**Cognizant Deep Skilling Mandatory Hands-On Questions**

**TOPIC- Design pattern and Principles**

1. **Implementing the Singleton Pattern**

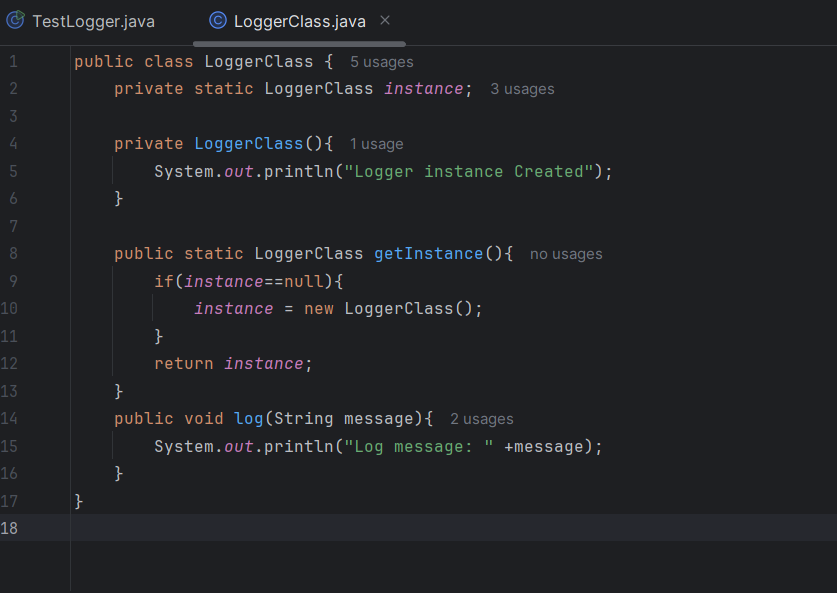
Scenario:

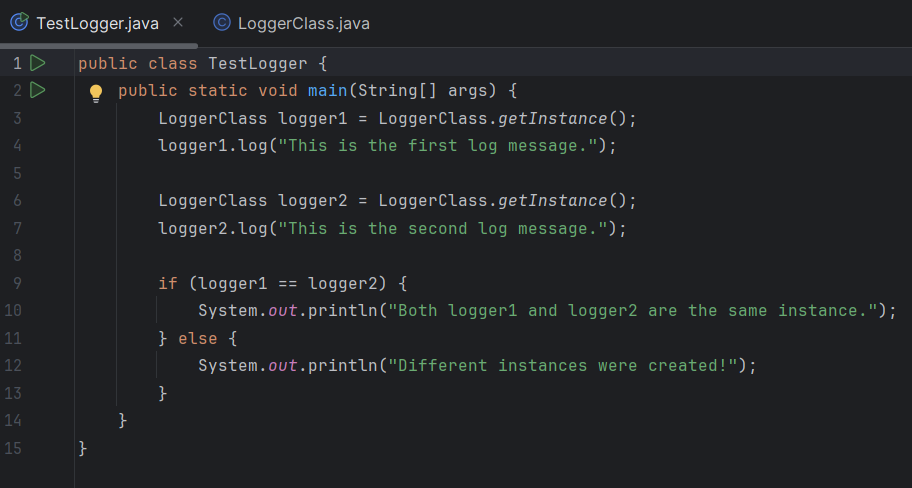
You need to ensure that a logging utility class in your application has only one instance throughout the application lifecycle to ensure consistent logging.

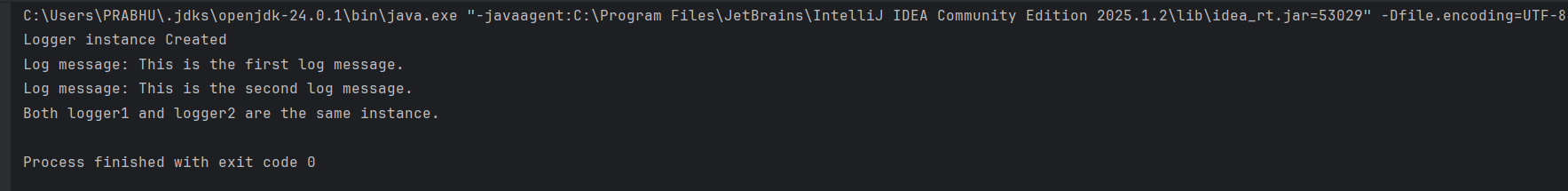
Ans)

**CODE**:

LoggerClass:



TestLogger Class:

OUTPUT:

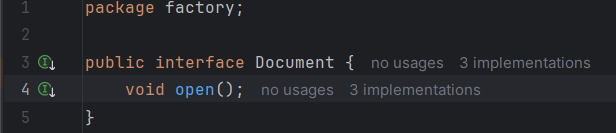
1. **Exercise 2: Implementing the Factory Method Pattern**

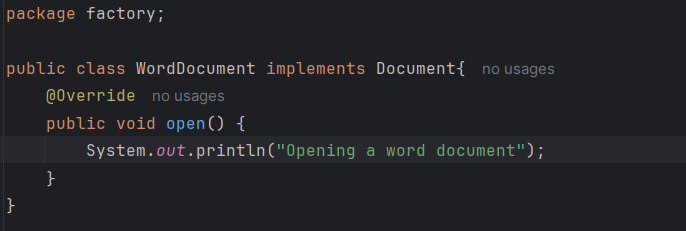
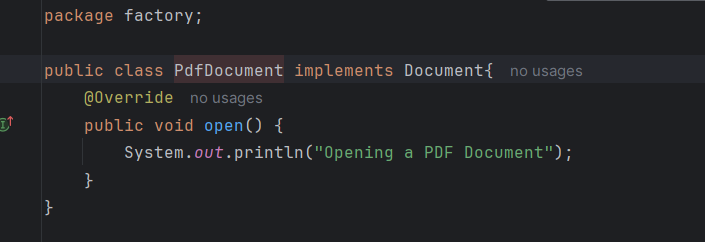
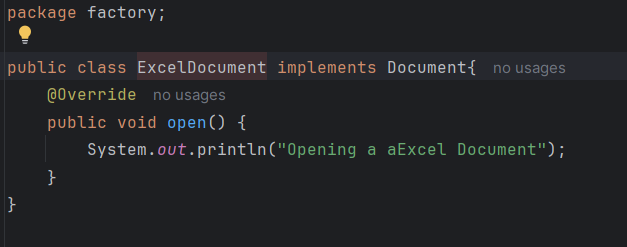
Scenario**:**

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.

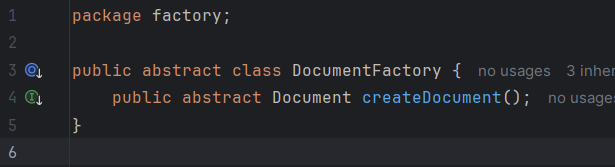
Ans)

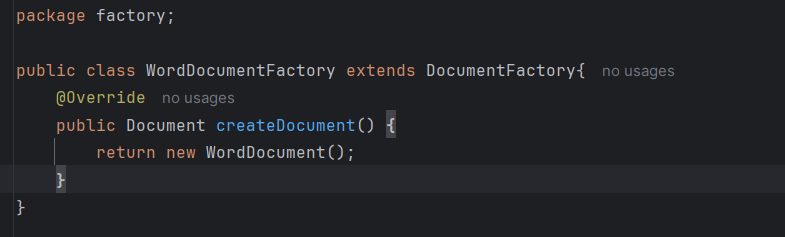
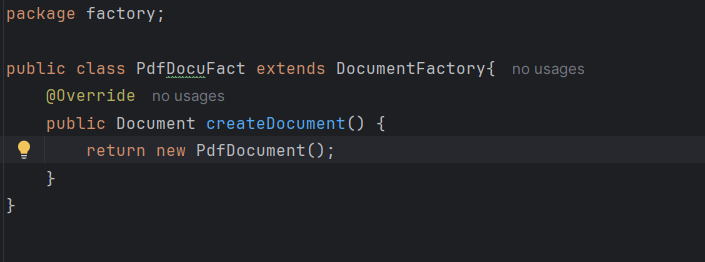
CODE:  
Create a Document Interface:

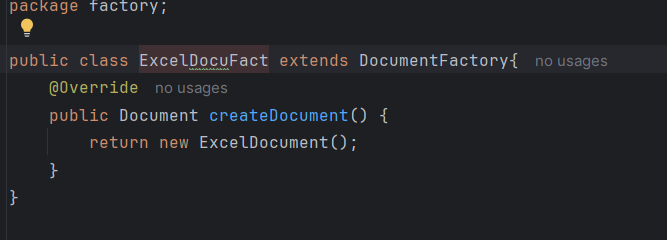


Create different classes:

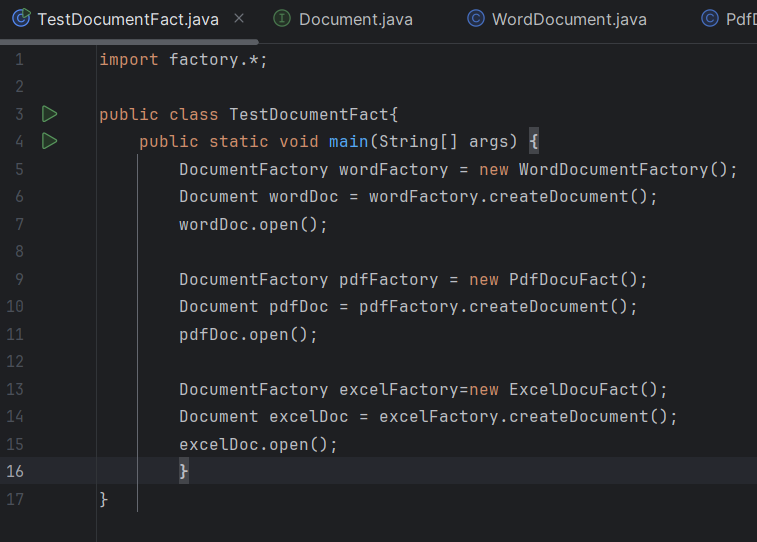
Create the abstract factory class:

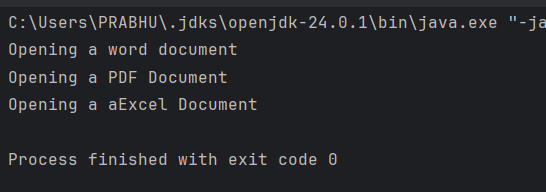


Create Concrete Factory classes:



Create TestDocument (Main) Class:



Output:

**TOPIC- Data structures and Algorithms**

**Q.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**.**

Ans)

**Big O Notation** :- Big O notation is a mathematical concept used to describe how an algorithm’s runtime grows as the input size (n) increases. It allows developers to evaluate and compare the efficiency of different algorithms.

**Common Notations:**

| **Big O** | **Meaning** | **Example Algorithm** |
| --- | --- | --- |
| O(1) | Constant time | Accessing an array index |
| O(log n) | Logarithmic | Binary search |
| O(n) | Linear time | Linear search |
| O(n log n) | Linearithmic | Merge sort, Quick sort |
| O(n²) | Quadratic | Bubble sort, Nested loops |

Two Types of Searching:

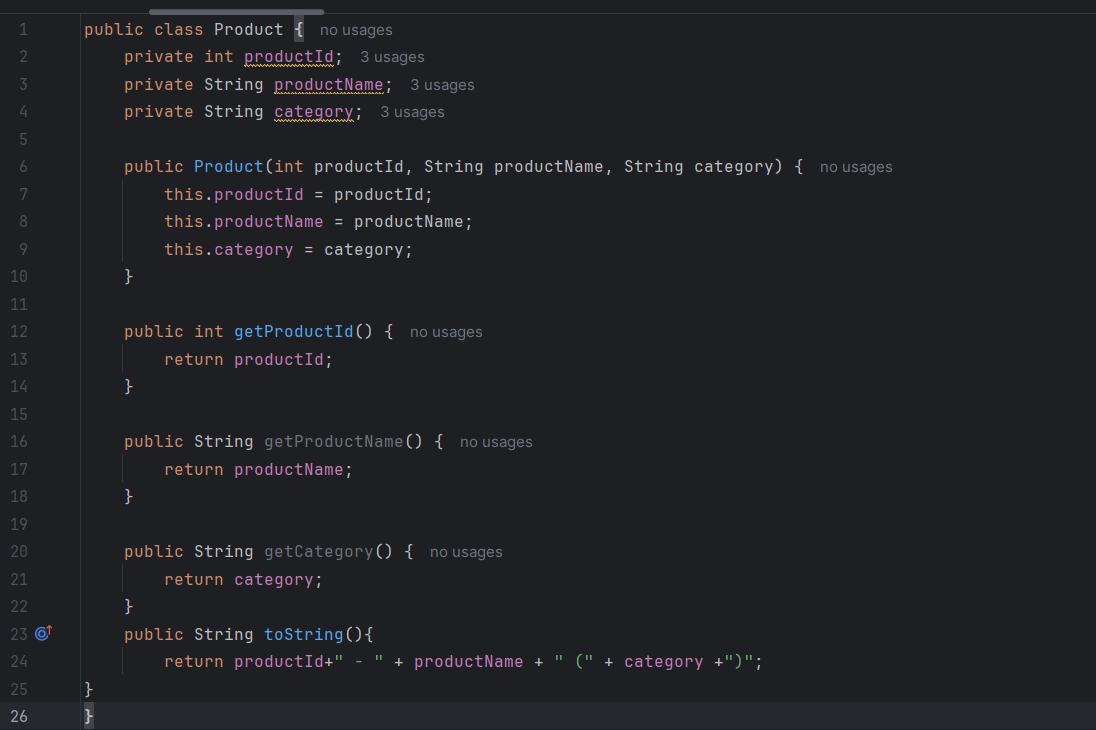
**Linear Search**: Scans one by one. Best case is when the item is first.

**Binary Search**: Only works on sorteddata and splits the array every step.

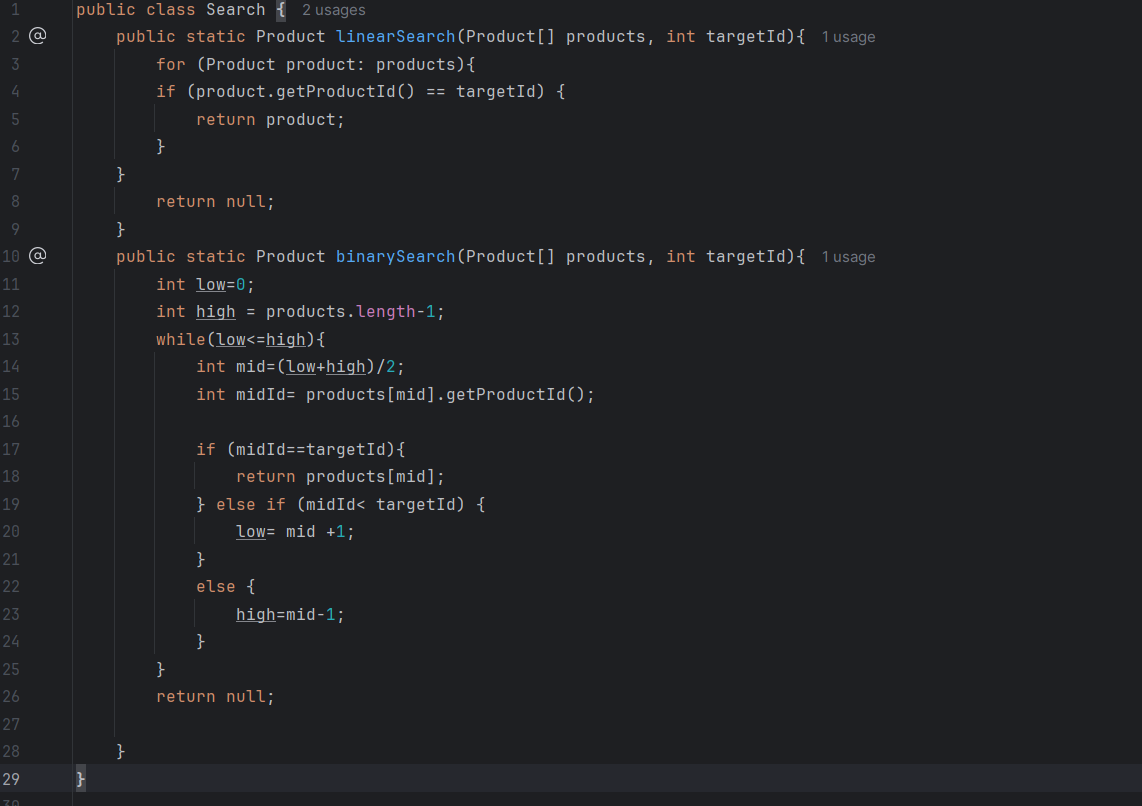
**Best, Average, and Worst Case for Searching**

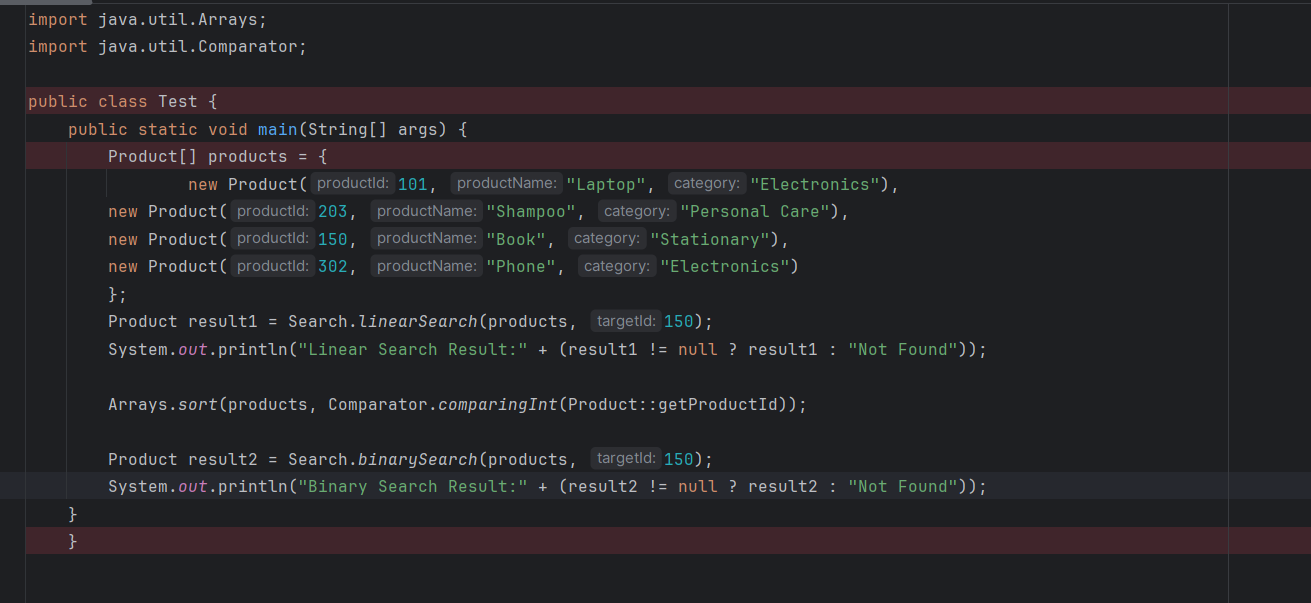
| **Algorithm** | **Best Case** | **Average Case** | **Worst Case** |
| --- | --- | --- | --- |
| Linear Search | O(1) | O(n) | O(n) |
| Binary Search | O(1) | O(log n) | O(log n) |

CODE:

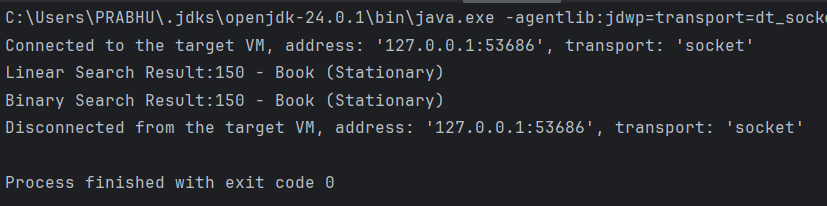
Product Class:

Search Class:



Test Class:

Output:



Binary Search is faster (O(log n)), but:

It requires a sorted array.

Sorting takes O(n log n) (done once if data doesn't change frequently).

Use Binary Search if:

Product list is large.

Products are sorted or can be sorted once and reused.

Use Linear Search if:

Small number of products.

Array is unsorted and performance is not critical.

**Q.7: Financial Forecasting**

**Scenario:**

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

Ans)

What is Recursion?

Recursion is when a method callsitself to solve a smaller instance of the same problem.

It’s divided into two parts**:**

* Basecase**:** Stops recursion to prevent infinite calls.
* Recursivecase**:** Reduces the problem in each call.

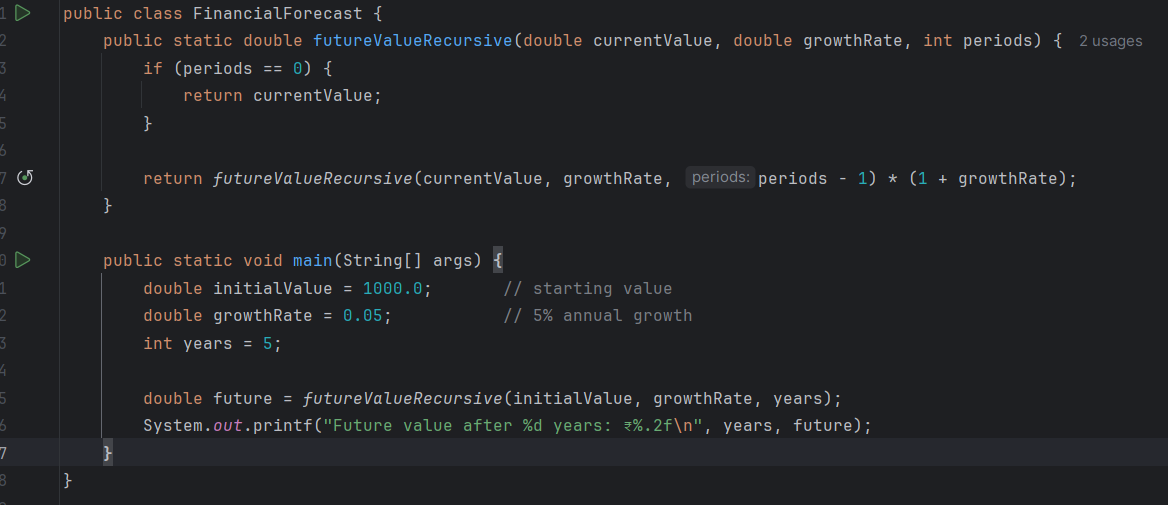
We should use recursions because It simplifies problems that have a repetitiveorhierarchicalstructure such as: Tree traversal, Factorial or Fibonacci and Forecasting future values from previous data.

**Forecasting Future Value**

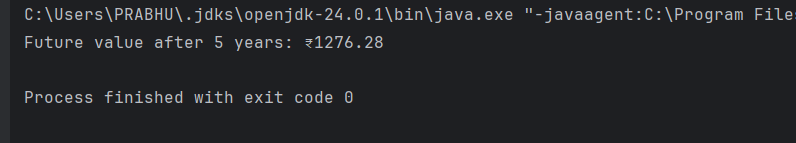
We will calculate futurefinancialvalue based on:

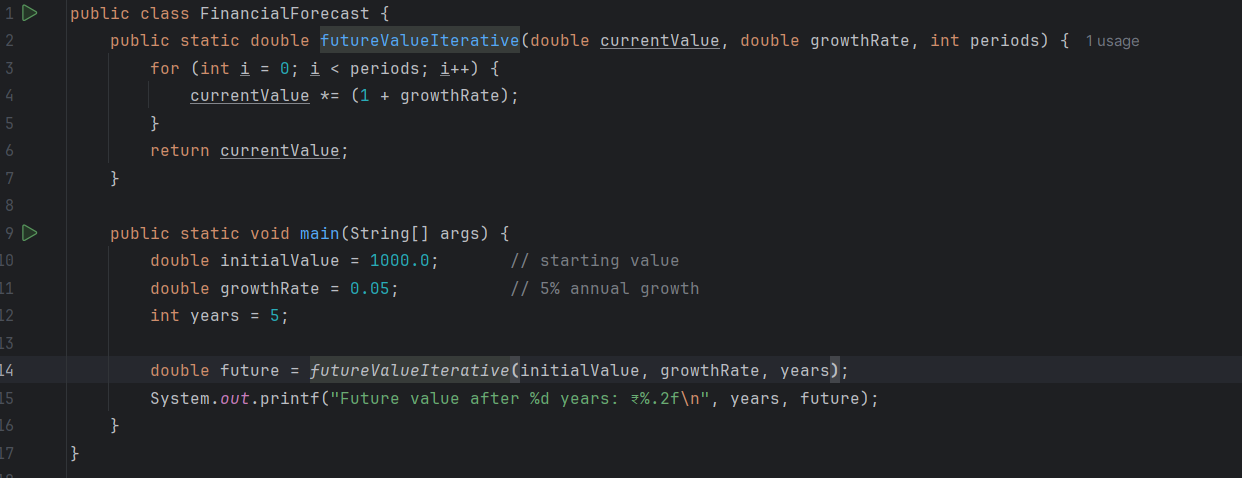
* Currentvalue
* Growthrate **(**asa percentage)
* Number of periods (years/months/etc.)

CODE:

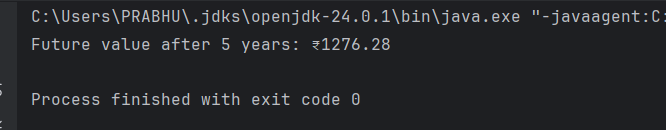
Recursive Algorithm:

Output:



Iterative Algorithm: This avoids recursion and is more efficient for very large inputs.

Output:



Comparison between both algorithms:

| **Feature** | **Recursive** | **Iterative** |
| --- | --- | --- |
| Simplicity | Easy to write | Slightly more code |
| Time Complexity | O(n) | O(n) |
| Space Complexity (Stack) | O(n) | O(1) |
| Risk | Stack overflow for large n | Safe for large n |