**Exercise 1: Implementing the Singleton Pattern**

Logger.java

public class Logger {

    private static Logger instance;

    private Logger() {

        System.out.println("Logger Initialized");

    }

    public static Logger getInstance() {

        if (instance == null) {

            instance = new Logger();

        }

        return instance;

    }

    public void log(String message) {

        System.out.println("Log: " + message);

    }

}

SingletonTest.java

public class SingletonTest {

    public static void main(String[] args) {

        Logger logger1 = Logger.getInstance();

        logger1.log("First log");

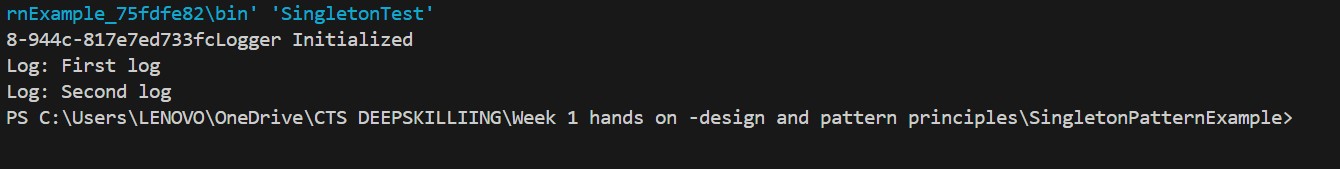
        Logger logger2 = Logger.getInstance();

        logger2.log("Second log");

    }

}

OUTPUT



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

Document.java

interface Document {

    void open();

}

WordDocument.java

class WordDocument implements Document {

    public void open() {

        System.out.println("Opening Word Document");

    }

}

PdfDocument.java

class PdfDocument implements Document {

    public void open() {

        System.out.println("Opening PDF Document");

    }

}

ExcelDocument.java

class ExcelDocument implements Document {

    public void open() {

        System.out.println("Opening Excel Document");

    }

}

DocumentFactory.java

abstract class DocumentFactory {

    abstract Document createDocument();

}

WordFactory.java

class WordFactory extends DocumentFactory {

    public Document createDocument() {

        return new WordDocument();

    }

}

PdfFactory.java

class PdfFactory extends DocumentFactory {

    public Document createDocument() {

        return new PdfDocument();

    }

}

ExcelFactory.java

class ExcelFactory extends DocumentFactory {

    public Document createDocument() {

        return new ExcelDocument();

    }

}

FactoryTest.java

public class FactoryTest {

    public static void main(String[] args) {

        DocumentFactory factory = new WordFactory();

        Document doc1 = factory.createDocument();

        doc1.open();

        factory = new PdfFactory();

        Document doc2 = factory.createDocument();

        doc2.open();

        factory = new ExcelFactory();

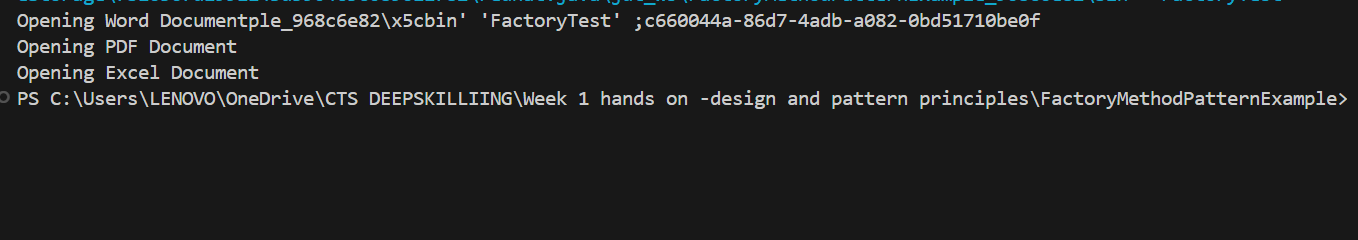
        Document doc3 = factory.createDocument();

        doc3.open();

    }

}

OUTPUT



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

SearchFunction.java

import java.util.\*;

class ProductSearch {

    int productId;

    String productName;

    String category;

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

        this.productId = productId;

        this.productName = productName;

        this.category = category;

    }

}

public class SearchFunction {

    static ProductSearch[] products = {

        new ProductSearch(1, "Laptop", "Electronics"),

        new ProductSearch(2, "Keyboard", "Electronics"),

        new ProductSearch(3, "Book", "Stationery")

    };

    public static int linearSearch(String name) {

        for (int i = 0; i < products.length; i++) {

            if (products[i].productName.equalsIgnoreCase(name)) return i;

        }

        return -1;

    }

    public static int binarySearch(String name) {

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

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

        while (low <= high) {

            int mid = (low + high) / 2;

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

            if (cmp == 0) return mid;

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

            else high = mid - 1;

        }

        return -1;

    }

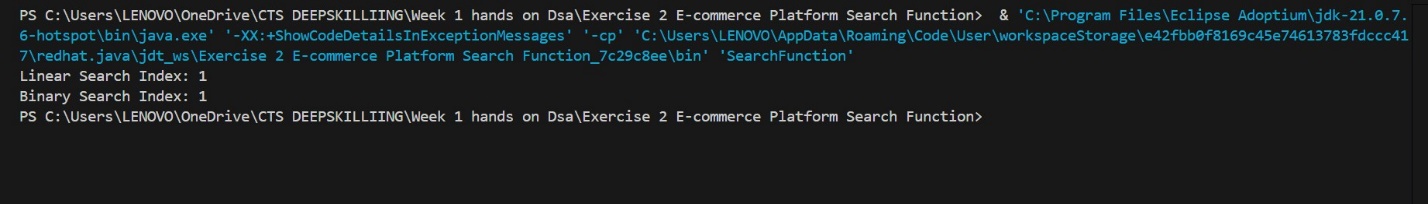
    public static void main(String[] args) {

        System.out.println("Linear Search Index: " + linearSearch("Keyboard"));

        System.out.println("Binary Search Index: " + binarySearch("Keyboard"));

    }

}

OUTPUT

**Exercise 7: Financial Forecasting**

FinancialForecast.java

public class FinancialForecast {

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

        if (years == 0) return currentValue;

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

    }

    public static void main(String[] args) {

        double futureValue = forecastValue(10000, 0.1, 5);

        System.out.println("Future Value: " + futureValue);

    }

}

OUTPUT

