**Exercise 1: Implementing the Singleton Pattern**

**CODE**

public class Logger {

    private static Logger instance;

    private Logger() {

        System.out.println("Logger instance created");

    }

    public static Logger getInstance() {

        if (instance == null) {

            instance = new Logger();

        }

        return instance;

    }

}

class Test {

    public static void main(String[] args) {

        Logger obj1 = Logger.getInstance();

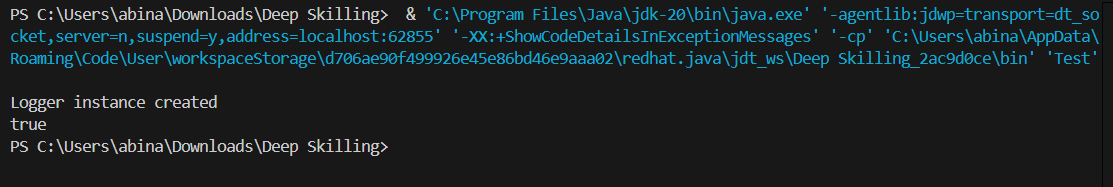
        Logger obj2 = Logger.getInstance();

        System.out.println(obj1 == obj2);

    }

}

**OUTPUT**

****

**Exercise 2: Implementing the Factory Method Pattern**

**CODE**

package FactoryPatternExample;

interface Document {

    void open();

}

class WordDocument implements Document {

    public void open() {

        System.out.println("Word Document is opened.");

    }

}

class PdfDocument implements Document {

    public void open() {

        System.out.println("PDF Document is opened.");

    }

}

class ExcelDocument implements Document {

    public void open() {

        System.out.println("Excel Document is opened.");

    }

}

abstract class DocumentFactory {

    public abstract Document createDocument();

}

class WordDocumentFactory extends DocumentFactory {

    public Document createDocument() {

        return new WordDocument();

    }

}

class PdfDocumentFactory extends DocumentFactory {

    public Document createDocument() {

        return new PdfDocument();

    }

}

class ExcelDocumentFactory extends DocumentFactory {

    public Document createDocument() {

        return new ExcelDocument();

    }

}

public class FactoryMethodExample {

    public static void main(String[] args) {

        DocumentFactory wordFactory = new WordDocumentFactory();

        Document wordDoc = wordFactory.createDocument();

        wordDoc.open();

        DocumentFactory pdfFactory = new PdfDocumentFactory();

        Document pdfDoc = pdfFactory.createDocument();

        pdfDoc.open();

        DocumentFactory excelFactory = new ExcelDocumentFactory();

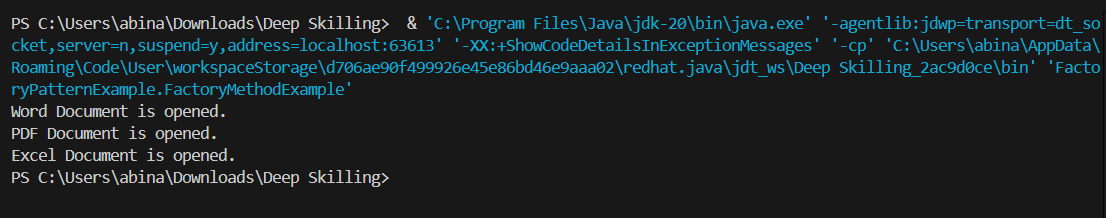
        Document excelDoc = excelFactory.createDocument();

        excelDoc.open();

    }

}

**OUTPUT**

****

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

**CODE**

import java.util.\*;

class Product {

    int productId;

    String productName;

    String category;

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

        this.productId = productId;

        this.productName = productName;

        this.category = category;

    }

    public String toString() {

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

    }}public class ProductSearch {

    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[] sortedProducts, String name) {

        int left = 0;

        int right = sortedProducts.length - 1;

        while (left <= right) {

            int mid = (left + right) / 2;

            int cmp = sortedProducts[mid].productName.compareToIgnoreCase(name);

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

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

            else right = mid - 1;

        }

        return null;

    }

    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, "Book", "Education")

        };

        Product[] sortedProducts = Arrays.copyOf(products, products.length);

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

        String searchTerm = "Phone";

        Product result1 = linearSearch(products, searchTerm);

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

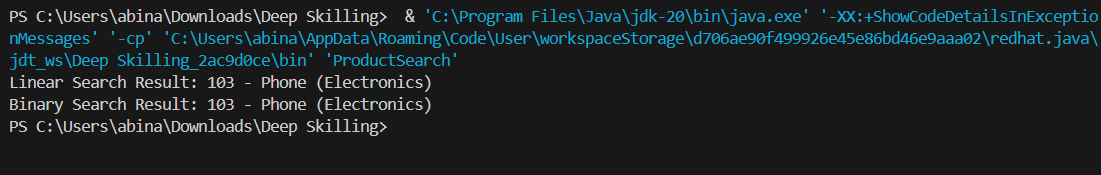
        Product result2 = binarySearch(sortedProducts, searchTerm);

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

    }

}

**OUTPUT**

****

**Exercise 7: Financial Forecasting**

**CODE**

package ForecastTool;

public class ForecastTool {

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

if (years == 0) return initial;

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

    }

    public static void main(String[] args) {

        double initialInvestment = 20000;

        double annualGrowthRate = 0.09;

        int futureYears = 3;

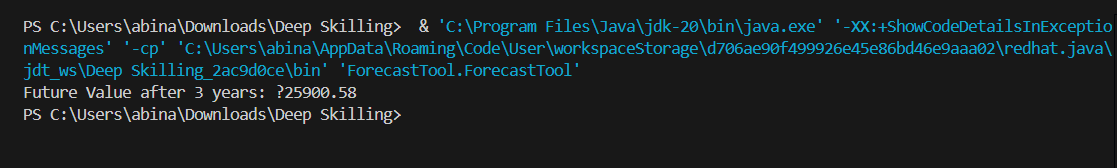
        double futureValue = forecast(futureYears, annualGrowthRate, initialInvestment);

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

    }

}

**OUTPUT**

****