**Exercise 1: Inventory Management System**

**InventorySystem.java**

import java.util.HashMap;

import java.util.Scanner;

public class InventorySystem {

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

    Scanner sc = new Scanner(System.in);

    // Add product

    public void addProduct() {

        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();

        Product p = new Product(id, name, qty, price);

        inventory.put(id, p);

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

    }

    // Update product

    public void updateProduct() {

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

        int id = sc.nextInt();

        if (inventory.containsKey(id)) {

            sc.nextLine();

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

            String name = sc.nextLine();

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

            int qty = sc.nextInt();

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

            double price = sc.nextDouble();

            Product p = new Product(id, name, qty, price);

            inventory.put(id, p);

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

        } else {

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

        }

    }

    // Delete product

    public void deleteProduct() {

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

        int id = sc.nextInt();

        if (inventory.containsKey(id)) {

            inventory.remove(id);

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

        } else {

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

        }

    }

    // Display inventory

    public void showInventory() {

        if (inventory.isEmpty()) {

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

        } else {

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

                System.out.println(p);

            }

        }

        System.out.println();

    }

    public static void main(String[] args) {

        InventorySystem system = new InventorySystem();

        Scanner sc = new Scanner(System.in);

        while (true) {

            System.out.println("1. Add Product\n2. Update Product\n3. Delete Product\n4. Show Inventory\n5. Exit");

            int choice = sc.nextInt();

            switch (choice) {

                case 1: system.addProduct(); break;

                case 2: system.updateProduct(); break;

                case 3: system.deleteProduct(); break;

                case 4: system.showInventory(); break;

                case 5: return;

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

            }

        }

    }

}

**Product.java**

public 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;

    }

    @Override

    public String toString() {

        return "ID: " + productId + ", Name: " + productName +

               ", Quantity: " + quantity + ", Price: ₹" + price;

    }

}

**Output:**

****

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

**Product.java**

public 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 "ID: " + productId + ", Name: " + productName + ", Category: " + category;

    }

}

**SearchFunctions.java**

import java.util.Arrays;

import java.util.Comparator;

public class SearchFunctions {

    // Linear Search

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

        for (Product p : products) {

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

                return p;

            }

        }

        return null;

    }

    // Sort products by name (required for binary search)

    public static void sortProductsByName(Product[] products) {

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

    }

    // Binary Search

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

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

        while (left <= right) {

            int mid = (left + right) / 2;

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

            if (compare == 0) {

                return products[mid];

            } else if (compare < 0) {

                right = mid - 1;

            } else {

                left = mid + 1;

            }

        }

        return null;

    }

}

**SearchDemo.java**

public class SearchDemo {

    public static void main(String[] args) {

        Product[] products = {

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

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

            new Product(103, "Shoes", "Fashion"),

            new Product(104, "T-Shirt", "Fashion"),

            new Product(105, "Monitor", "Electronics")

        };

        // LINEAR SEARCH

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

        Product result1 = SearchFunctions.linearSearch(products, "Shoes");

        if (result1 != null)

            System.out.println("Found: " + result1);

        else

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

        // SORT FOR BINARY SEARCH

        SearchFunctions.sortProductsByName(products);

        // BINARY SEARCH

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

        Product result2 = SearchFunctions.binarySearch(products, "Keyboard");

        if (result2 != null)

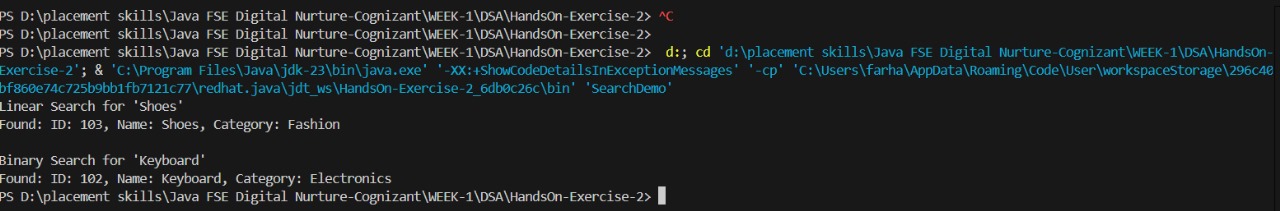
            System.out.println("Found: " + result2);

        else

            System.out.println("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) {

                    // Swap

                    Order temp = orders[j];

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

                    orders[j + 1] = temp;

                }

            }

        }

    }

}

**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;

    }

}

**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;

    }

    @Override

    public String toString() {

        return "Order ID: " + orderId + ", Customer: " + customerName + ", Total: Rs " + totalPrice;

    }

}

**SortDemo.java**

public class SortDemo {

    public static void printOrders(Order[] orders, String label) {

        System.out.println("\n" + label);

        for (Order o : orders) {

            System.out.println(o);

        }

    }

    public static void main(String[] args) {

        Order[] orders1 = {

            new Order(1, "Alice", 2500.00),

            new Order(2, "Bob", 499.99),

            new Order(3, "Charlie", 1750.50),

            new Order(4, "David", 3200.00)

        };

        // Clone for quick sort to avoid modifying original

        Order[] orders2 = orders1.clone();

        // Bubble Sort

        BubbleSort.sort(orders1);

        printOrders(orders1, "Sorted by Bubble Sort:");

        // Quick Sort

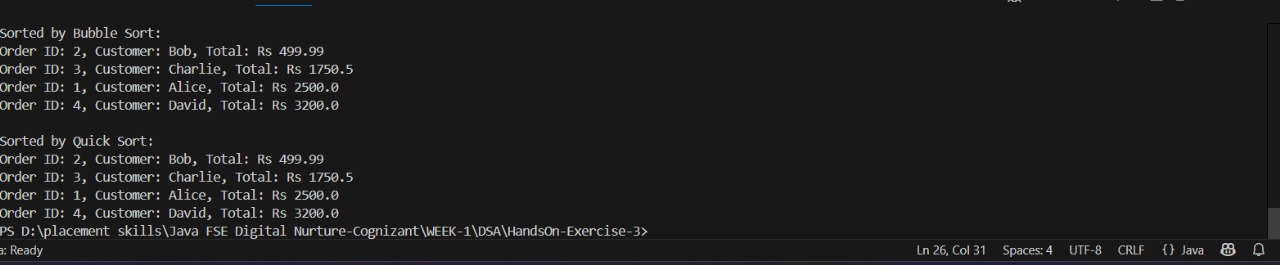
        QuickSort.sort(orders2, 0, orders2.length - 1);

        printOrders(orders2, "Sorted by Quick Sort:");

    }

}

**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 "ID: " + employeeId + ", Name: " + name +

               ", Position: " + position + ", Salary: Rs " + salary;

    }

}

**EmployeeSystem.java**

import java.util.Scanner;

public class EmployeeSystem {

    static final int MAX = 100;

    Employee[] employees = new Employee[MAX];

    int count = 0;

    Scanner sc = new Scanner(System.in);

    // Add employee

    public void addEmployee() {

        if (count >= MAX) {

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

            return;

        }

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

        int id = sc.nextInt();

        sc.nextLine();

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

        String name = sc.nextLine();

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

        String position = sc.nextLine();

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

        double salary = sc.nextDouble();

        employees[count++] = new Employee(id, name, position, salary);

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

    }

    // Search employee by ID

    public void searchEmployee() {

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

        int id = sc.nextInt();

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

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

                System.out.println("Found: " + employees[i]);

                return;

            }

        }

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

    }

    // Traverse all employees

    public 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]);

        }

    }

    // Delete employee by ID

    public void deleteEmployee() {

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

        int id = sc.nextInt();

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

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

                // Shift left

                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.");

    }

    public static void main(String[] args) {

        EmployeeSystem system = new EmployeeSystem();

        Scanner sc = new Scanner(System.in);

        while (true) {

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

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

            int choice = sc.nextInt();

            switch (choice) {

                case 1 -> system.addEmployee();

                case 2 -> system.searchEmployee();

                case 3 -> system.displayEmployees();

                case 4 -> system.deleteEmployee();

                case 5 -> {

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

                    return;

                }

                default -> System.out.println("Invalid choice.");

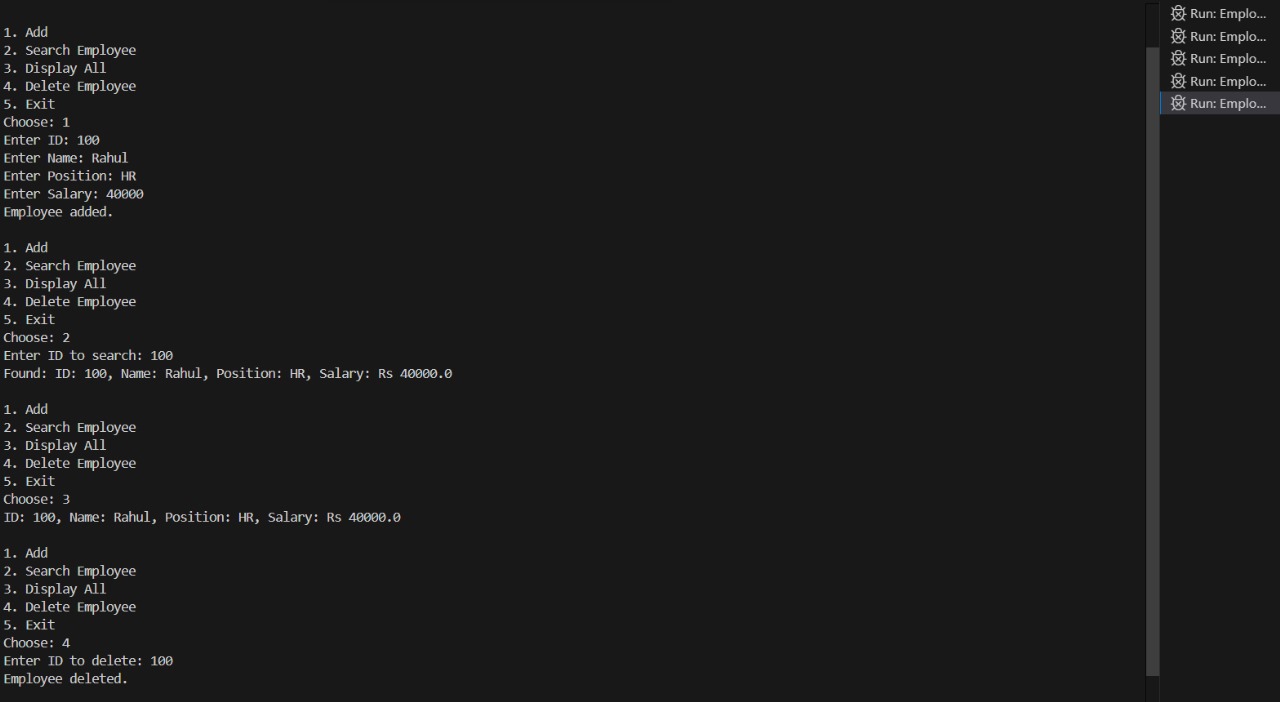
            }

        }

    }

}

**Output:**

****

**Exercise 5: Task Management System**

**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 String toString() {

        return "Task ID: " + taskId + ", Name: " + taskName + ", Status: " + status;

    }

}

**TaskManager.java**

import java.util.Scanner;

public class TaskManager {

    Task head = null;

    Scanner sc = new Scanner(System.in);

    // Add Task at end

    public void addTask() {

        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();

        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.");

    }

    // Search Task by ID

    public void searchTask() {

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

        int id = sc.nextInt();

        Task current = head;

        while (current != null) {

            if (current.taskId == id) {

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

                return;

            }

            current = current.next;

        }

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

    }

    // Traverse all tasks

    public void displayTasks() {

        if (head == null) {

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

        } else {

            Task current = head;

            while (current != null) {

                System.out.println(current);

                current = current.next;

            }

        }

    }

    // Delete Task by ID

    public void deleteTask() {

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

        int id = sc.nextInt();

        if (head == null) {

            System.out.println("Task list is empty.");

            return;

        }

        if (head.taskId == id) {

            head = head.next;

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

            return;

        }

        Task current = head;

        Task previous = null;

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

            previous = current;

            current = current.next;

        }

        if (current == null) {

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

        } else {

            previous.next = current.next;

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

        }

    }

    public static void main(String[] args) {

        TaskManager manager = new TaskManager();

        Scanner sc = new Scanner(System.in);

        int choice;

        while (true) {

            System.out.println("\n1. Add Task");

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

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

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

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

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

            choice = sc.nextInt();

            if (choice == 1) {

                manager.addTask();

            } else if (choice == 2) {

                manager.searchTask();

            } else if (choice == 3) {

                manager.displayTasks();

            } else if (choice == 4) {

                manager.deleteTask();

            } else if (choice == 5) {

                break;

            } else {

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

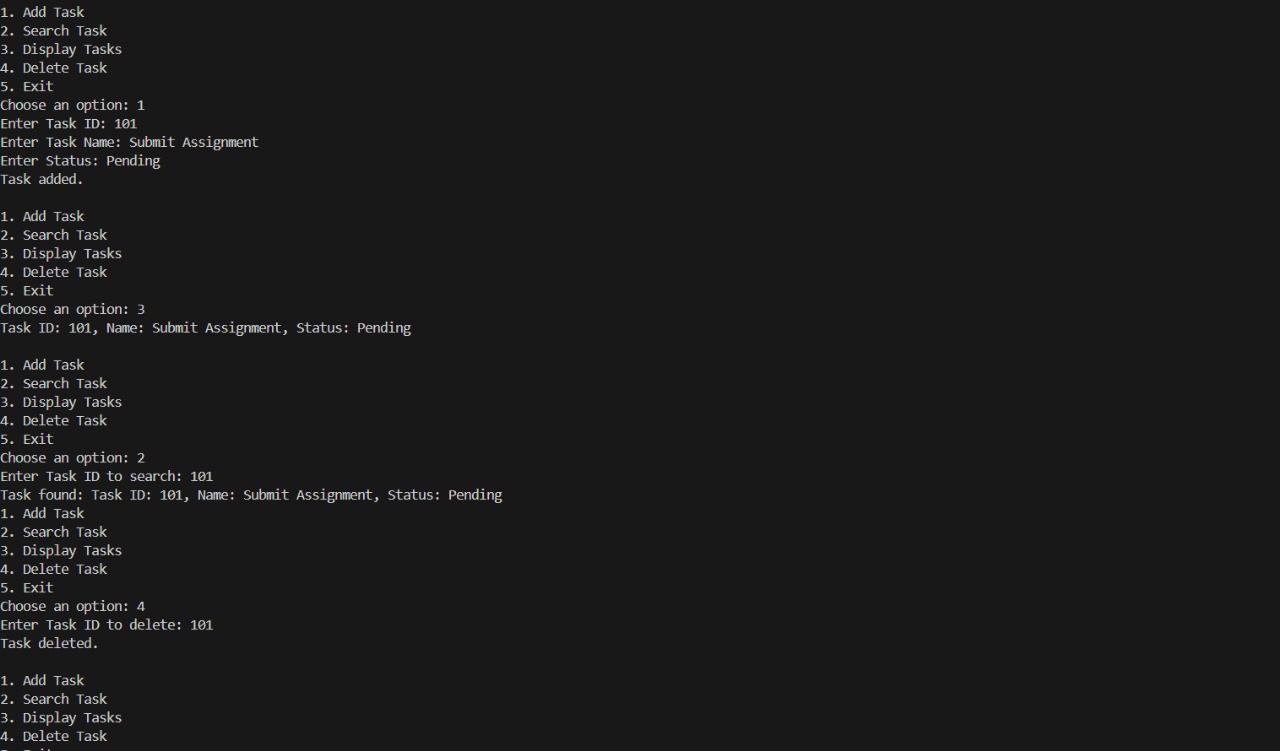
            }

        }

    }

}

**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;

    }

    public String toString() {

        return "ID: " + bookId + ", Title: " + title + ", Author: " + author;

    }

}

**LibrarySearch.java**

import java.util.Arrays;

import java.util.Comparator;

public class LibrarySearch {

    // Linear Search by Title

    public static Book linearSearch(Book[] books, String title) {

        for (Book b : books) {

            if (b.title.equalsIgnoreCase(title)) {

                return b;

            }

        }

        return null;

    }

    // Sort books by title

    public static void sortBooksByTitle(Book[] books) {

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

    }

    // Binary Search by Title (after sorting)

    public static Book binarySearch(Book[] books, String title) {

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

        while (left <= right) {

            int mid = (left + right) / 2;

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

            if (comparison == 0) {

                return books[mid];

            } else if (comparison < 0) {

                right = mid - 1;

            } else {

                left = mid + 1;

            }

        }

        return null;

    }

}

**LibraryDemo.java**

public class LibraryDemo {

    public static void main(String[] args) {

        Book[] books = {

            new Book(1, "Java Basics", "John Smith"),

            new Book(2, "C Programming", "Brian Kernighan"),

            new Book(3, "Python Essentials", "Guido van Rossum"),

            new Book(4, "Data Structures", "Narasimha Karumanchi"),

            new Book(5, "Operating Systems", "Abraham Silberschatz")

        };

        // Linear Search

        String target1 = "Python Essentials";

        Book result1 = LibrarySearch.linearSearch(books, target1);

        if (result1 != null) {

            System.out.println("Linear Search Result: " + result1);

        } else {

            System.out.println("Book not found (linear search).");

        }

        // Sort books before binary search

        LibrarySearch.sortBooksByTitle(books);

        // Binary Search

        String target2 = "Data Structures";

        Book result2 = LibrarySearch.binarySearch(books, target2);

        if (result2 != null) {

            System.out.println("Binary Search Result: " + result2);

        } else {

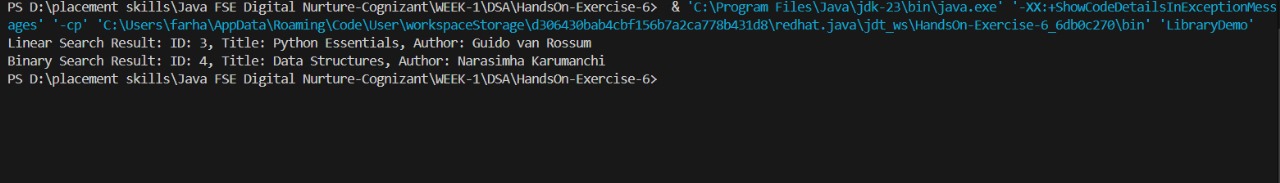
            System.out.println("Book not found (binary search).");

        }

    }

}

**Output:**



**Exercise 7: Financial Forecasting**

**FinancialForecast.java**

public class FinancialForecast {

    // Recursive method to predict future value

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

        if (years == 0) {

            return initialValue; // base case

        } else {

            // Recursive case: grow by rate

            double previousValue = forecastValue(initialValue, growthRate, years - 1);

            return previousValue + (previousValue \* growthRate);

        }

    }

    public static void main(String[] args) {

        double initialValue = 10000.0;  // Initial investment

        double growthRate = 0.08;       // 8% annual growth

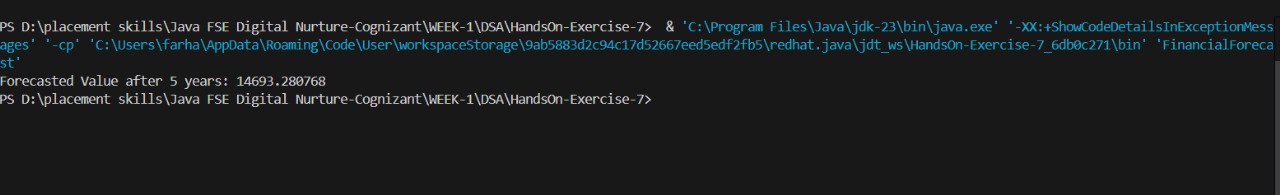
        int years = 5; double futureValue = forecastValue(initialValue, growthRate, years);

        System.out.println("Forecasted Value after " + years + " years: " + futureValue);

    }

}

**Output:**

****