high) {

            int mid = (low + high) / 2;

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

            if (comparison == 0) {

                return products[mid];

            } else if (comparison < 0) {

                low = mid + 1;

            } else {

                high = mid - 1;

            }

        }

        return null;

    }

}

**LinearSearch.java**

public class LinearSearch {

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

        for (Product product : products) {

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

                return product;

            }

        }

        return null;

    }

}

**Main.java**

public class Main {

    public static void main(String[] args) {

        Product[] products = {

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

            new Product(102, "Shoes", "Fashion"),

            new Product(103, "Phone", "Electronics"),

            new Product(104, "Watch", "Accessories")

        };

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

        Product result1 = LinearSearch.search(products, "Phone");

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

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

        Product result2 = BinarySearch.search(products, "Phone");

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

    }

}

**Product.java**

public 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;

    }

    public String toString() {

        return productId + " - " + productName + " (" + category + ")";

    }

}

**SearchTest.java**

public class SearchTest {

    public static void main(String[] args) {

        Product[] products = {

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

            new Product(102, "Shoes", "Fashion"),

            new Product(103, "Phone", "Electronics"),

            new Product(104, "Watch", "Accessories")

        };

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

        Product result1 = LinearSearch.search(products, "Phone");

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

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

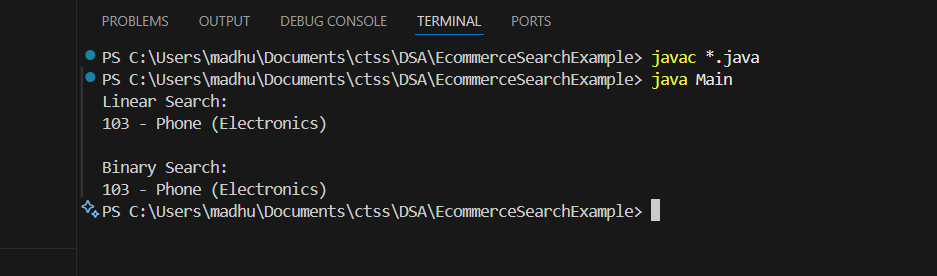
        Product result2 = BinarySearch.search(products, "Phone");

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

    }

}

**OUTPUT:**

****

**Exercise 3: Sorting Customer Orders**

**Bubblesort.java**

public class BubbleSort {

    public static void sort(Order[] 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].totalPrice > orders[j + 1].totalPrice) {

                    Order temp = orders[j];

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

                    orders[j + 1] = temp;

                }

            }

        }

    }

}

Main.java

public class Main {

    public static void main(String[] args) {

        Order[] orders = {

            new Order(1, "Alice", 500.0),

            new Order(2, "Bob", 150.0),

            new Order(3, "Charlie", 350.0),

            new Order(4, "David", 700.0),

            new Order(5, "Eve", 250.0)

        };

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

        for (Order order : orders) {

            System.out.println(order);

        }

        Order[] ordersForBubble = orders.clone();

        Order[] ordersForQuick = orders.clone();

        BubbleSort.sort(ordersForBubble);

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

        for (Order order : ordersForBubble) {

            System.out.println(order);

        }

        QuickSort.sort(ordersForQuick, 0, ordersForQuick.length - 1);

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

        for (Order order : ordersForQuick) {

            System.out.println(order);

        }

    }

}

**Order.java**

public 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 "Order{" +

            "orderId=" + orderId +

            ", customerName='" + customerName + '\'' +

            ", totalPrice=" + totalPrice +

            '}';

}

}

**Quicksort.java**

public class QuickSort {

    public static void sort(Order[] 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(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];

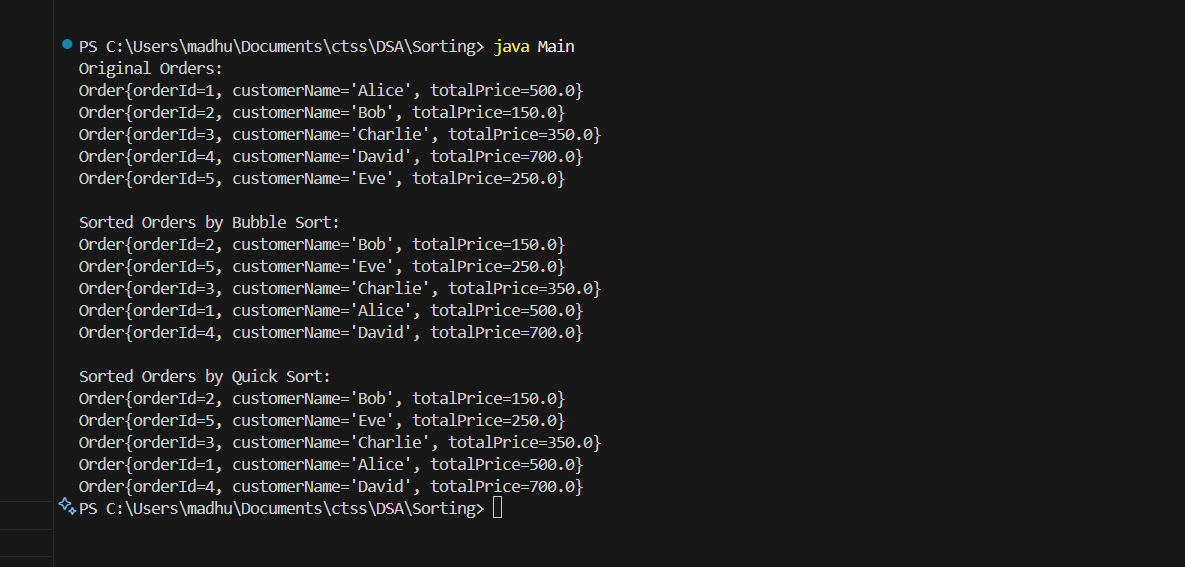
        orders[high] = temp;

return i + 1;

    }

}

**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;

    }

    public void display() {

        System.out.println("ID: " + employeeId + ", Name: " + name + ", Position: " + position + ", Salary: " + salary);

    }

}

**EmployeeManager.java**

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;

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

        } else {

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

        }

    }

    public void searchEmployee(int id) {

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

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

                employees[i].display();

                return;

            }

        }

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

    }

    public void traverseEmployees() {

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

            employees[i].display();

        }

    }

    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] = null;

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

                return;

            }

        }

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

    }

}

**Main.java**

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", "Designer", 65000));

        manager.addEmployee(new Employee(3, "Charlie", "Manager", 85000));

        System.out.println("\nAll Employees:");

        manager.traverseEmployees();

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

        manager.searchEmployee(2);

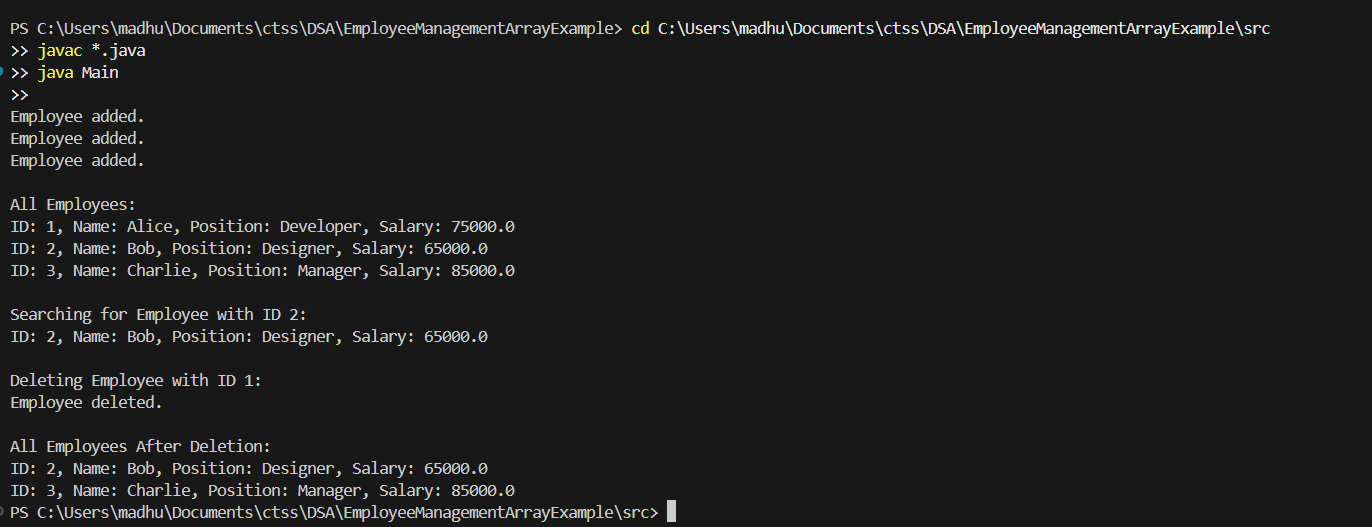
        System.out.println("\nDeleting Employee with ID 1:");

        manager.deleteEmployee(1);

        System.out.println("\nAll Employees After Deletion:");

        manager.traverseEmployees();

    }}

Output:

**Exercise 5: Task Management System**

**Main.java**

public class Main {

    public static void main(String[] args) {

        TaskManager manager = new TaskManager();

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

        manager.addTask(2, "Implement Backend", "In Progress");

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

System.out.println("\nAll Tasks:");

        manager.traverseTasks();

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

        manager.searchTask(2);

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

        manager.deleteTask(1);

System.out.println("\nAll Tasks After Deletion:");

        manager.traverseTasks();

    }

}

**Task.java**

public 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 void display() {

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

    }

}

TaskManager.java

public class TaskManager {

    private 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 current = head;

            while (current.next != null) {

                current = current.next;

            }

            current.next = newTask;

        }

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

    }

    public void searchTask(int id) {

        Task current = head;

        while (current != null) {

            if (current.taskId == id) {

                current.display();

                return;

            }

            current = current.next;

        }

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

    }

    public void traverseTasks() {

        Task current = head;

        while (current != null) {

            current.display();

            current = current.next;

        }

    } public void deleteTask(int id) {

        if (head == null) {

            System.out.println("No tasks to delete.");

            return;

        }

        if (head.taskId == id) {

            head = head.next;

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

            return;

        }Task current = head;

        while (current.next != null) {

            if (current.next.taskId == id) {

                current.next = current.next.next;

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

                return;

            }

            current = current.next;

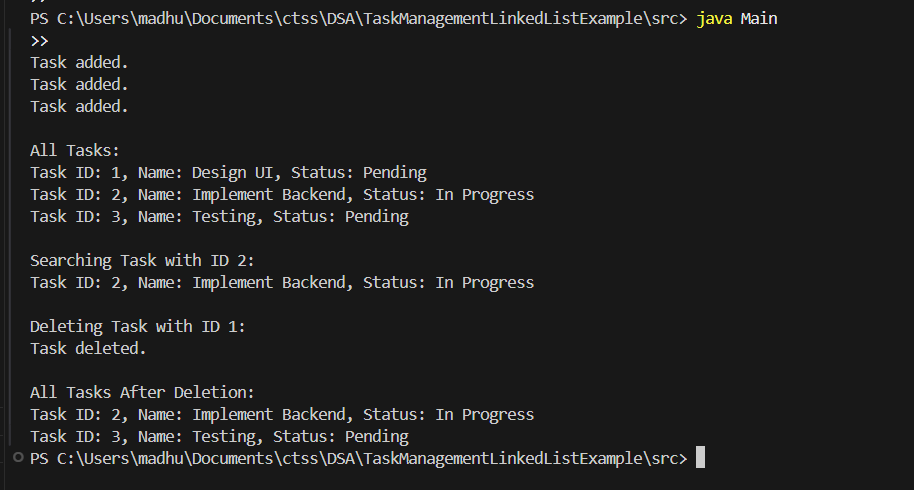
        }

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

    }

}

**Output:**

****

**Exercise 6: Library Management System**

**Book.java**

public class Book {

    private int bookId;

    private String title;

    private String author;

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

        this.bookId = bookId;

        this.title = title;

        this.author = author;

    }

    public String getTitle() {

        return title;

    }

    public String getAuthor() {

        return author;

    }

    public String toString() {

        return "Book{" +

                "ID=" + bookId +

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

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

                '}';

    }

}

**Library.java**

import java.util.Arrays;

import java.util.Comparator;

public class Library {

private Book[] books;

private int size;

public Library(int capacity) {

books = new Book[capacity];

size = 0;

}

public void addBook(Book book) {

if (size < books.length) {

books[size++] = book;

}

}

public Book linearSearchByTitle(String title) {

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

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

return books[i];

}

}

return null;

}

public Book binarySearchByTitle(String title) {

Arrays.sort(books, 0, size, Comparator.comparing(Book::getTitle));

int left = 0, right = size - 1;

while (left <= right) {

int mid = (left + right) / 2;

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

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

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

else right = mid - 1;

}

return null;

}

public void displayBooks() {

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

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

}

}

}

**Main.java**

public class Main {

    public static void main(String[] args) {

        Library library = new Library(10);

        library.addBook(new Book(1, "Java Basics", "John Doe"));

        library.addBook(new Book(2, "Data Structures", "Jane Smith"));

        library.addBook(new Book(3, "Algorithms", "Alan Turing"));

        library.addBook(new Book(4, "Python Essentials", "Guido Rossum"));

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

        library.displayBooks();

System.out.println("\nLinear Search for 'Algorithms':");

        Book foundLinear = library.linearSearchByTitle("Algorithms");

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

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

        Book foundBinary = library.binarySearchByTitle("Python Essentials");

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

    }

}

**Output:**

