# CHALLA VENKATA ANIRUDH

## ALGORITHMS DATA STRUCTURES

## EXERCISE 2: E-COMMERCE PLATFORM SEARCH FUNCTION

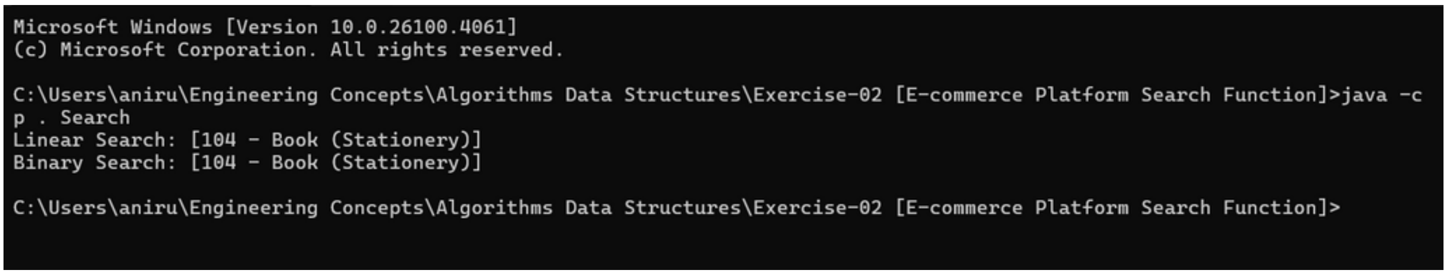
### SCENARIO:

YOU ARE WORKING ON THE SEARCH FUNCTIONALITY OF AN E-COMMERCE PLATFORM. THE SEARCH NEEDS TO BE OPTIMIZED FOR FAST PERFORMANCE.

STEPS:

1. UNDERSTAND ASYMPTOTIC NOTATION:
   - EXPLAIN BIG O NOTATION AND HOW IT HELPS IN ANALYZING ALGORITHMS.
   - DESCRIBE THE BEST, AVERAGE, AND WORST-CASE SCENARIOS FOR SEARCH OPERATIONS.
2. SETUP:
   - CREATE A CLASS PRODUCT WITH ATTRIBUTES FOR SEARCHING, SUCH AS PRODUCTID, PRODUCTNAME, AND CATEGORY.
3. IMPLEMENTATION:
   - IMPLEMENT LINEAR SEARCH AND BINARY SEARCH ALGORITHMS.
   - STORE PRODUCTS IN AN ARRAY FOR LINEAR SEARCH AND A SORTED ARRAY FOR BINARY SEARCH.
4. ANALYSIS:
   - COMPARE THE TIME COMPLEXITY OF LINEAR AND BINARY SEARCH ALGORITHMS.
   - DISCUSS WHICH ALGORITHM IS MORE SUITABLE FOR YOUR PLATFORM AND WHY.

### OUTPUT:-



```
Microsoft Windows [Version 10.0.26100.4061]
(c) Microsoft Corporation. All rights reserved.

C:\Users\aniru\Engineering Concepts\Algorithms Data Structures\Exercise-02 [E-commerce Platform Search Function]>java -c
p . Search
Linear Search: [104 - Book (Stationery)]
Binary Search: [104 - Book (Stationery)]

C:\Users\aniru\Engineering Concepts\Algorithms Data Structures\Exercise-02 [E-commerce Platform Search Function]>
```

## EXERCISE 7: FINANCIAL FORECASTING

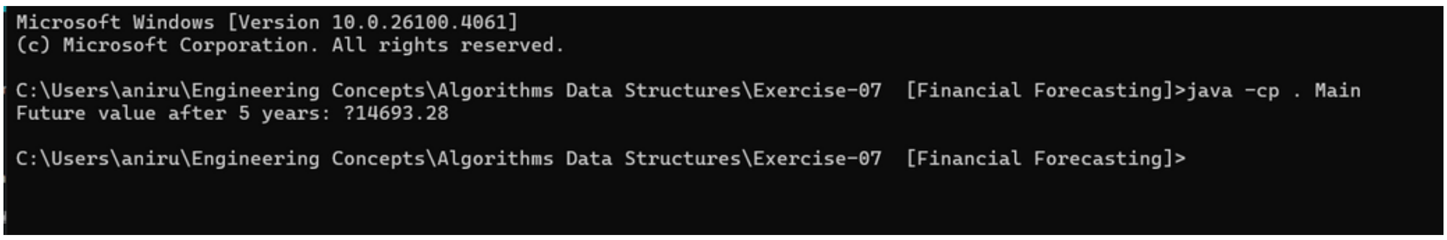
## SCENARIO:

YOU ARE DEVELOPING A FINANCIAL FORECASTING TOOL THAT PREDICTS FUTURE VALUES BASED ON PAST DATA.

STEPS:

1. UNDERSTAND RECURSIVE ALGORITHMS:
   - EXPLAIN THE CONCEPT OF RECURSION AND HOW IT CAN SIMPLIFY CERTAIN PROBLEMS.
2. SETUP:
   - CREATE A METHOD TO CALCULATE THE FUTURE VALUE USING A RECURSIVE APPROACH.
3. IMPLEMENTATION:
   - IMPLEMENT A RECURSIVE ALGORITHM TO PREDICT FUTURE VALUES BASED ON PAST GROWTH RATES.
4. ANALYSIS:
   - DISCUSS THE TIME COMPLEXITY OF YOUR RECURSIVE ALGORITHM.
   - EXPLAIN HOW TO OPTIMIZE THE RECURSIVE SOLUTION TO AVOID EXCESSIVE COMPUTATION.

## OUTPUT:-



```
Microsoft Windows [Version 10.0.26100.4061]
(c) Microsoft Corporation. All rights reserved.

C:\Users\aniru\Engineering Concepts\Algorithms Data Structures\Exercise-07  [Financial Forecasting]>java -cp . Main
Future value after 5 years: ?14693.28

C:\Users\aniru\Engineering Concepts\Algorithms Data Structures\Exercise-07  [Financial Forecasting]>
```

# DESIGN PATTERNS AND PRINCIPLES

## EXERCISE 1: IMPLEMENTING THE SINGLETON PATTERN

## SCENARIO:

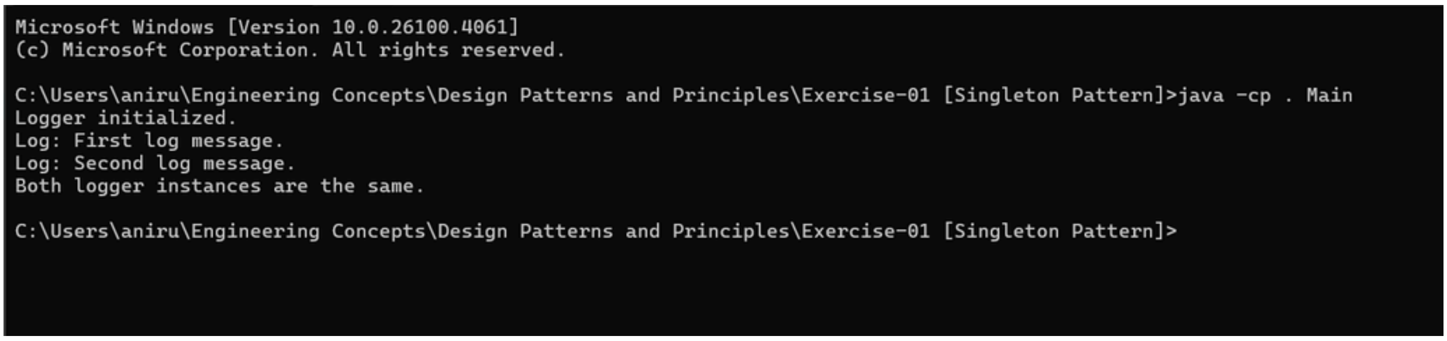
YOU NEED TO ENSURE THAT A LOGGING UTILITY CLASS IN YOUR APPLICATION HAS ONLY ONE INSTANCE THROUGHOUT THE APPLICATION LIFECYCLE TO ENSURE CONSISTENT LOGGING.

STEPS:

1. CREATE A NEW JAVA PROJECT:
   - CREATE A NEW JAVA PROJECT NAMED SINGLETONPATTERNEXAMPLE.
2. DEFINE A SINGLETON CLASS:
   - CREATE A CLASS NAMED LOGGER THAT HAS A PRIVATE STATIC INSTANCE OF ITSELF.
   - ENSURE THE CONSTRUCTOR OF LOGGER IS PRIVATE.
   - PROVIDE A PUBLIC STATIC METHOD TO GET THE INSTANCE OF THE LOGGER CLASS.
3. IMPLEMENT THE SINGLETON PATTERN:
   - WRITE CODE TO ENSURE THAT THE LOGGER CLASS FOLLOWS THE SINGLETON DESIGN PATTERN.

4. **TEST THE SINGLETON IMPLEMENTATION:**
    - CREATE A TEST CLASS TO VERIFY THAT ONLY ONE INSTANCE OF LOGGER IS CREATED AND USED ACROSS THE APPLICATION

## OUTPUT:-



```
Microsoft Windows [Version 10.0.26100.4061]
(c) Microsoft Corporation. All rights reserved.

C:\Users\aniru\Engineering Concepts\Design Patterns and Principles\Exercise-01 [Singleton Pattern]>java -cp . Main
Logger initialized.
Log: First log message.
Log: Second log message.
Both logger instances are the same.

C:\Users\aniru\Engineering Concepts\Design Patterns and Principles\Exercise-01 [Singleton Pattern]>
```

---

## EXERCISE 2: IMPLEMENTING THE FACTORY METHOD PATTERN

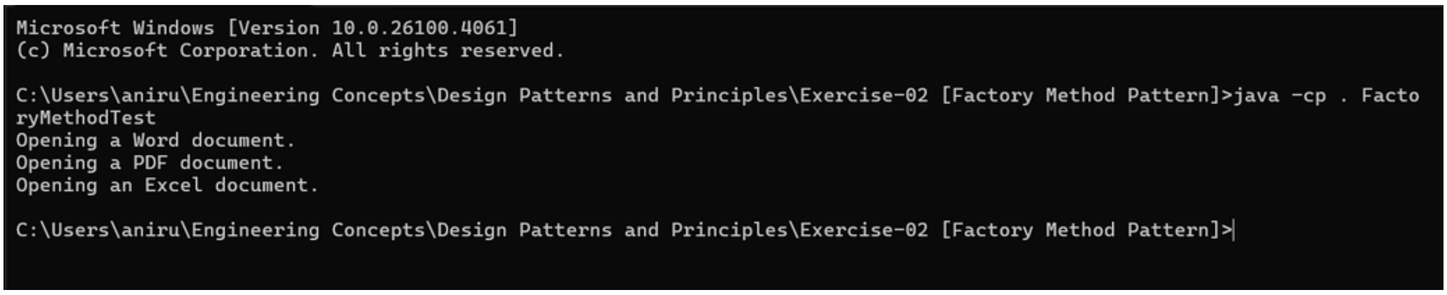
## SCENARIO:

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.

STEPS:

1. CREATE A NEW JAVA PROJECT:
    - CREATE A NEW JAVA PROJECT NAMED FACTORYMETHODPATTERNEXAMPLE.
2. DEFINE DOCUMENT CLASSES:
    - CREATE INTERFACES OR ABSTRACT CLASSES FOR DIFFERENT DOCUMENT TYPES SUCH AS WORDDOCUMENT, PDFDOCUMENT, AND EXCELDOCUMENT.
3. CREATE CONCRETE DOCUMENT CLASSES:
    - IMPLEMENT CONCRETE CLASSES FOR EACH DOCUMENT TYPE THAT IMPLEMENTS OR EXTENDS THE ABOVE INTERFACES OR ABSTRACT CLASSES.
4. IMPLEMENT THE FACTORY METHOD:
    - CREATE AN ABSTRACT CLASS DOCUMENTFACTORY WITH A METHOD CREATEDOCUMENT().
    - CREATE CONCRETE FACTORY CLASSES FOR EACH DOCUMENT TYPE THAT EXTENDS DOCUMENTFACTORY AND IMPLEMENTS THE CREATEDOCUMENT() METHOD.
5. TEST THE FACTORY METHOD IMPLEMENTATION:
    - CREATE A TEST CLASS TO DEMONSTRATE THE CREATION OF DIFFERENT DOCUMENT TYPES USING THE FACTORY METHOD.

## OUTPUT:-



```
Microsoft Windows [Version 10.0.26100.4061]
(c) Microsoft Corporation. All rights reserved.

C:\Users\aniru\Engineering Concepts\Design Patterns and Principles\Exercise-02 [Factory Method Pattern]>java -cp . Facto
ryMethodTest
Opening a Word document.
Opening a PDF document.
Opening an Excel document.

C:\Users\aniru\Engineering Concepts\Design Patterns and Principles\Exercise-02 [Factory Method Pattern]>
```