# Exercise 1: Implementing the Singleton Pattern

// You are using Java

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 msg)

    {

        System.out.println("[LOG]:"+msg);

    }

}

class App{

    public static void main(String[] args)

    {

        Logger l1=Logger.getInstance();

        l1.log("Application started successfully");

        Logger l2=Logger.getInstance();

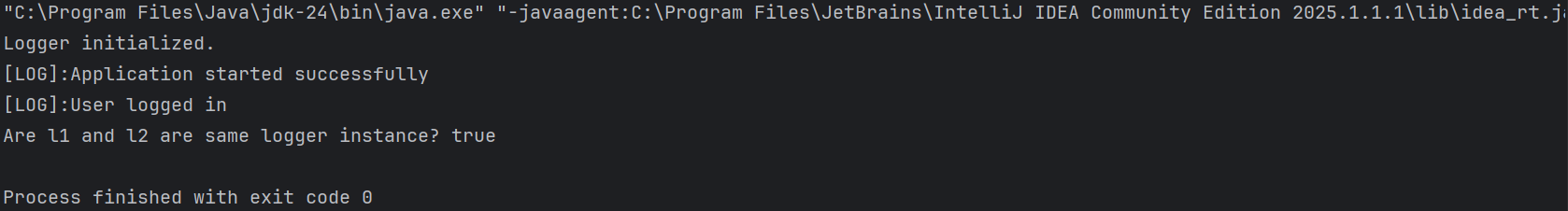
        l2.log("User logged in");

        System.out.println("Are l1 and l2 are same logger instance? "+(l1==l2));

    }

}

**OUTPUT**



# Exercise 2: Implementing the Factory Method Pattern

// You are using Java

interface Shape{//common interface for all shapes

    void draw();

}

class Circle implements Shape{

    public void draw()

    {

        System.out.println("Drawing a Circle⭕");

    }

}

class Rectangle implements Shape{

    public void draw()

    {

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

    }

}

class Square implements Shape{

    public void draw()

    {

        System.out.println("Drawing a Square🔲");

    }

}

class Pentagon implements Shape{

    public void draw()

    {

        System.out.println("Drawing a Pentagon");

    }

}

class ShapeFactory{

    public Shape getShape(String s)

    {

        if(s==null)

        {

            return null;

        }

        switch(s.toLowerCase())

        {

            case "circle":

                return new Circle();

            case "rectangle":

                return new Rectangle();

            case "square":

                return new Square();

            case "pentagon":

                return new Pentagon();

            default:

            return null;

        }

    }

}

class Main{

    public static void main(String[] args)

    {

        ShapeFactory sf=new ShapeFactory();

        Shape s1=sf.getShape("circle");

        s1.draw();

        Shape s2=sf.getShape("Rectangle");

        s2.draw();

        Shape s3=sf.getShape("Square");

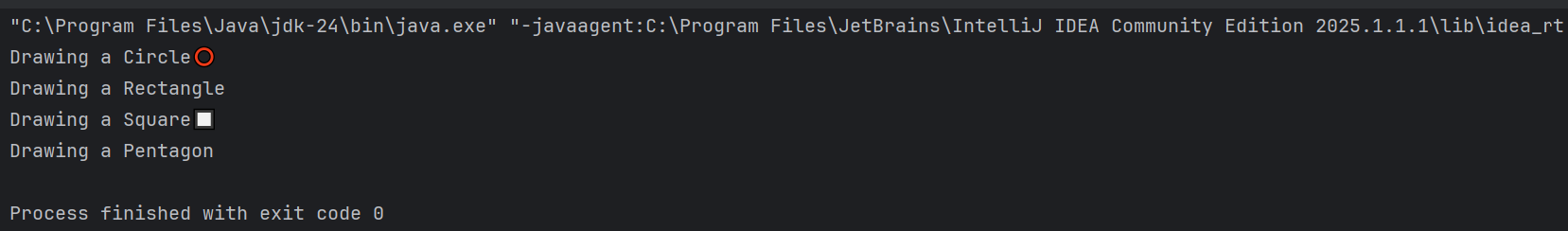
        s3.draw();

        Shape s4=sf.getShape("Pentagon");

        s4.draw();

    }

}

**OUTPUT  
  
**

# Exercise 2: E-commerce Platform Search Function

// You are using Java

import java.util.\*;

class Product{

    String name;

    String category;

    double price;

    public Product(String name,String category,double price)

    {

        this.name=name;

        this.category=category;

        this.price=price;

    }

    public String toString()

    {

        return name+" | "+category+" | $"+price;

    }

}

class TrieNode{

    Map<Character,TrieNode> Children=new HashMap<>();

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

    boolean isEndOfWord=false;

}

class ProductTrie{

    private TrieNode root=new TrieNode();

    public void insertProduct(Product product) {

        String[] keywords = product.name.toLowerCase().split(" ");

        for (String keyword : keywords) {

            TrieNode node = root;

            for (char ch : keyword.toCharArray()) {

                node = node.Children.computeIfAbsent(ch, c -> new TrieNode());

                node.products.add(product);

            }

            node.isEndOfWord = true;

        }

    }

    public List<Product> search(String prefix) {

        TrieNode node = root;

        for (char ch : prefix.toLowerCase().toCharArray()) {

            node = node.Children.get(ch);

            if (node == null) return Collections.emptyList();

        }

        return node.products;

    }

}

class ECommerceSearch {

    public static void main(String[] args) {

        ProductTrie searchIndex = new ProductTrie();

        searchIndex.insertProduct(new Product("Red Shirt", "Clothing", 499));

        searchIndex.insertProduct(new Product("Red Sneakers", "Footwear", 1199));

        searchIndex.insertProduct(new Product("Smart Refrigerator", "Appliances", 24999));

        searchIndex.insertProduct(new Product("Blue Shirt", "Clothing", 599));

        searchIndex.insertProduct(new Product("Bluetooth Speaker", "Electronics", 899));

        searchIndex.insertProduct(new Product("Reebok Shoes", "Footwear", 2199));

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter search keyword: ");

        String keyword = sc.nextLine();

        System.out.print(keyword);

        List<Product> results = searchIndex.search(keyword);

        results.sort(Comparator.comparingDouble(p -> p.price));

        System.out.println("\n🔎 Search Results:");

        if (results.isEmpty()) {

            System.out.println("No products found.");

        } else {

            for (Product p : results) {

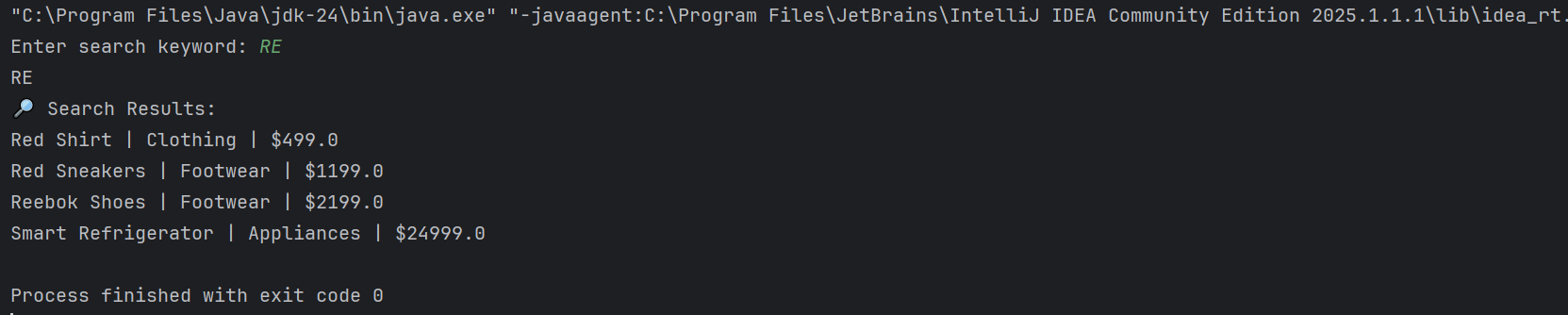
                System.out.println(p);

            }

        }

    }

}

**OUTPUT  
**

# Exercise 7: Financial Forecasting

// You are using Java

import java.util.\*;

class FinancialData{

    int month;

    double income;

    double expense;

    FinancialData(int month,double income,double expense)

    {

        this.month=month;

        this.income=income;

        this.expense=expense;

    }

    double getBalance()

    {

        return income-expense;

    }

    public String toString()

    {

        return "Month "+month+": Income ₹ "+income+", Expense ₹"+expense+", Balance ₹"+getBalance();

    }

}

class FinancialForecasting{

    static List<FinancialData> data=new ArrayList<>();

    static Stack<FinancialData> undo=new Stack<>();

    public static void main(String[] args)

    {

        Scanner sc=new Scanner(System.in);

        System.out.println("Enter number of months to analyze:");

        int n=sc.nextInt();

        for(int i=1;i<=n;i++)

        {

            System.out.println("Month "+i+" Income: ");

            double income=sc.nextDouble();

            System.out.println("Month "+i+" Expense: ");

            double expense=sc.nextDouble();

            FinancialData fd=new FinancialData(i,income,expense);

            data.add(fd);

            undo.push(fd);

        }

        System.out.println("\n📊 Monthly Report:");

        for(FinancialData fd:data)

        {

            System.out.println(fd);

        }

        forecastNextMonthBalance(3);

        showTop3Expenses();

        undoLastEntry();

    }

    static void forecastNextMonthBalance(int N)

    {

        int size = data.size();

        if (size < N) {

            System.out.println("\nNot enough data to forecast.");

            return;

        }

        double totalIncome = 0, totalExpense = 0;

        for (int i = size - N; i < size; i++) {

            totalIncome += data.get(i).income;

            totalExpense += data.get(i).expense;

        }

        double avgIncome = totalIncome / N;

        double avgExpense = totalExpense / N;

        System.out.println("\n📈 Forecast for Next Month:");

        System.out.println("Expected Income: ₹" + avgIncome);

        System.out.println("Expected Expense: ₹" + avgExpense);

        System.out.println("Expected Balance: ₹" + (avgIncome - avgExpense));

    }

    static void showTop3Expenses()

    {

        List<FinancialData> sorted = new ArrayList<>(data);

        sorted.sort((a, b) -> Double.compare(b.expense, a.expense));

        System.out.println("\n🔥 Top 3 Expense Months:");

        for (int i = 0; i < Math.min(3, sorted.size()); i++) {

            System.out.println(sorted.get(i));

        }

    }

    static void undoLastEntry()

    {

        if (!undo.isEmpty()) {

            FinancialData removed = undo.pop();

            data.remove(data.size() - 1);

            System.out.println("\n⏪ Last entry undone: " + removed);

        } else {

            System.out.println("\nUndo stack is empty.");

        }

    }

}

**OUTPUT**

