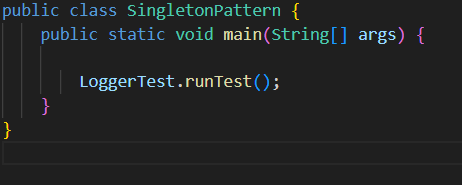
**Week–1**

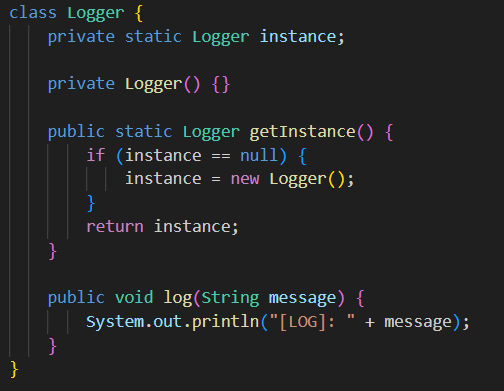
**Hands-on Exercises**

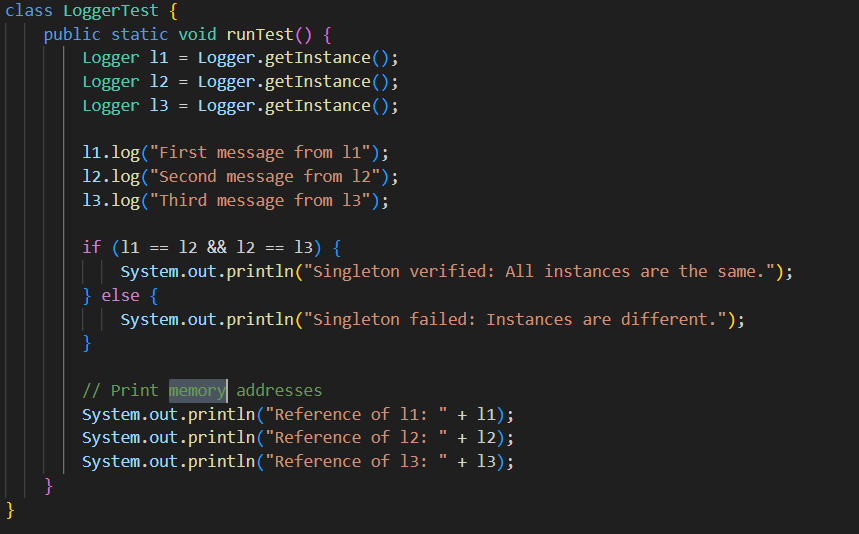
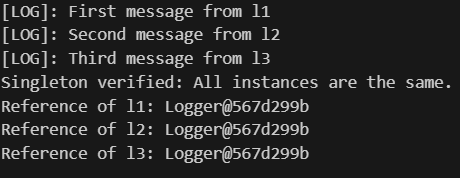
**Exercise 1: Implementing the Singleton Pattern**

**Steps:**

1. **Create a New Java Project:**
   * A new project named **SingletonPattern** is created as shown below.



1. **Define a Singleton Class:**
   * Created a class named Logger that has a private static instance of itself.
   * Ensured the constructor of Logger is private.
   * Provide a public static method to get the instance of the Logger class.
2. **Implement the Singleton Pattern:**
   * Written code to ensure that the Logger class follows the Singleton design pattern as shown below.
3. **Test the Singleton Implementation:**
   * Create a test class to verify that only one instance of Logger is created and used across the application.

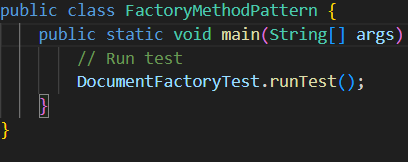
****

1. **Output for the Test Class Implementation.**

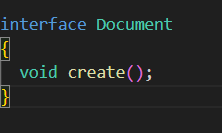
**Exercise 2: Implementing the Factory Method Pattern**

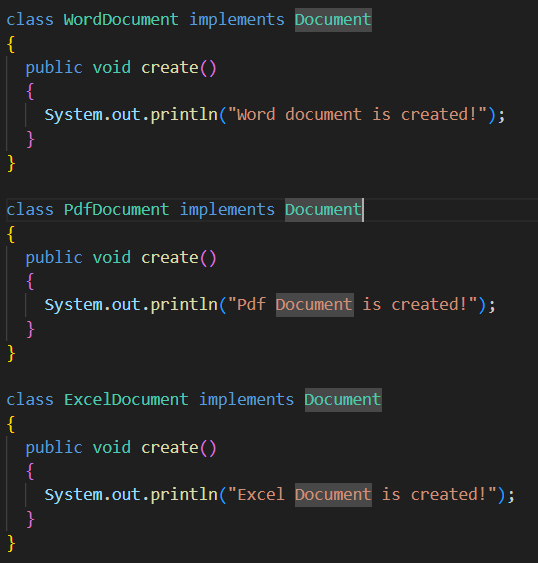
**Steps:**

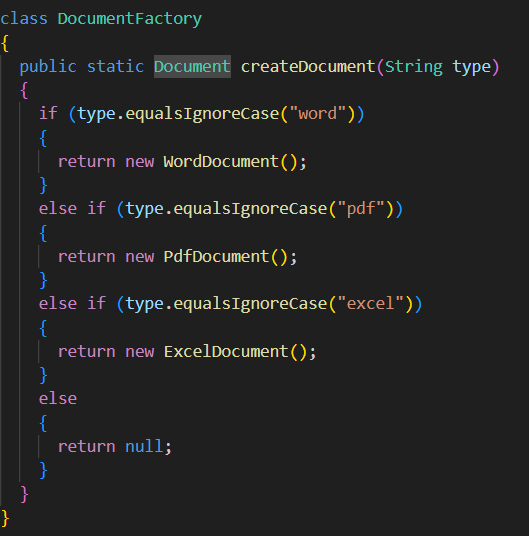
1. **Create a New Java Project:**
   * Created a new Java project named **FactoryMethodPattern**.

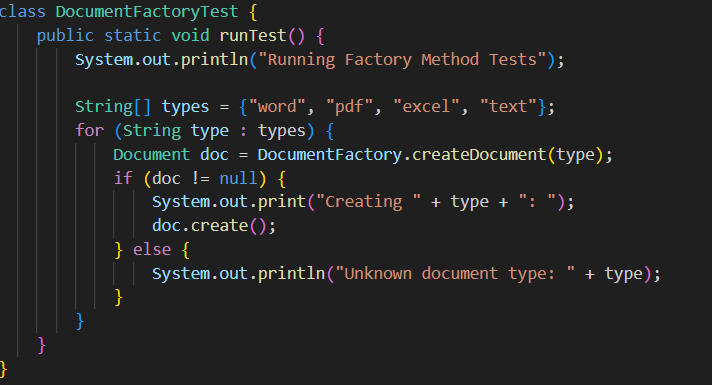


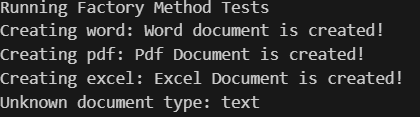
1. **Define Document Classes:**
   * Created an interface or abstract classes for different document types such as **WordDocument**, **PdfDocument**, and **ExcelDocument**.



1. **Create Concrete Document Classes:**
   * Implement concrete classes for each document type that implements or extends the above interfaces or abstract classes.
2. **Implement the Factory Method:**
   * Created an abstract class **DocumentFactory** with a method **createDocument()**.
   * Create concrete factory classes for each document type that extends DocumentFactory and implements the **createDocument()** method.

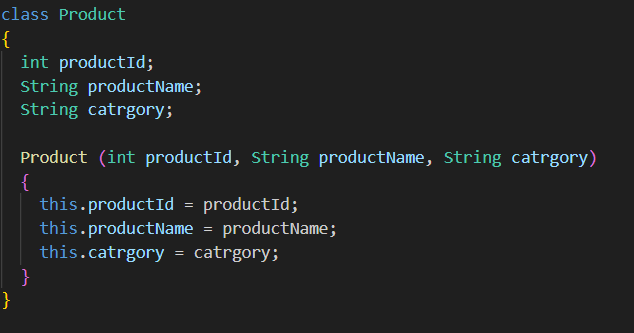


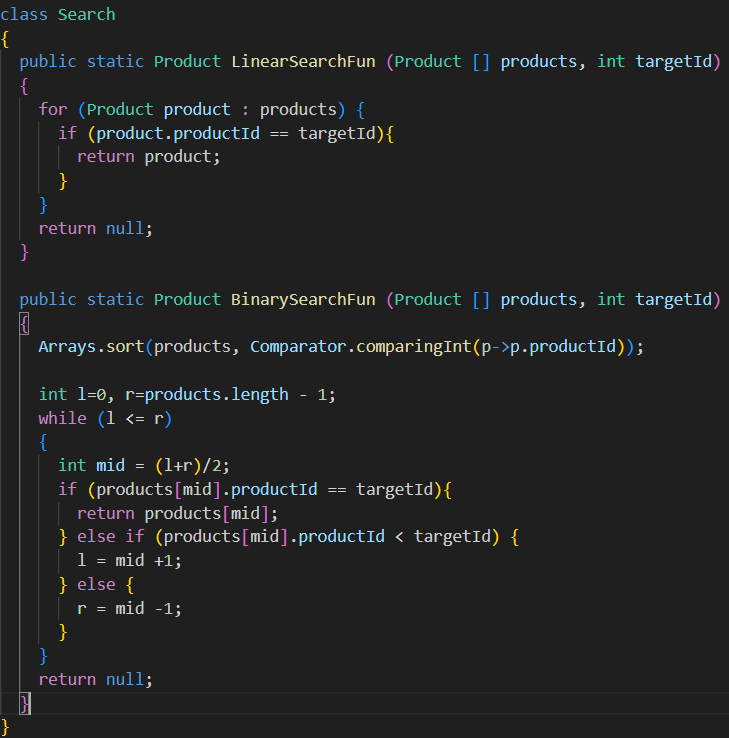
1. **Test the Factory Method Implementation:**
   * Create a test class to demonstrate the creation of different document types using the factory method.



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

**Steps:**

1. **Understand Asymptotic Notation:**
   * **Explain Big O notation and how it helps in analyzing algorithms**. - Big O notation describes how the runtime of an algorithm grows. It helps to compare the efficiency of the programs.
   * **Describe the best, average, and worst-case scenarios for search operations**. – For Search operations, the best, average, and worst-case scenarios for a normal search operation would be
     1. **O(1)** – The best case,
     2. **O(n/2)** – The average case (approximately equals O(n)),
     3. **O(n)** – The worst case.
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.



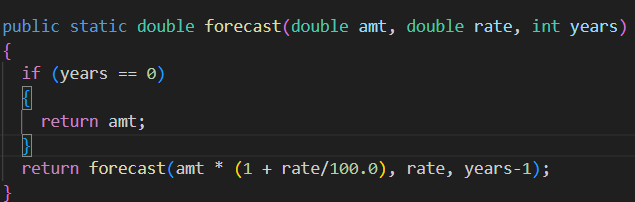
1. **Analysis:**
   * **Compare the time complexity of linear and binary search algorithms. –** 
     1. **Linear search -** O(n) (Checks items one by one).
     2. **Binary search –** O(log n) (divides the array in half each step).
   * **Discuss which algorithm is more suitable for your platform and why. –** 
     1. **For small or unsorted datasets**, linear search is a simple and effective approach.
     2. **For large and sorted datasets,** Binary search is suitable as it reduces comparisons.

Hence, **Binary Search** is more suitable for an E-commerce Platform.

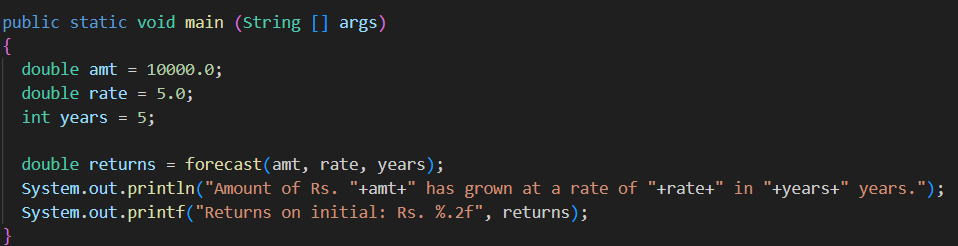
**Exercise 7: Financial Forecasting**

**Steps:**

1. **Understand Recursive Algorithms:**
   * **Explain the concept of recursion and how it can simplify certain problems. -** Recursion is a technique in which a method calls itself to solve a problem. Each recursive call works on a smaller sub-part until a base condition is met. Recursion simplifies complex problems by breaking them into subparts and solving them.
2. **Setup:**
   * Create a method to calculate the future value using a recursive approach.



1. **Implementation:**
   * Implement a recursive algorithm to predict future values based on past growth rates.





1. **Analysis:**
   * **Discuss the time complexity of your recursive algorithm. –** O(n) because it calculates one year at a time.
   * **Explain how to optimize the recursive solution to avoid excessive computation. –** For better optimization, we can cut off the recursion and use the formula and use of Math.pow() to get a better solution.