**DESIGN PATTERNS AND PRINCIPLES**

**Exercise 1:** Implementing Singleton Pattern

**Source Code:**

import java.util.Scanner;

import java.net.MalformedURLException;

import java.net.URL;

import java.time.LocalDateTime;

import java.time.format.DateTimeFormatter;

enum Logger {

INSTANCE;

public void log(String message) {

String timestamp = LocalDateTime.now().format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss"));

System.out.println("[LOG " + timestamp + "] " + message);

}

}

enum ConfigManager {

INSTANCE;

private String environment;

private String dbUrl;

public String getEnvironment() {

return environment;

}

public void setEnvironment(String environment) {

this.environment = environment;

}

public String getDbUrl() {

return dbUrl;

}

public void setDbUrl(String dbUrl) {

this.dbUrl = dbUrl;

}

public void printConfig() {

System.out.println("Environment: " + environment);

System.out.println("Database URL: " + dbUrl);

}

public boolean isValidUrl(String url) {

try {

new URL(url);

return true;

} catch (MalformedURLException e) {

return false;

}

}

}

enum AppThemeManager {

INSTANCE;

private String theme = "light";

public String getTheme() {

return theme;

}

public void setTheme(String theme) {

if (theme.equalsIgnoreCase("light") || theme.equalsIgnoreCase("dark")) {

this.theme = theme.toLowerCase();

} else {

System.out.println("⚠️ Invalid theme. Use 'light' or 'dark'.");

}

}

public void applyTheme() {

System.out.println("🎨 Applying " + theme + " theme...");

}

}

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int choice;

do {

System.out.println("\n===== Singleton Demo Menu =====");

System.out.println("1. Log a Message");

System.out.println("2. Set Configuration");

System.out.println("3. Set App Theme");

System.out.println("0. Exit");

System.out.print("Enter your choice: ");

choice = readInt(scanner);

switch (choice) {

case 1:

System.out.print("Enter log message: ");

String msg = scanner.nextLine();

Logger.INSTANCE.log(msg);

break;

case 2:

handleConfig(scanner);

break;

case 3:

handleTheme(scanner);

break;

case 0:

System.out.println("👋 Exiting...");

break;

default:

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

}

} while (choice != 0);

scanner.close();

}

private static void handleConfig(Scanner scanner) {

ConfigManager config = ConfigManager.INSTANCE;

System.out.print("Enter environment (dev/prod): ");

config.setEnvironment(scanner.nextLine());

while (true) {

System.out.print("Enter database URL: ");

String url = scanner.nextLine();

if (config.isValidUrl(url)) {

config.setDbUrl(url);

break;

} else {

System.out.println("❌ Invalid URL. Try again.");

}

}

System.out.println("\n📦 Configuration Saved:");

config.printConfig();

}

private static void handleTheme(Scanner scanner) {

AppThemeManager themeManager = AppThemeManager.INSTANCE;

System.out.print("Enter theme (light/dark): ");

themeManager.setTheme(scanner.nextLine());

themeManager.applyTheme();

System.out.println("Current theme: " + themeManager.getTheme());

}

private static int readInt(Scanner scanner) {

while (true) {

try {

String line = scanner.nextLine();

return Integer.parseInt(line);

} catch (NumberFormatException e) {

System.out.print("Please enter a valid number: ");

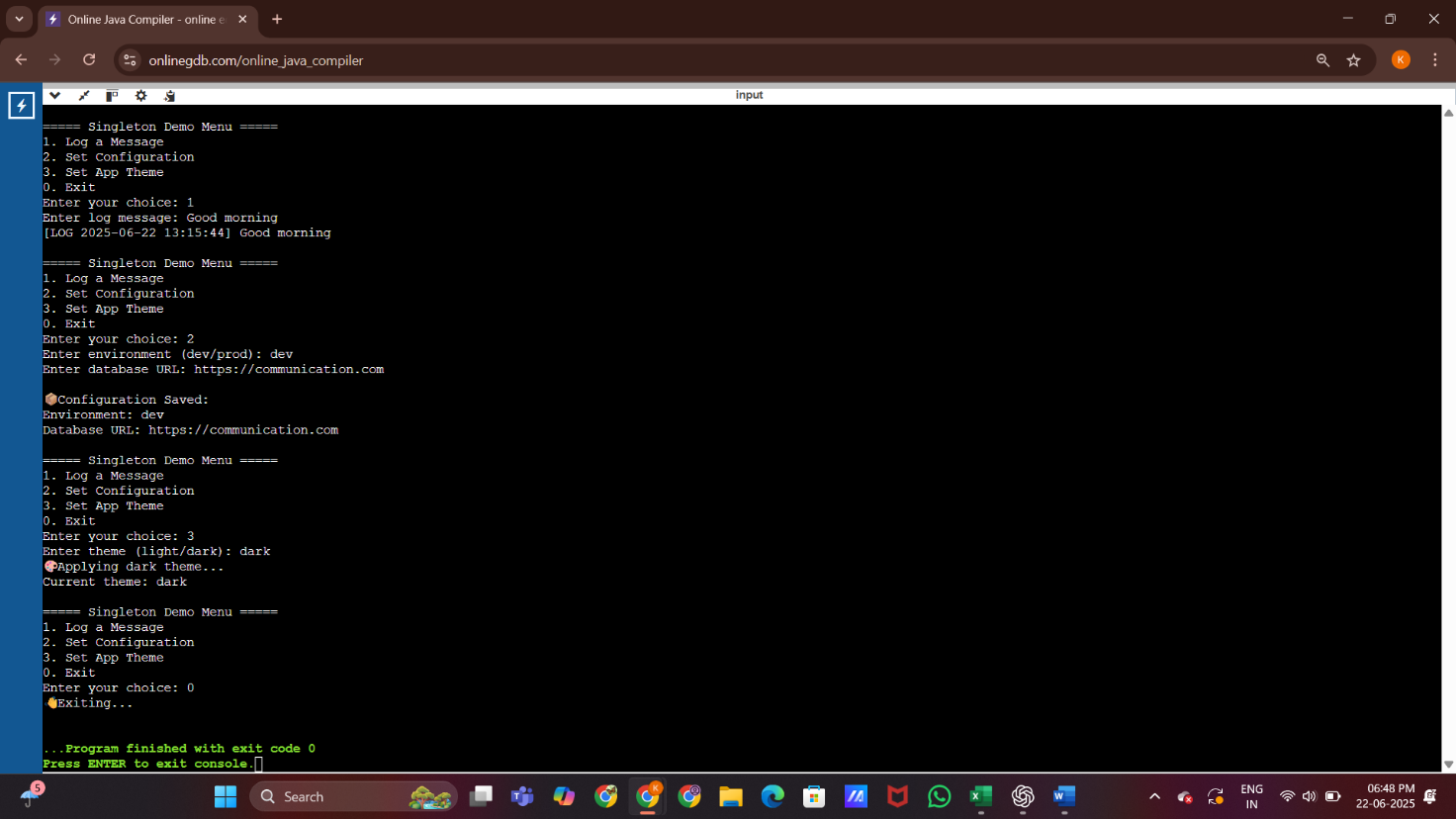
}

}

}

}

**Output:**



**Exercise 2:** Implementing factory Method pattern

**Source code :**

import java.util.Scanner;

interface Shape {

void draw();

}

class Circle implements Shape {

public void draw() {

System.out.println("\u26AB Drawing a Circle");

}

}

class Rectangle implements Shape {

public void draw() {

System.out.println("⬛ Drawing a Rectangle");

}

}

class Triangle implements Shape{

public void draw(){

System.out.println("\u25B2 Drawing a Triangle");

}

}

class Diamond implements Shape{

public void draw(){

System.out.println("🔷 Drawing a Diamond");

}

}

class Star implements Shape{

public void draw(){

System.out.println("⭐ Drawing a Star");

}

}

class ShapeFactory {

public static Shape getShape(String shapeType) {

if (shapeType == null) return null;

switch (shapeType.toLowerCase()) {

case "circle":

return new Circle();

case "rectangle":

return new Rectangle();

case "triangle":

return new Triangle();

case "diamond":

return new Diamond();

case "star":

return new Star();

default:

return null;

}

}

}

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Enter shape type :");

String input = scanner.nextLine();

Shape shape = ShapeFactory.getShape(input);

if (shape != null) {

shape.draw();

} else {

System.out.println("❌ Invalid shape type.");

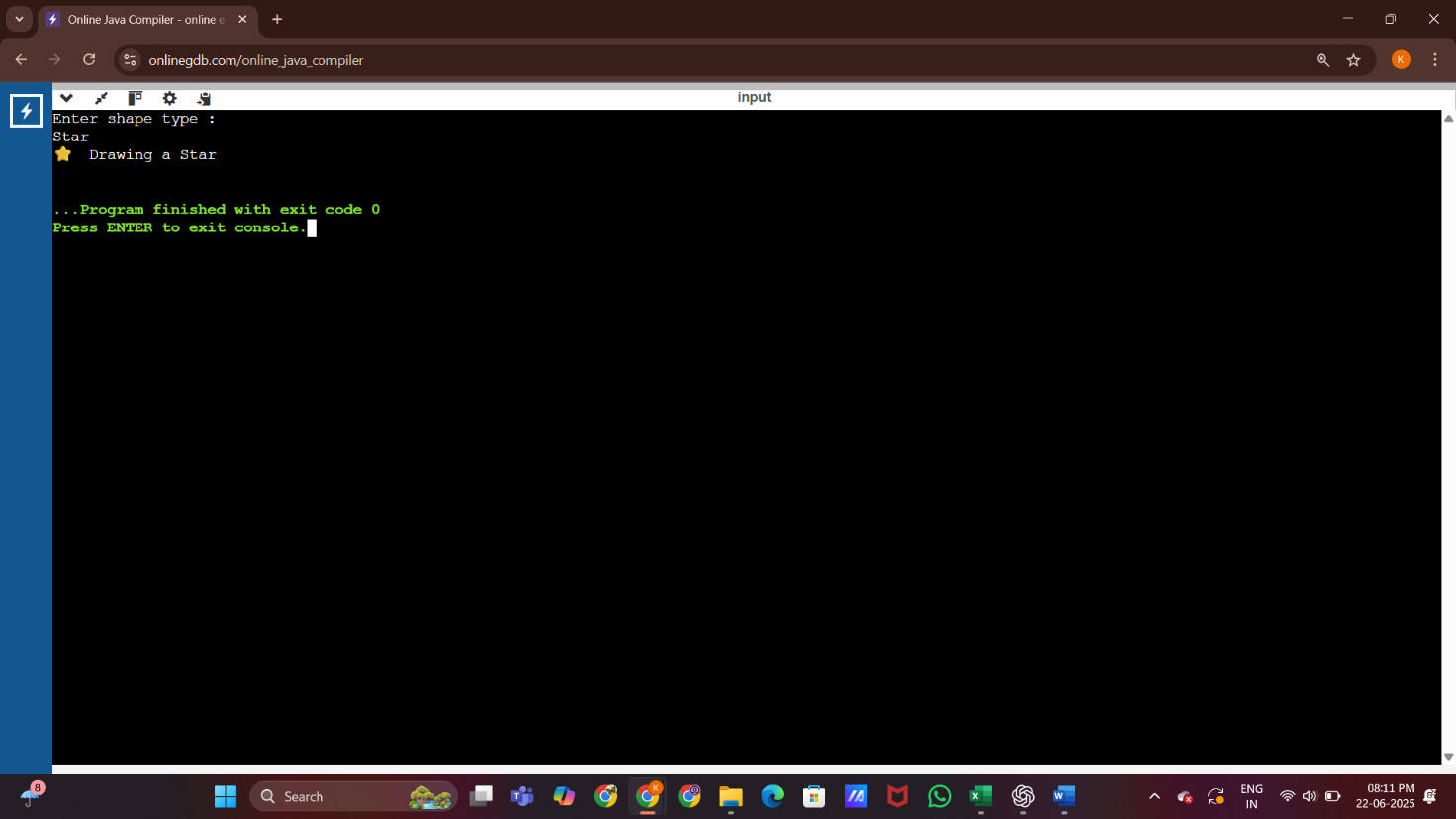
}

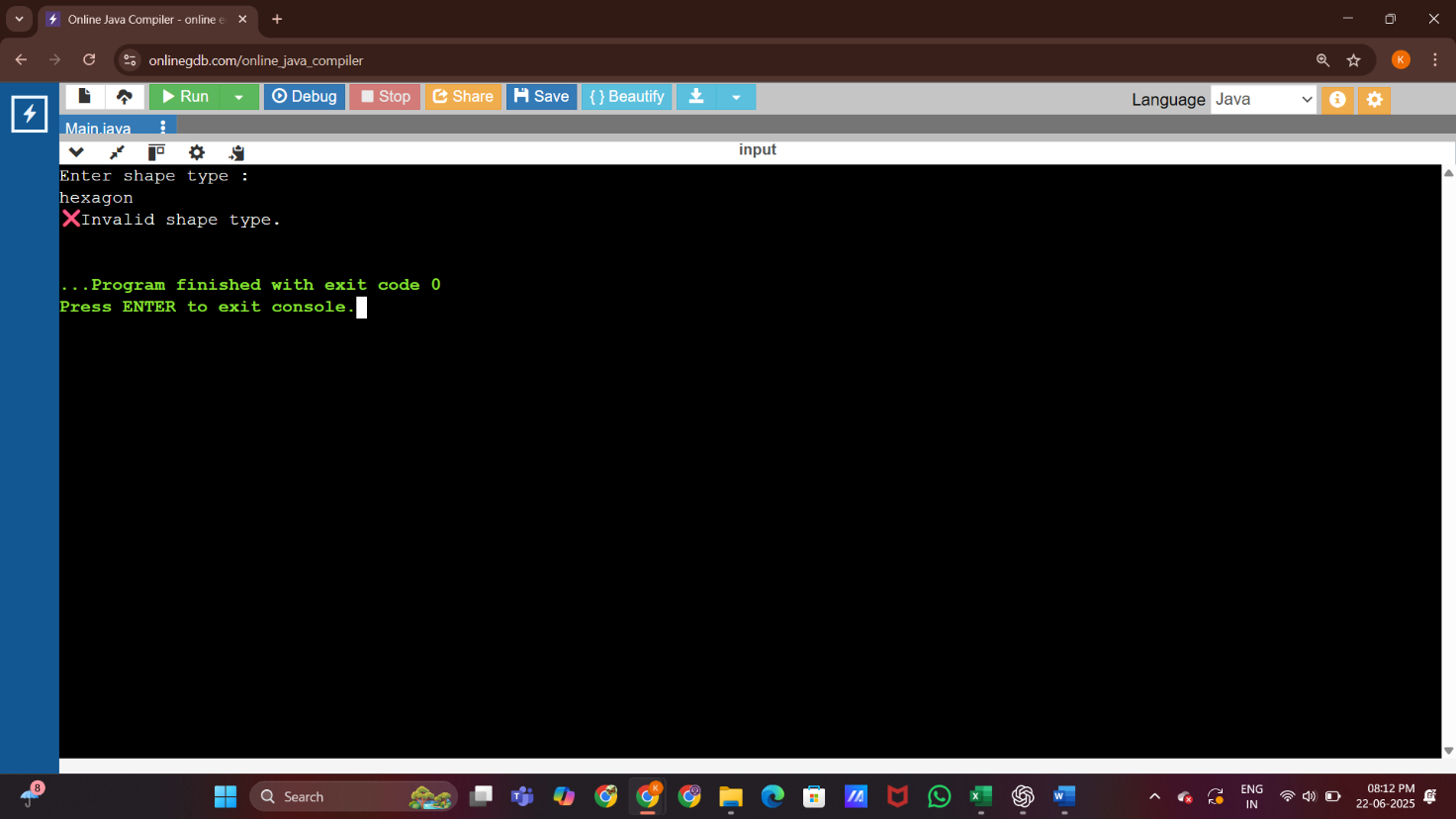
scanner.close();

}

}

**Output :**

****

****

**ALGORITHMS AND DATA STRUCTURES**

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

**Source Code:**

import java.util.\*;

class Product {

int id;

String name;

double price;

double rating;

public Product(int id, String name, double price, double rating) {

this.id = id;

this.name = name;

this.price = price;

this.rating = rating;

}

public void display() {

System.out.println("ID : " + id);

System.out.println("Name : " + name);

System.out.println("Price : $" + price);

System.out.println("Rating : " + rating + " ★");

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

}

}

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

System.out.println(" Welcome to ClicktiqueHub !!!");

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

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

products.add(new Product(101, "Wireless Mouse", 499, 4.2));

products.add(new Product(102, "Bluetooth Speaker", 1199, 4.5));

products.add(new Product(103, "Laptop Stand", 799, 3.9));

products.add(new Product(104, "USB Cable", 199, 4.0));

products.add(new Product(105, "Gaming Keyboard", 2499, 4.8));

products.add(new Product(106, "VCS Gaming Keyboard", 999, 4.3));

products.add(new Product(107, "Grocery Kit(Small)", 299, 4.0));

products.add(new Product(108, "Grocery Kit(Large)", 699, 4.1));

products.add(new Product(109, "File Organizer", 199, 3.8));

System.out.print("Enter product name : ");

String keyword = scanner.nextLine();

System.out.print("Do you want to filter by price range? (Y/N): ");

String choice = scanner.nextLine().trim();

double minPrice = 0;

double maxPrice = Double.MAX\_VALUE;

if (choice.equalsIgnoreCase("Y")) {

System.out.print("Enter minimum price: ");

minPrice = scanner.nextDouble();

System.out.print("Enter maximum price: ");

maxPrice = scanner.nextDouble();

}

boolean found = false;

System.out.println("\nFiltered Products:");

for (Product p : products) {

if (p.name.contains(keyword) && p.price >= minPrice && p.price <= maxPrice) {

p.display();

found = true;

}

}

if (!found) {

System.out.println("No products found matching your criteria.");

}

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

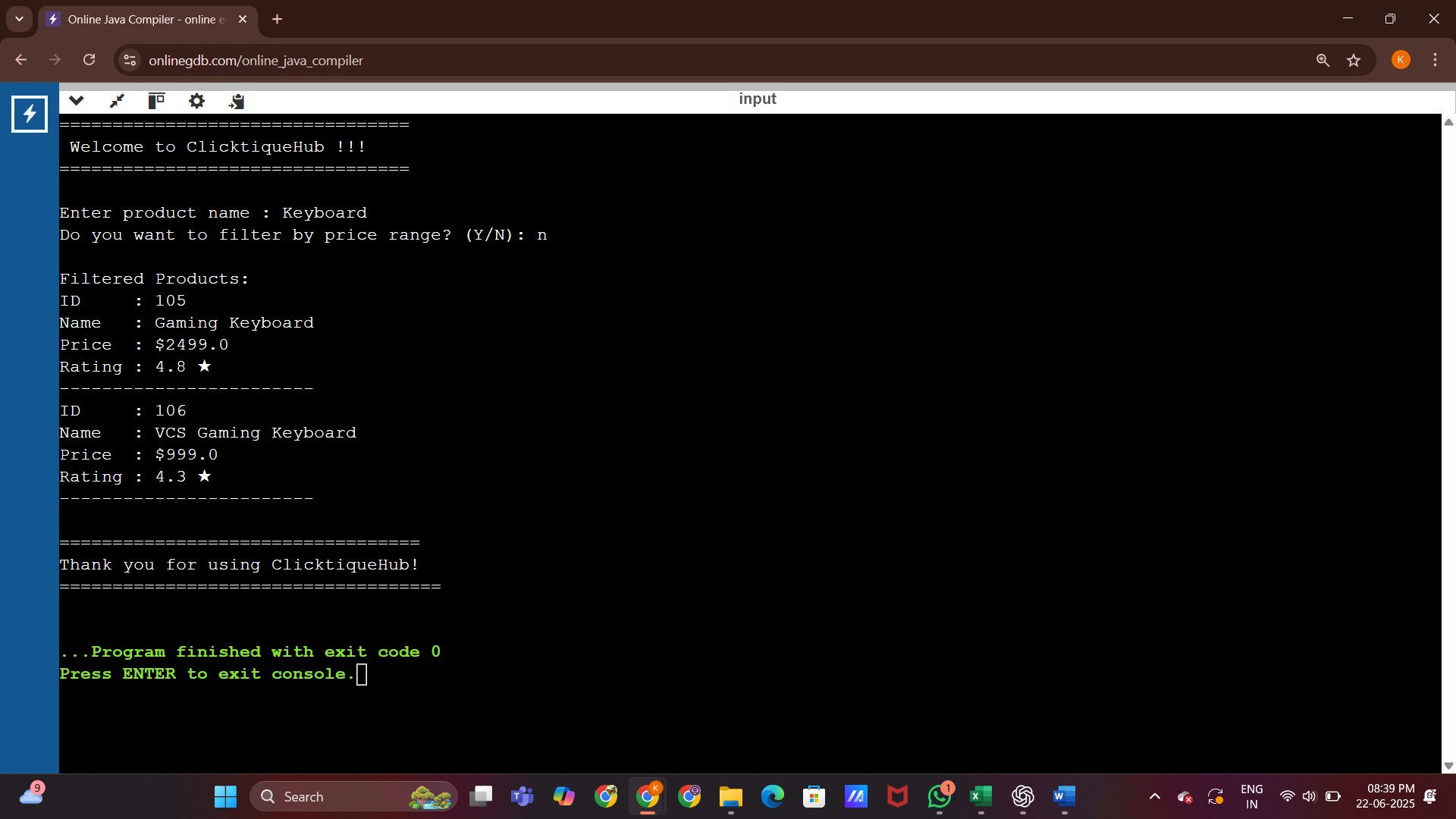
System.out.println("Thank you for using ClicktiqueHub!");

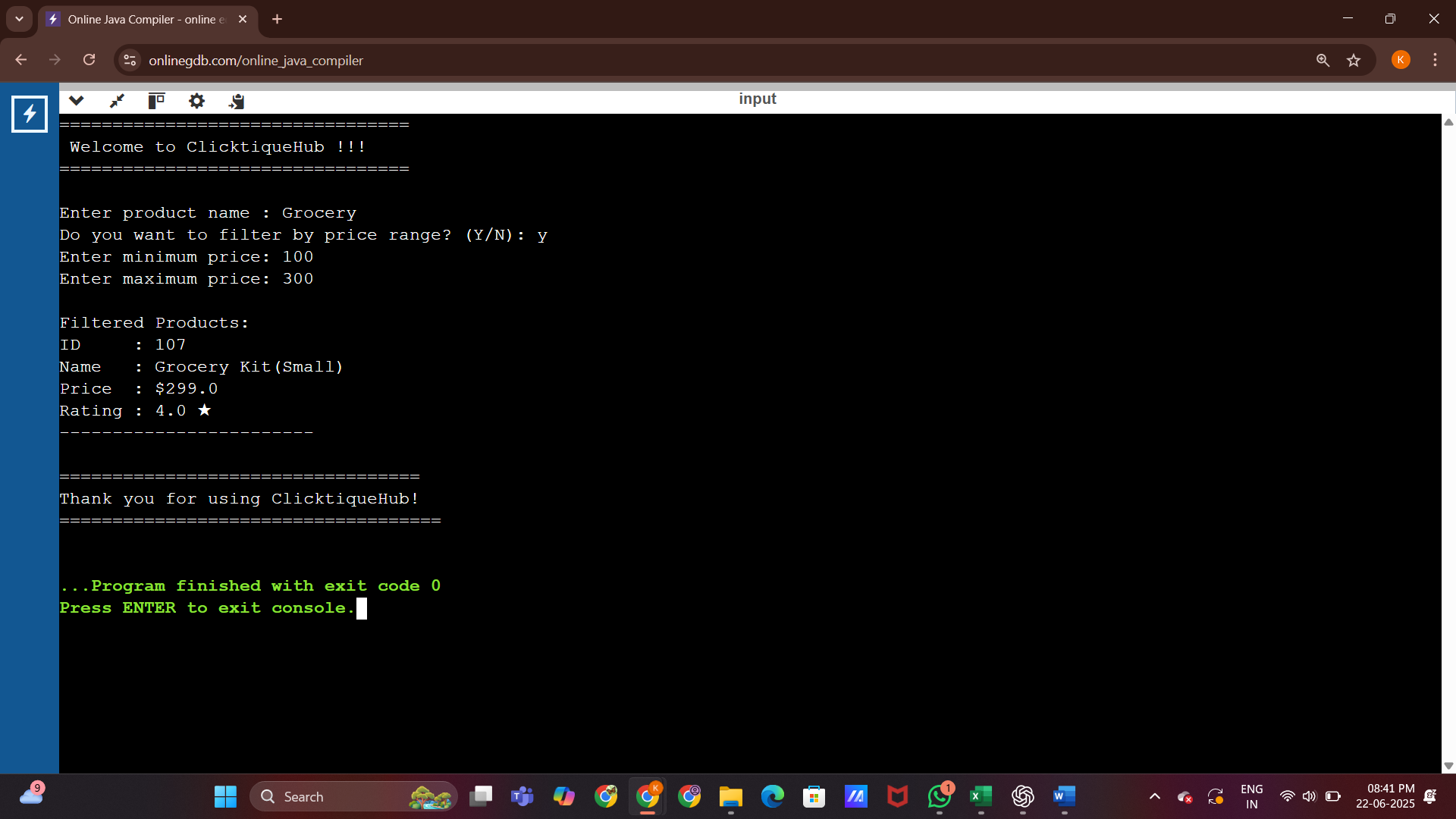
System.out.println("====================================");

}

}

**Output:**





**Exercise 7:** Financial forecasting

**Source Code:**

import java.util.\*;

public class FinancialForecasting {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter number of past months of revenue data: ");

int months = scanner.nextInt();

if (months < 2) {

System.out.println("Need at least 2 months of data to forecast.");

return;

}

double[] revenues = new double[months];

double[] expenses = new double[months];

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

System.out.print("Enter revenue for Month " + (i + 1) + ": ₹");

revenues[i] = scanner.nextDouble();

System.out.print("Enter expense for Month " + (i + 1) + ": ₹");

expenses[i] = scanner.nextDouble();

}

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

int futureMonths = scanner.nextInt();

double totalRevGrowth = 0, totalExpGrowth = 0;

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

totalRevGrowth += (revenues[i] - revenues[i - 1]) / revenues[i - 1];

totalExpGrowth += (expenses[i] - expenses[i - 1]) / expenses[i - 1];

}

double avgRevGrowth = totalRevGrowth / (months - 1);

double avgExpGrowth = totalExpGrowth / (months - 1);

System.out.printf("\nEstimated Revenue Growth Rate: %.2f%%\n", avgRevGrowth \* 100);

System.out.printf("Estimated Expense Growth Rate: %.2f%%\n", avgExpGrowth \* 100);

double forecastRev = revenues[months - 1];

double forecastExp = expenses[months - 1];

System.out.println("\nForecasted Revenue, Expense, and Profit:\n");

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

forecastRev \*= (1 + avgRevGrowth);

forecastExp \*= (1 + avgExpGrowth);

double profit = forecastRev - forecastExp;

System.out.printf("Month %d: Revenue ₹%.2f | Expense ₹%.2f | Profit ₹%.2f\n",

months + i, forecastRev, forecastExp, profit);

}

}

}

**Output:**

