**MODULE 1 - DESIGN PATTERNS & PRINICPLES**

**Exercise 1: Implementing the Singleton Pattern**

**Code:**

public class Logger {

private static Logger instance;

private Logger() {

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

}

public static Logger getInstance() {

if (instance == null) {

instance = new Logger();

}

return instance;

}

public void log(String msg) {

System.out.println("log: " + msg);

}

}

//Main

public class Main {

public static void main(String[] args) {

Logger logger1 = Logger.getInstance();

logger1.log("Start the app...");

Logger logger2 = Logger.getInstance();

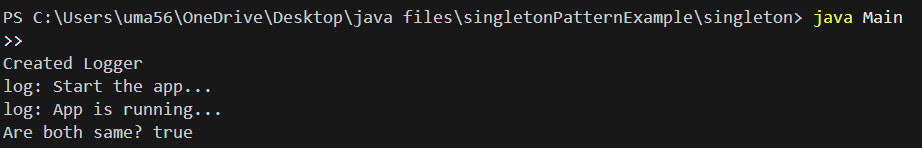
logger2.log("App is running...");

System.out.println("Are both same? " + (logger1 == logger2));

}

}

**Output:**

****

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

**Code**

interface document {

    void write();

}

// Concrete Classes

class Word implements document {

    public void write() {

        System.out.println("WordDocument is written!!");

    }

}

class Pdf implements document {

    public void write() {

        System.out.println("PDFDocument is written!!");

    }

}

class Excel implements document {

    public void write() {

        System.out.println("ExcelDocument is written!!");

    }

}

// Abstract Factory

abstract class DocumentFactory {

    public abstract document createDocument();

}

// Concrete Factories

class WordDocFactory extends DocumentFactory {

    public document createDocument() {

        return new Word();

    }

}

class PdfDocFactory extends DocumentFactory {

    public document createDocument() {

        return new Pdf();

    }

}

class ExcelDocFactory extends DocumentFactory {

    public document createDocument() {

        return new Excel();

    }

}

public class Main1 {

    public static void main(String[] args) {

        DocumentFactory wordFactory = new WordDocFactory();

        document wordDoc = wordFactory.createDocument();

        wordDoc.write();

        DocumentFactory pdfFactory = new PdfDocFactory();

        document pdfDoc = pdfFactory.createDocument();

        pdfDoc.write();

        DocumentFactory excelFactory = new ExcelDocFactory();

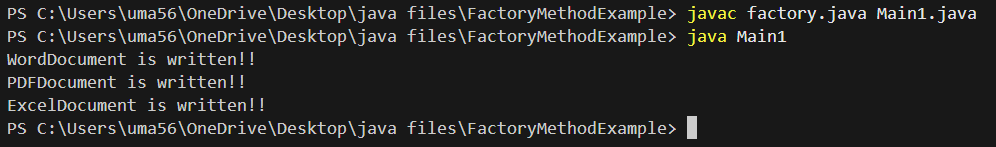
        document excelDoc = excelFactory.createDocument();

        excelDoc.write();

    }

}

**Output:**

****

**DATA STRUCTURES & ALGORITHMS**

**Exercise 1: E-commerce Platform Search Function**

Most efficient method is binary search because of its Time Complexity O(log n) sorted array.

**Code:**

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;

    }

    @Override

    public String toString() {

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

    }

    public String getProductName() {

        return productName.toLowerCase();

    }

}

import java.util.Arrays;

import java.util.Comparator;

public class BinarySearchDemo {

    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, "T-Shirt", "Apparel")

        };

        Arrays.sort(products, Comparator.comparing(Product::getProductName));

        Product found = binarySearch(products, "Watch");

if (found != null) {

            System.out.println("Product Found: " + found);

        } else {

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

        }

    }

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

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

        searchName = searchName.toLowerCase();

        while (left <= right) {

            int mid = (left + right) / 2;

            String midName = products[mid].getProductName();

            if (midName.equals(searchName)) {

                return products[mid];

            } else if (midName.compareTo(searchName) < 0) {

                left = mid + 1;

            } else {

                right = mid - 1;

            }

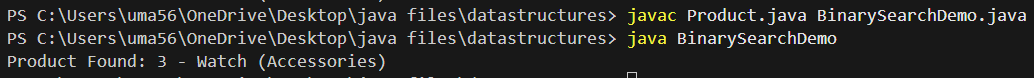
        }

        return null;

    }

}

**Output:**



**Exercise 2: Financial Forecasting**

**Code**

public class FinancialForecast {

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

    for (int i = 0; i < years; i++) {

        currentValue \*= (1 + growthRate);

    }

    return currentValue;

}

    public static void main(String[] args) {

        double presentValue = 10000;

        double annualGrowthRate = 0.08;

        int forecastYears = 5;

        double futureValue = calculateFutureValueIterative(presentValue, annualGrowthRate, forecastYears);

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

    }

}

**Output**

