**WEEK 1**

**DESIGN PATTERN AND PRINCIPLES/DATA STRUCTURES AND ALGORITHMS**

**DESIGN PATTERN AND PRINCIPLES**

**Exercise 1: Implementing the Singleton Pattern**

**Scenario:**

You need to ensure that a logging utility class in your application has only one instance throughout the application lifecycle to ensure consistent logging.

**Code:**

public class SingletonPatternExample {

static 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);

}

}

public static void main(String[] args) {

Logger logger1 = Logger.getInstance();

Logger logger2 = Logger.getInstance();

if (logger1 == logger2) {

System.out.println("Both logger instances are the same.");

} else {

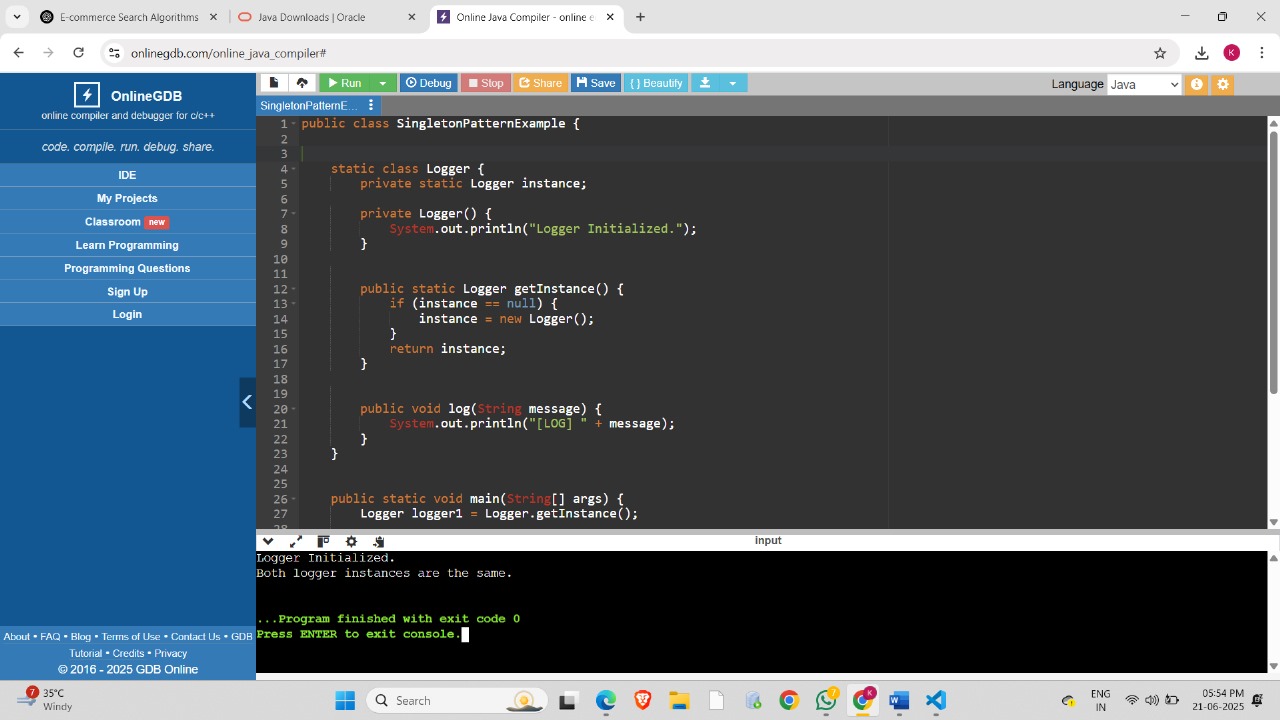
System.out.println("Different logger instances exist.");

}

}

}

**OUTPUT :**



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

**Scenario:**

You are developing a document management system that needs to create different types of documents (e.g., Word, PDF, Excel). Use the Factory Method Pattern to achieve this.

**Code:**

import java.util.Scanner;

public class FactoryMethodPatternExample {

interface Document {

void open();

}

static class WordDocument implements Document {

public void open() {

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

}

}

static class PdfDocument implements Document {

public void open() {

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

}

}

static class ExcelDocument implements Document {

public void open() {

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

}

}

static abstract class DocumentFactory {

public abstract Document createDocument();

}

static class WordFactory extends DocumentFactory {

public Document createDocument() {

return new WordDocument();

}

}

static class PdfFactory extends DocumentFactory {

public Document createDocument() {

return new PdfDocument();

}

}

static class ExcelFactory extends DocumentFactory {

public Document createDocument() {

return new ExcelDocument()

}

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter document type (word/pdf/excel): ");

String type = scanner.nextLine().toLowerCase();

DocumentFactory factory = null;

switch (type) {

case "word":

factory = new WordFactory();

break;

case "pdf":

factory = new PdfFactory();

break;

case "excel":

factory = new ExcelFactory();

break;

default:

System.out.println("Invalid document type.");

System.exit(0);

}

Document doc = factory.createDocument();

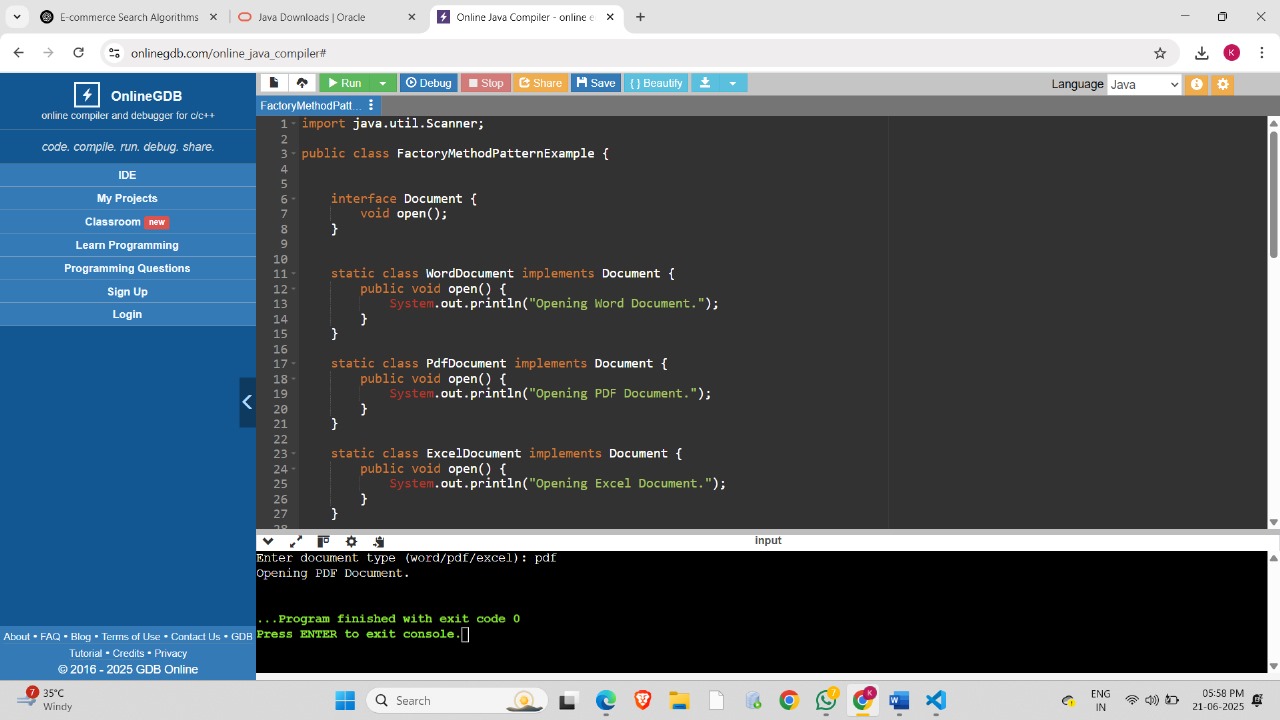
doc.open();

scanner.close();

}

}

**OUTPUT :**



**DATA STRUCTURES AND ALGORITHMS**

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

**Scenario:**

You are working on the search functionality of an e-commerce platform. The search needs to be optimized for fast performance.

**Code:**

import java.util.\*;

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 class ECommerceSearch {

public static Product linearSearch(List<Product> products, String targetName) {

for (Product product : products) {

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

return product;

}

}

return null;

}

public static Product binarySearch(List<Product> products, String targetName) {

int left = 0, right = products.size() - 1;

targetName = targetName.toLowerCase();

while (left <= right) {

int mid = (left + right) / 2;

String midName = products.get(mid).productName.toLowerCase();

if (midName.equals(targetName)) {

return products.get(mid);

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

left = mid + 1;

} else {

right = mid - 1;

}

}

return null;

}

public static void main(String[] args) {

List<Product> products = new ArrayList<>();

products.add(new Product(1, "Laptop", "Electronics"));

products.add(new Product(2, "Phone", "Electronics"));

products.add(new Product(3, "T-Shirt", "Clothing"));

products.add(new Product(4, "Book", "Stationery"));

products.add(new Product(5, "Shoes", "Footwear"));

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

String searchName = "Phone";

Product result1 = linearSearch(products, searchName);

System.out.println("Searching for '" + searchName + "': " + result1);

System.out.println("\n=== Binary Search ===");

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

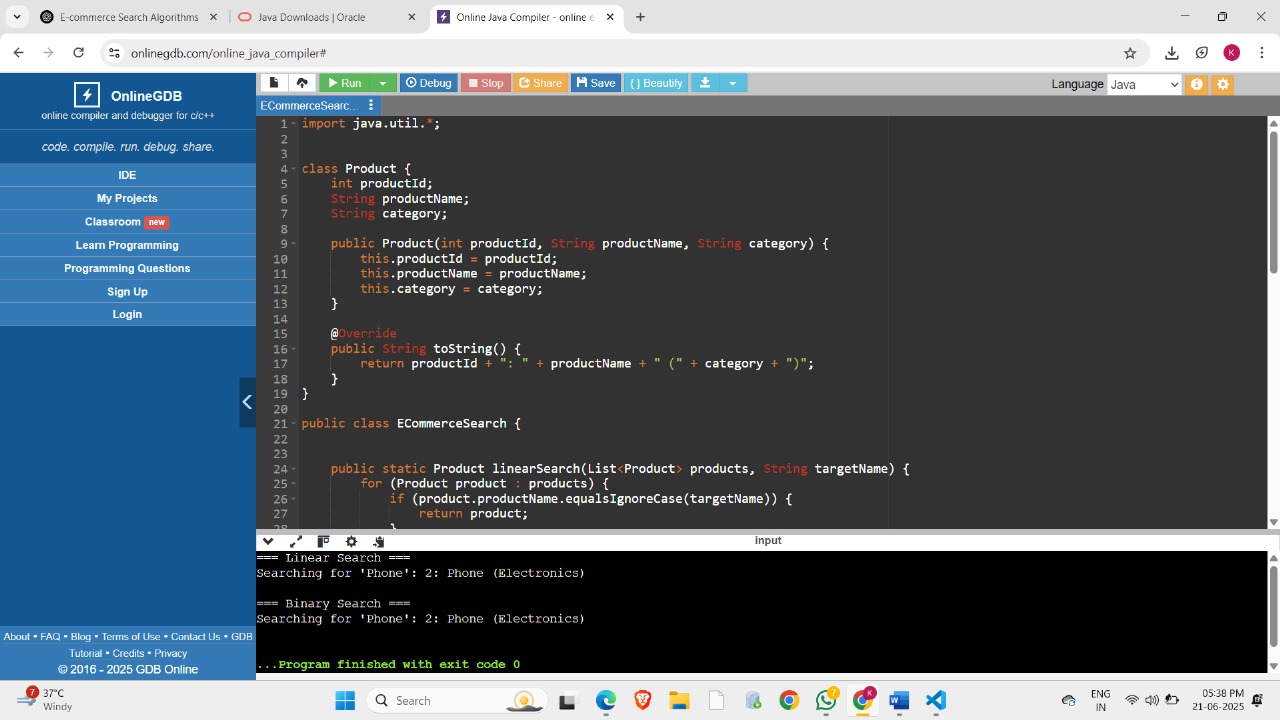
Product result2 = binarySearch(products, searchName);

System.out.println("Searching for '" + searchName + "': " + result2);

}

}

**OUTPUT :**



**Exercise 7: Financial Forecasting**

**Scenario:**

You are developing a financial forecasting tool that predicts future values based on past data.

**Code:**

import java.util.\*;

public class FinancialForecast {

public static double forecastSales(double lastYearSales, double growthRate, int years) {

if (years == 0) {

return lastYearSales;

} else {

return forecastSales(lastYearSales, growthRate, years - 1) \* (1 + growthRate);

}

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter number of past years of sales data: ");

int n = scanner.nextInt();

double[] sales = new double[n];

double sum = 0;

System.out.println("Enter sales for each year:");

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

System.out.print("Year " + (i + 1) + ": ₹");

sales[i] = scanner.nextDouble();

sum += sales[i];

}

double averageSales = sum / n;

System.out.printf("\nAverage sales over %d years: ₹%.2f\n", n, averageSales);

double growthRate = 0;

for (int i = 1; i < n; i++) {

growthRate += (sales[i] - sales[i - 1]) / sales[i - 1];

}

growthRate /= (n - 1);

System.out.printf("Estimated average growth rate: %.2f%%\n", growthRate \* 100);

System.out.print("Enter number of future years to forecast: ");

int futureYears = scanner.nextInt();

double forecast = forecastSales(sales[n - 1], growthRate, futureYears);

System.out.printf("Forecasted sales after %d years: ₹%.2f\n", futureYears, forecast);

scanner.close();

}

}

**OUTPUT :**

