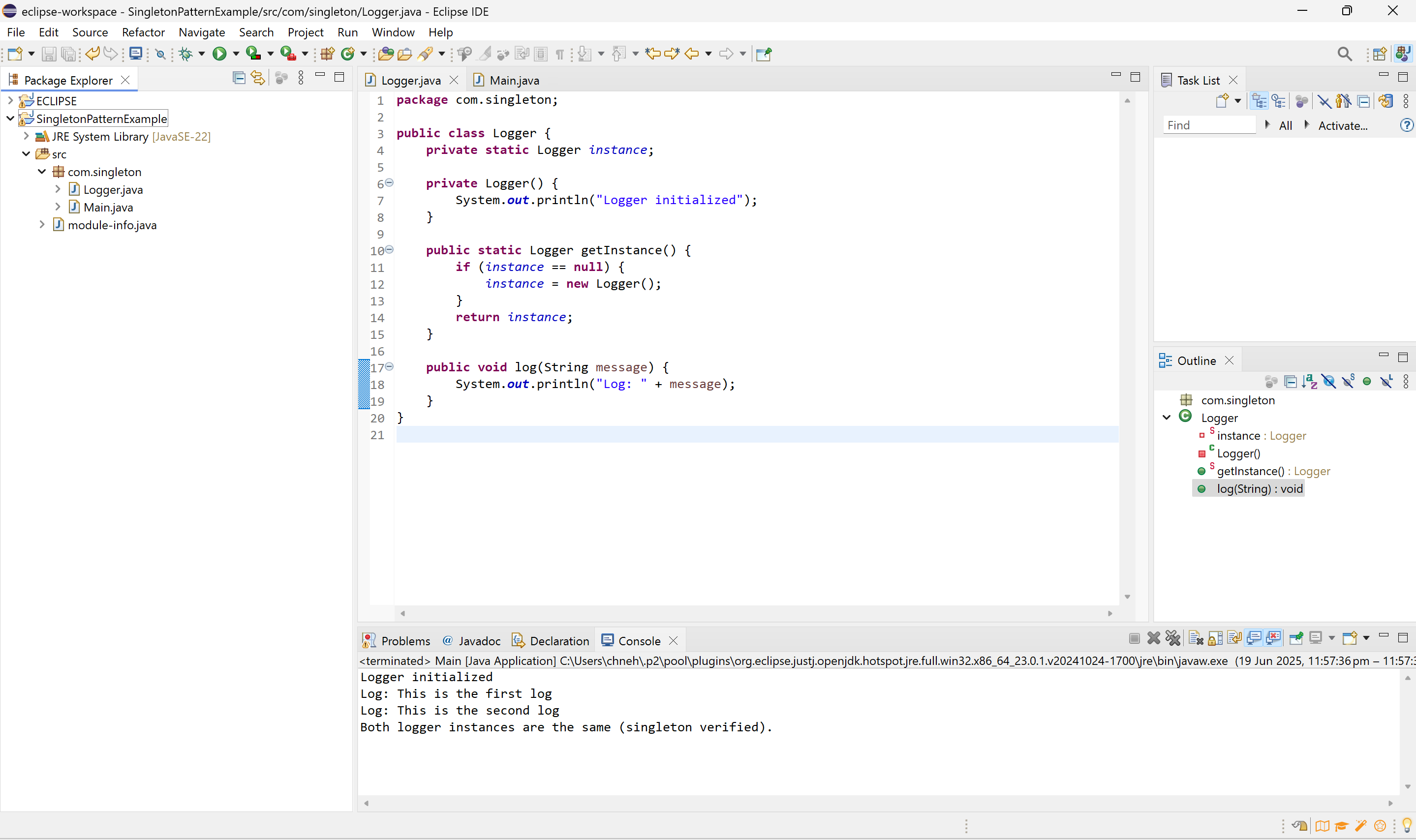
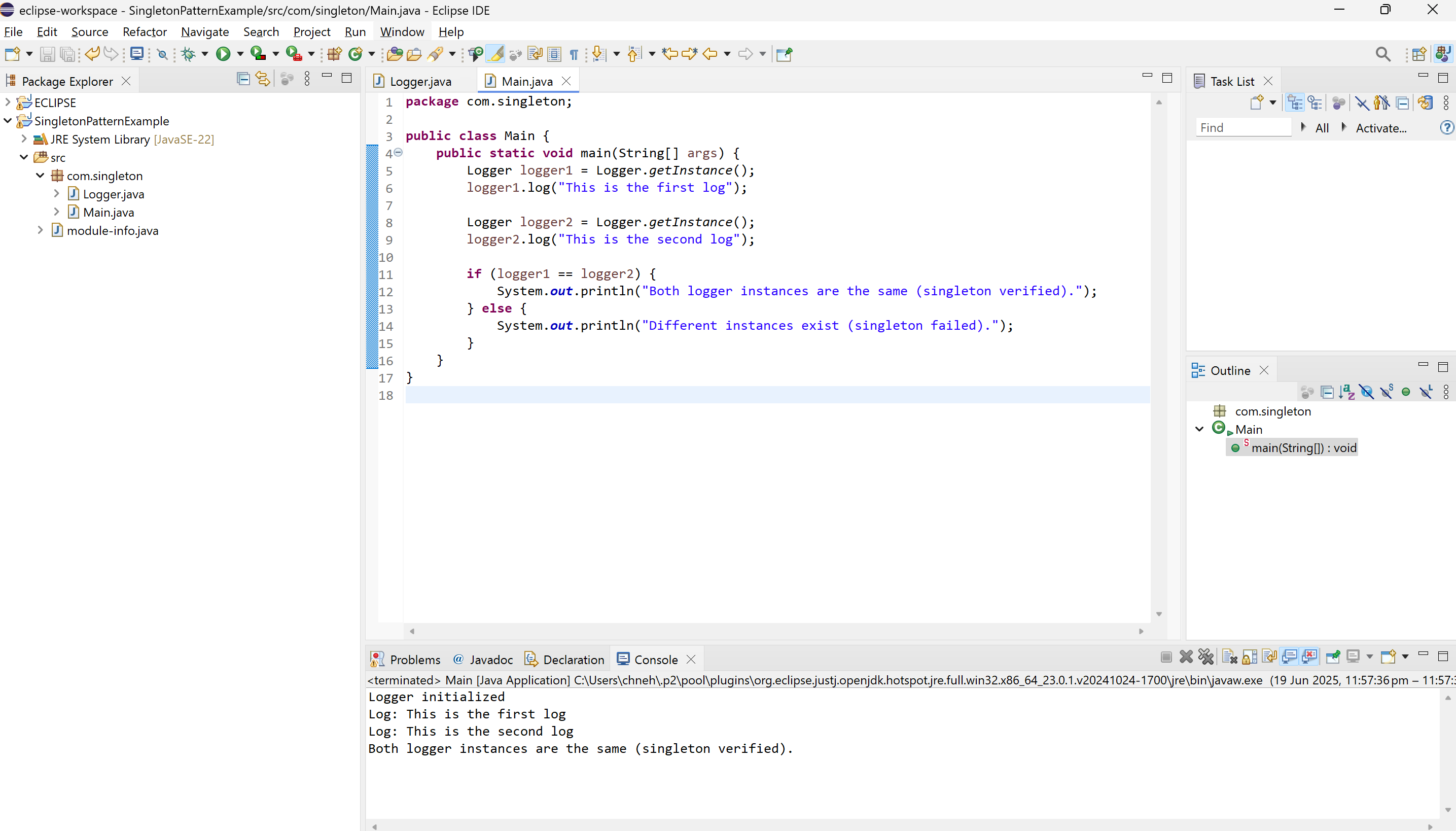
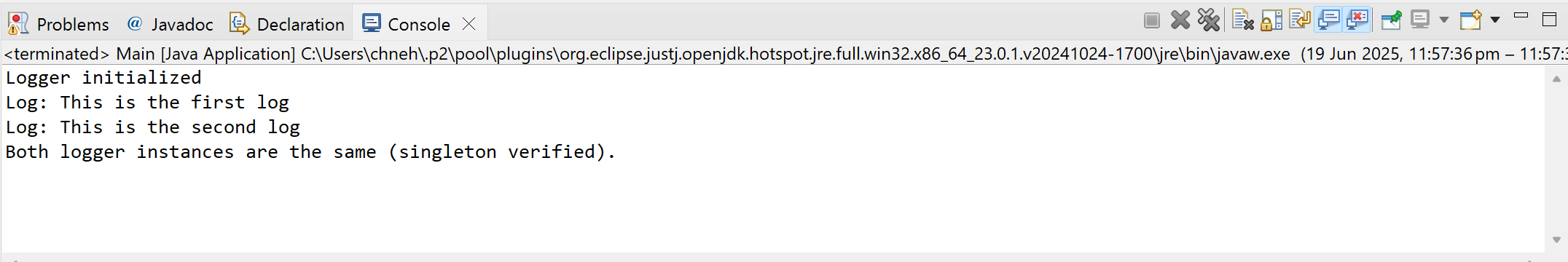
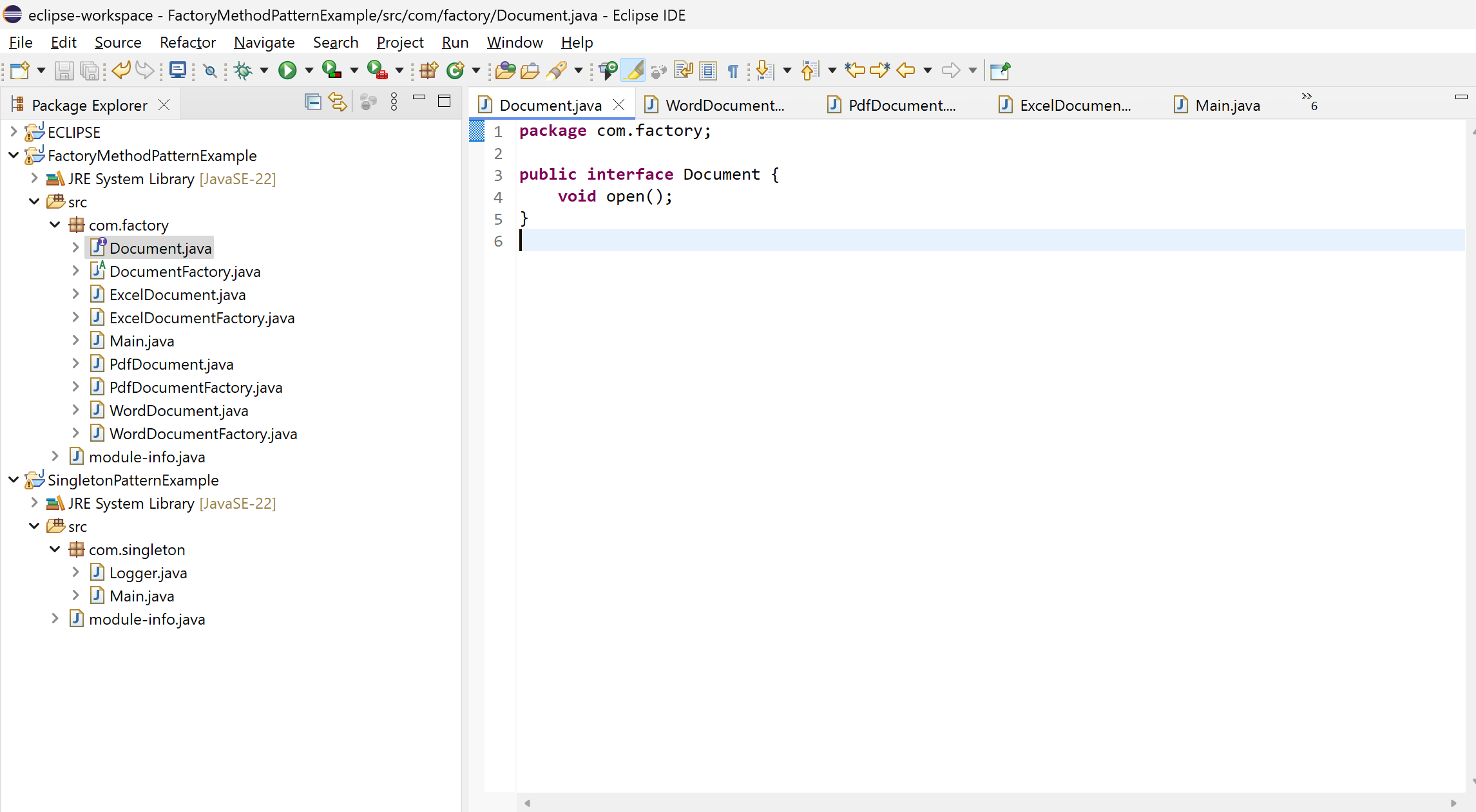
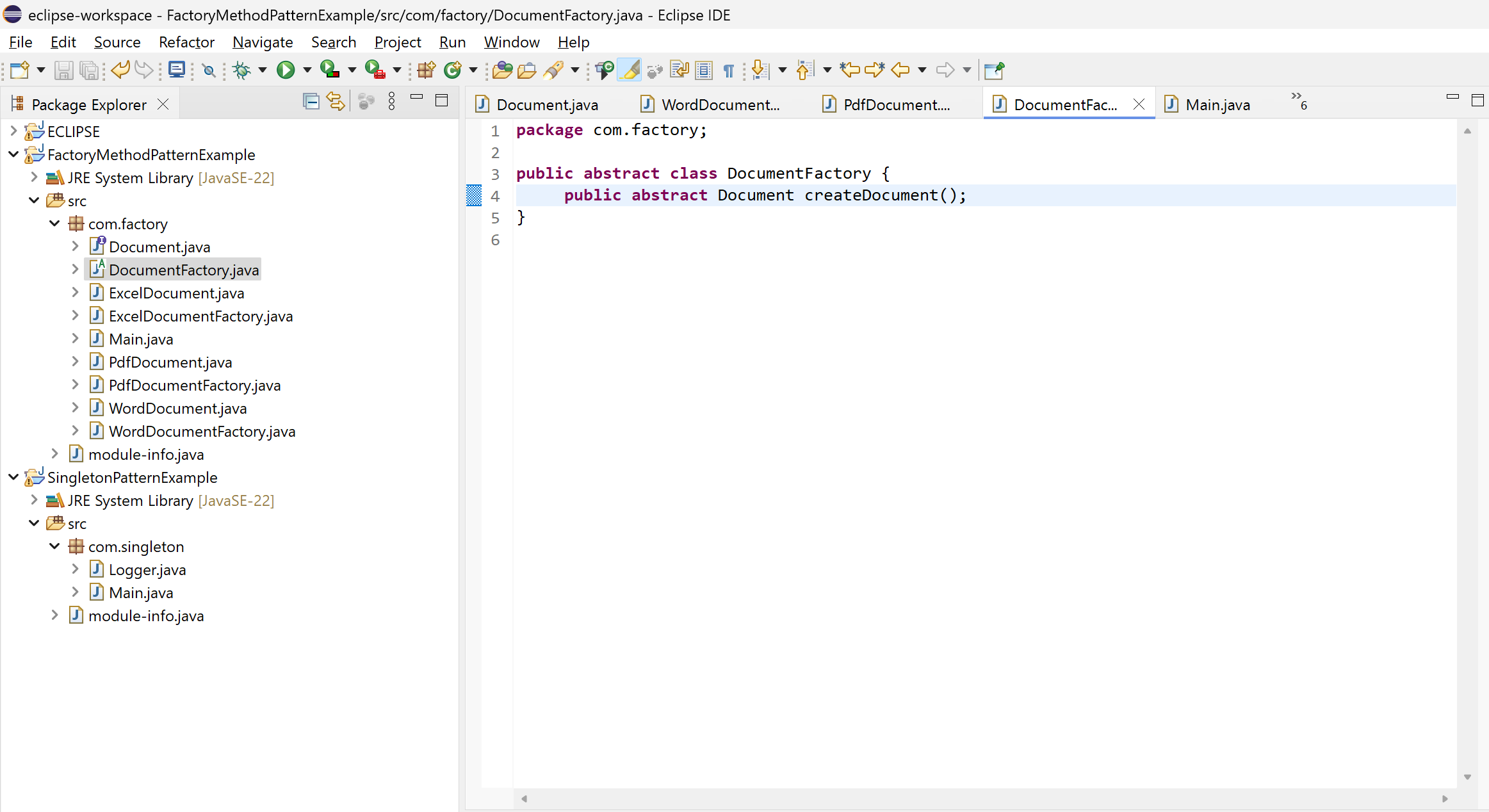
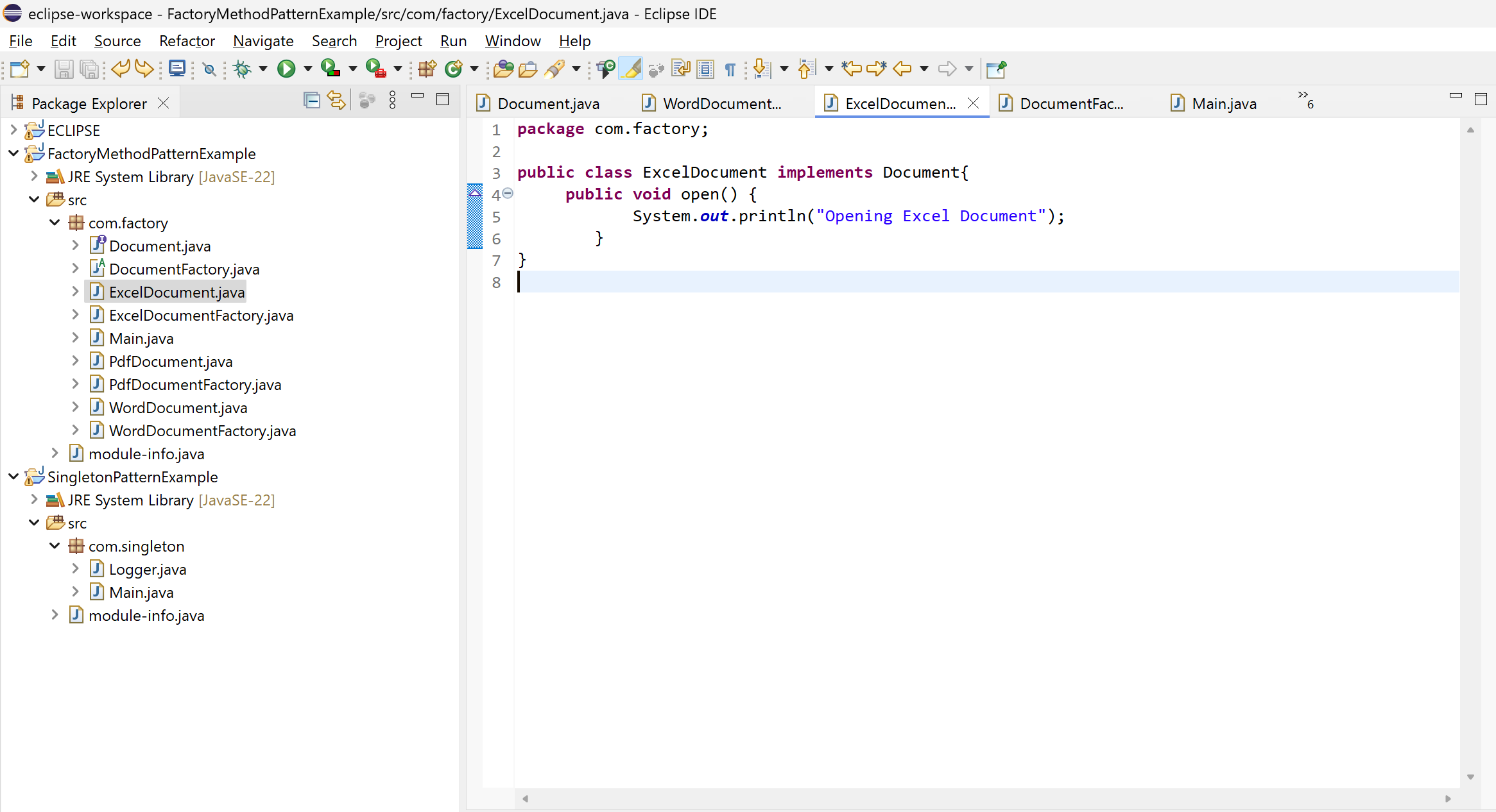
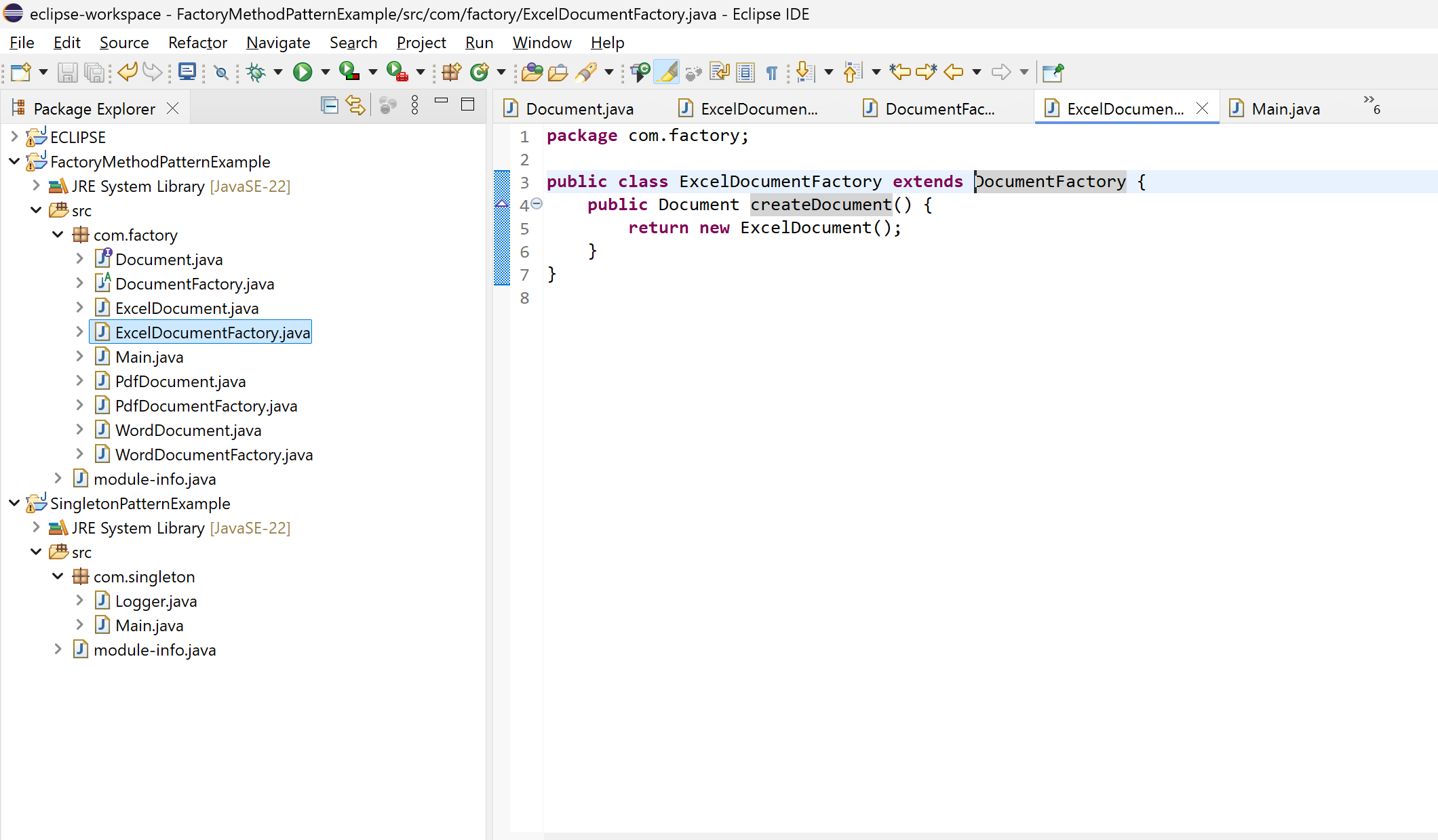
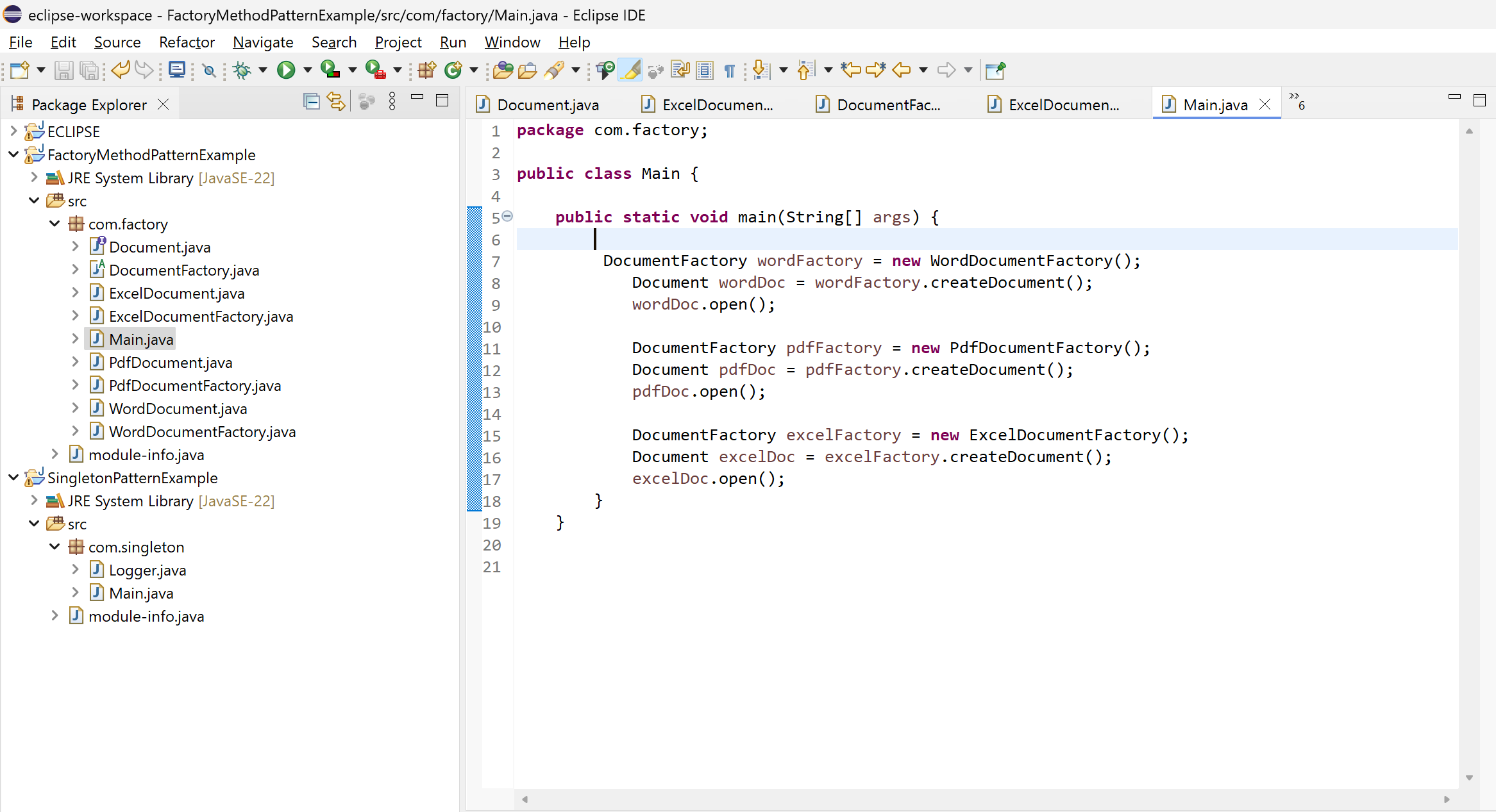
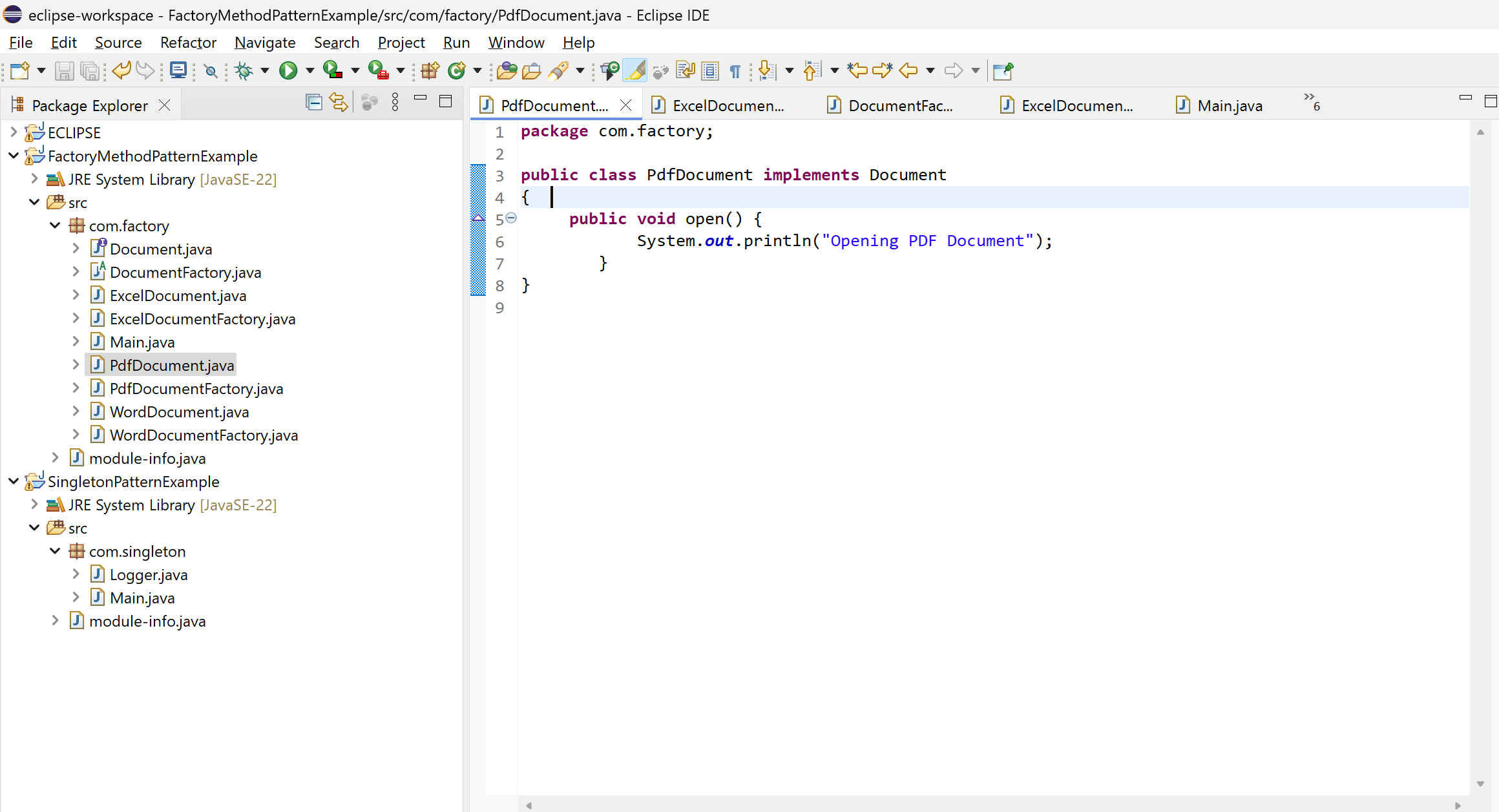
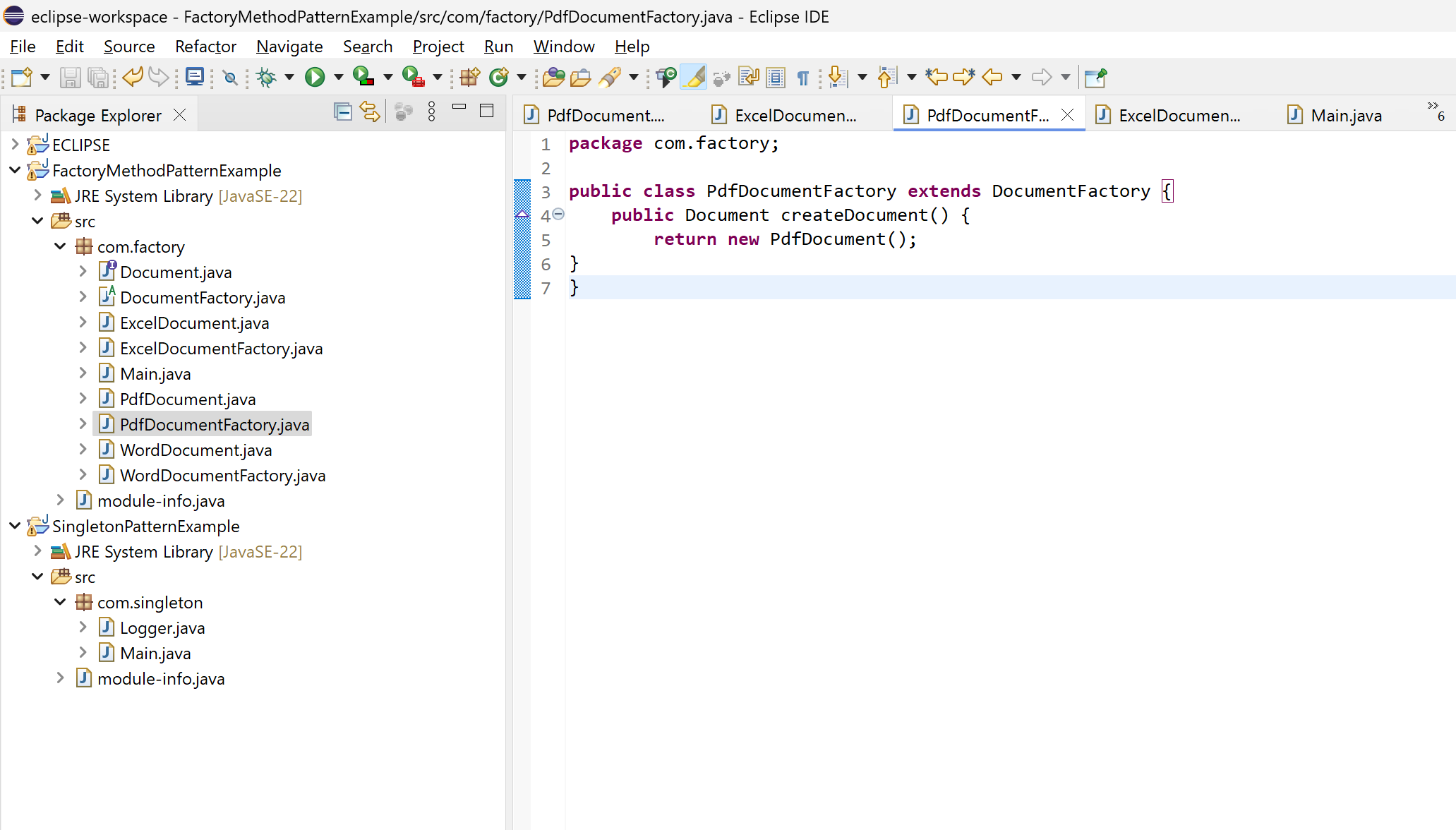
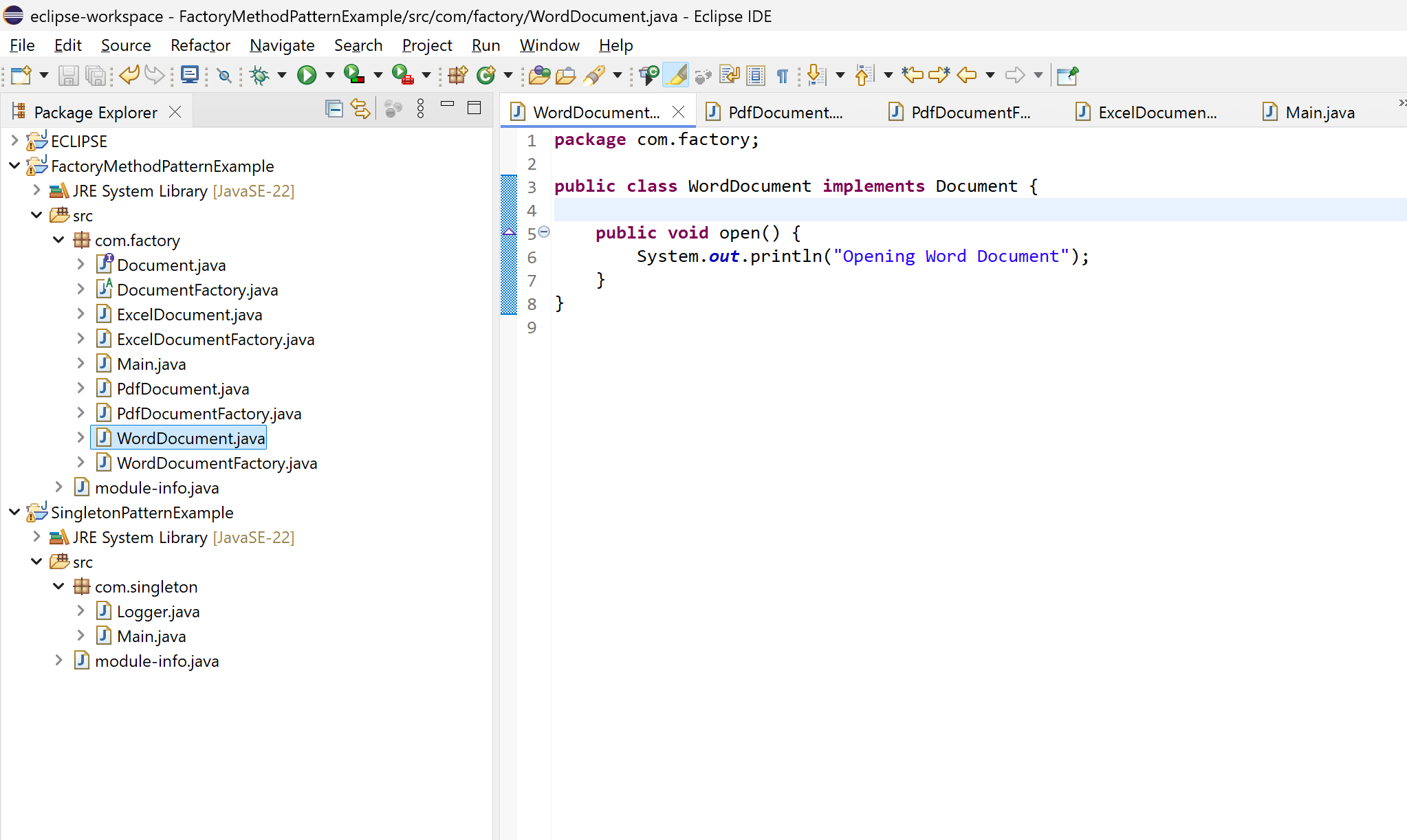
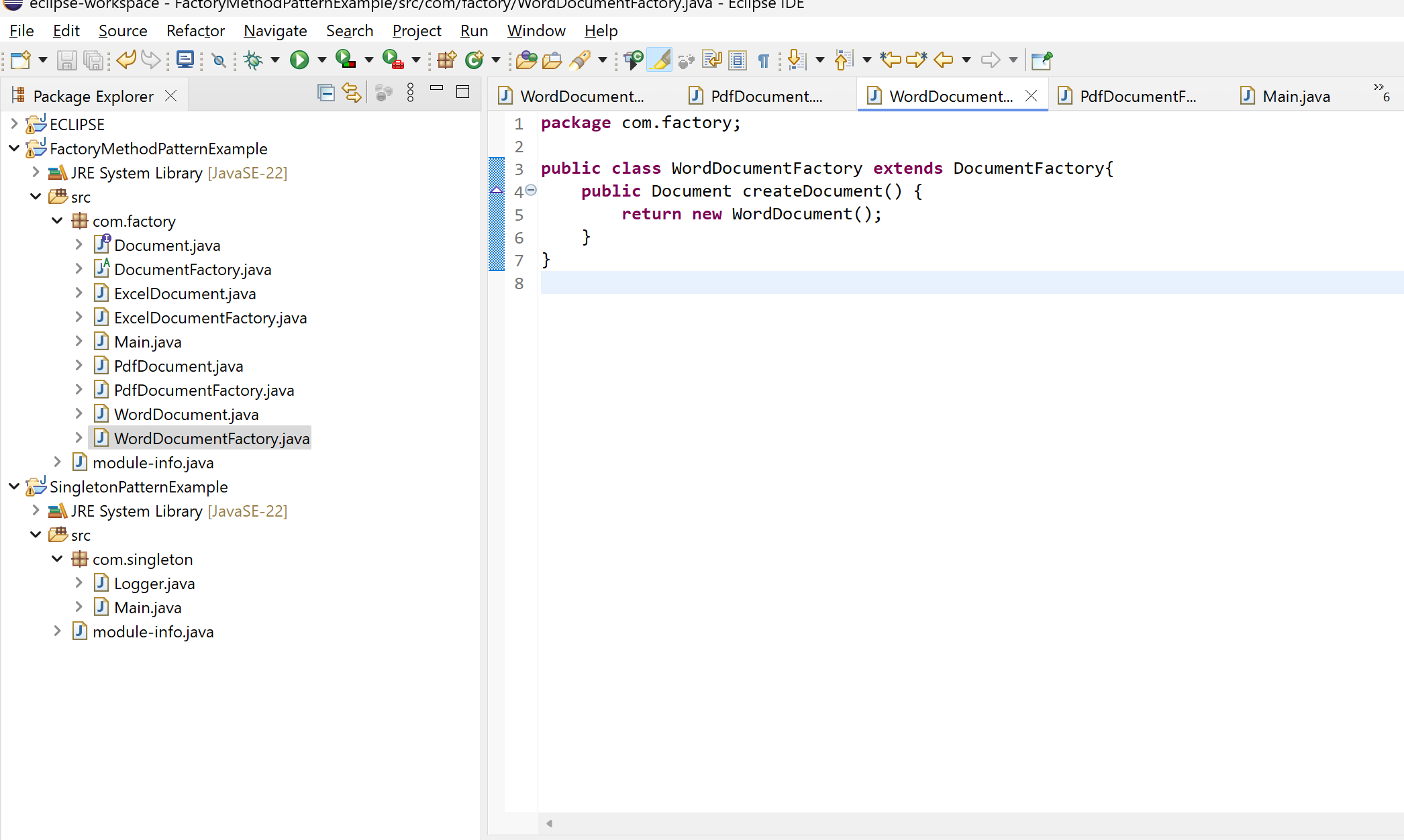
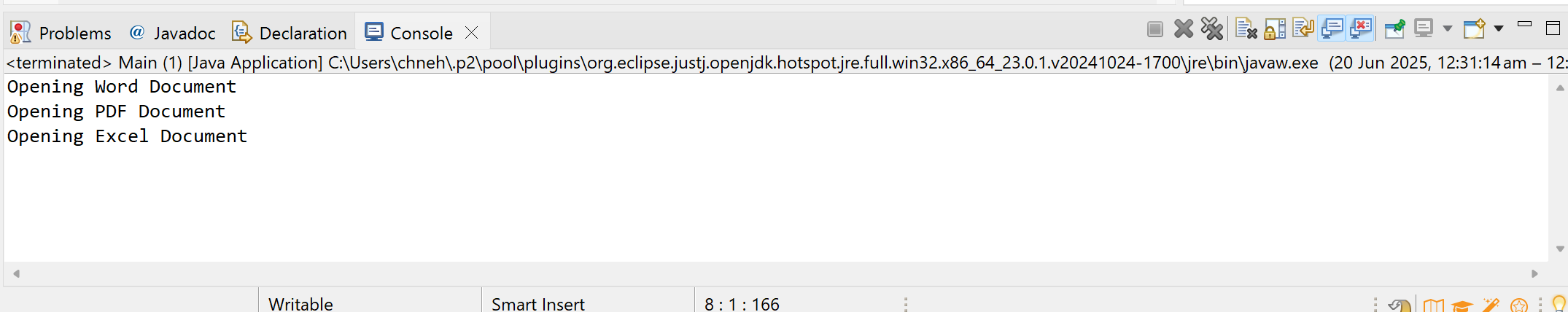
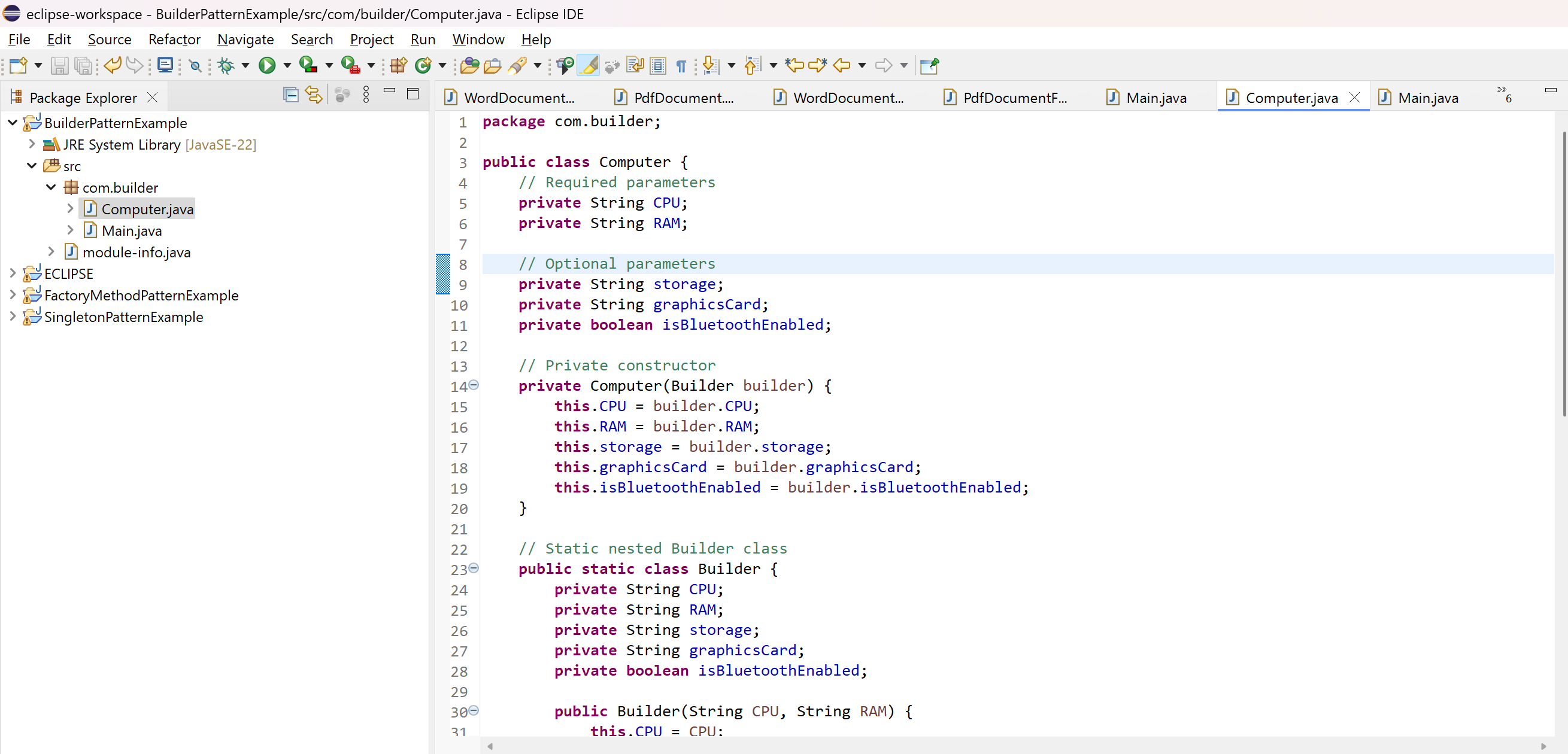
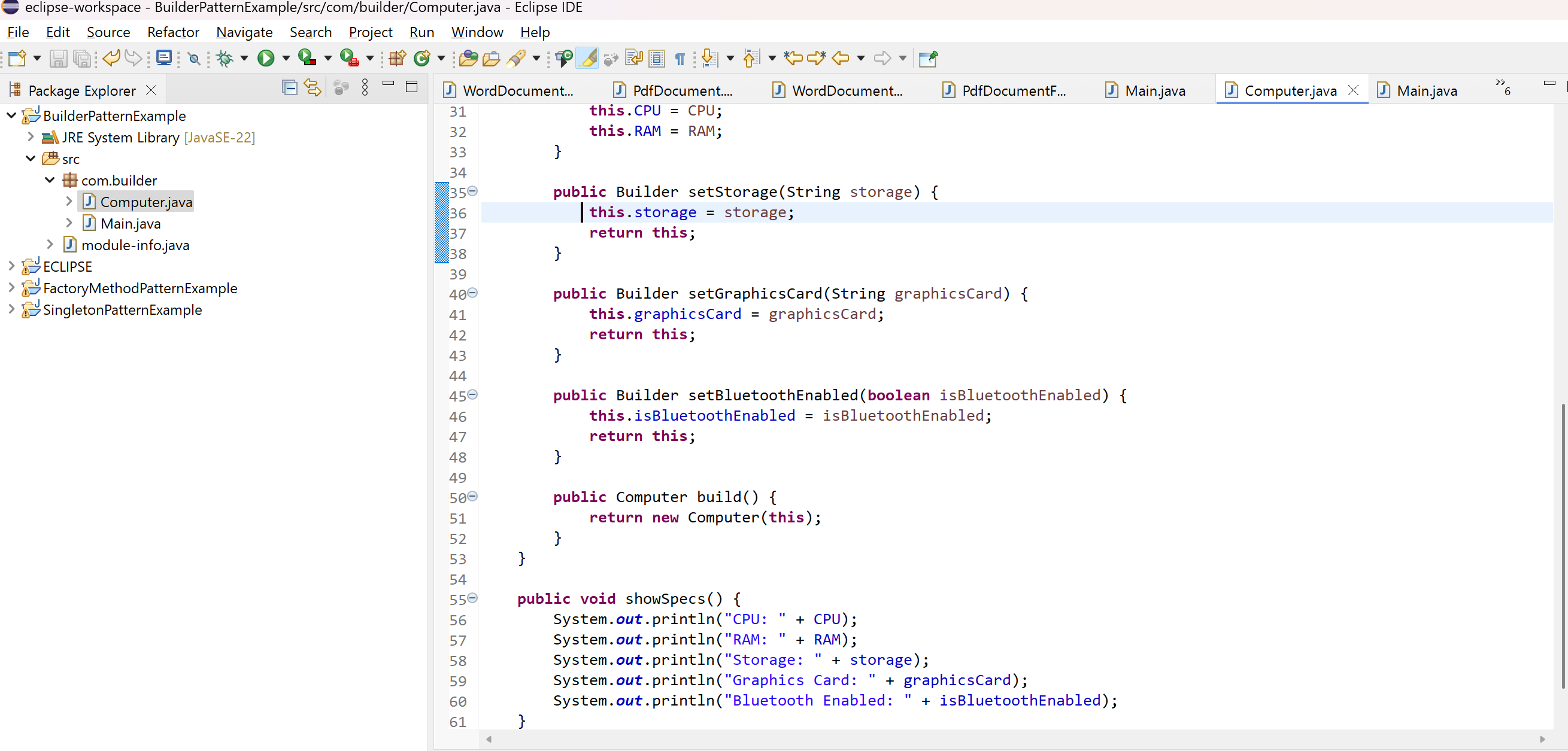
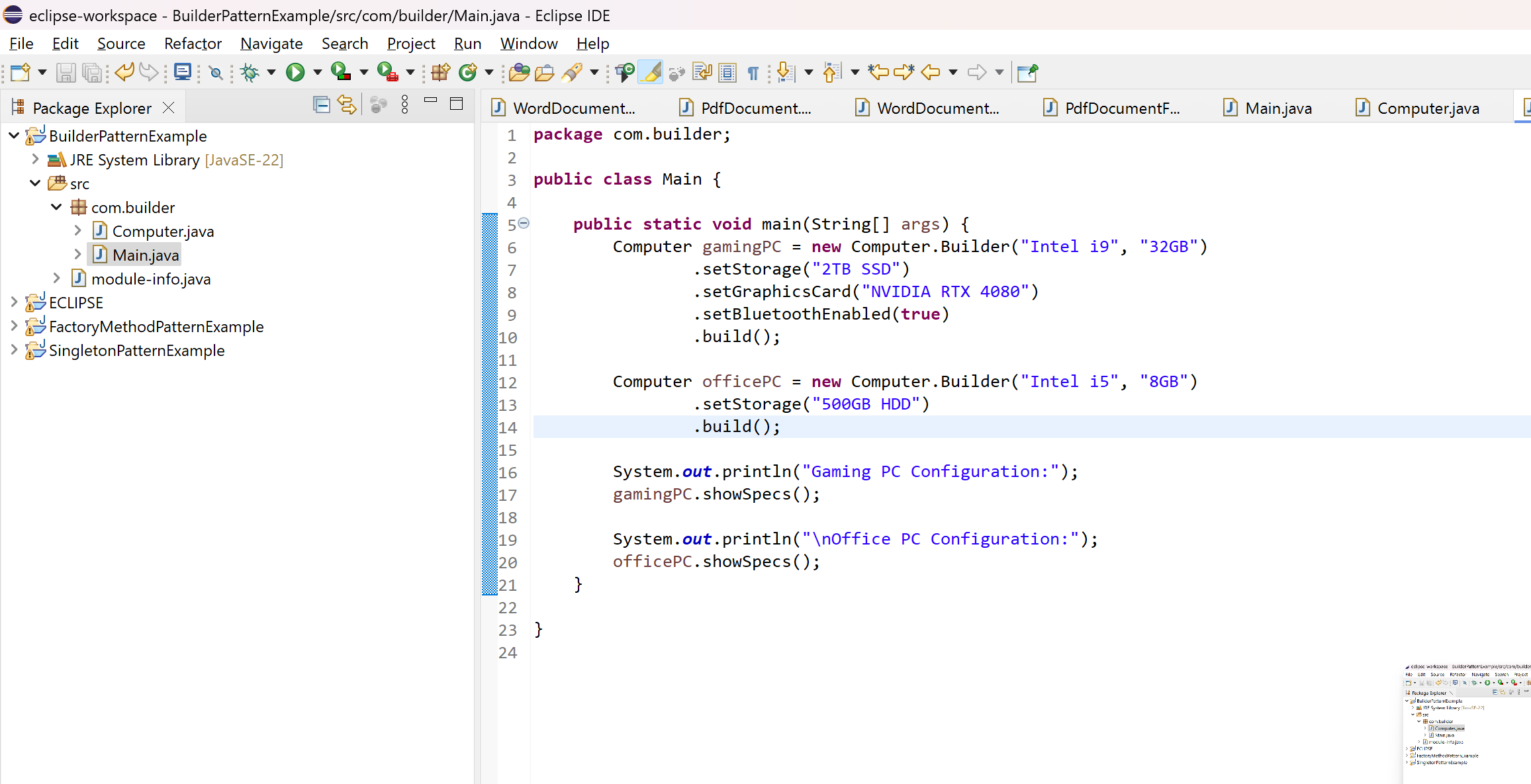
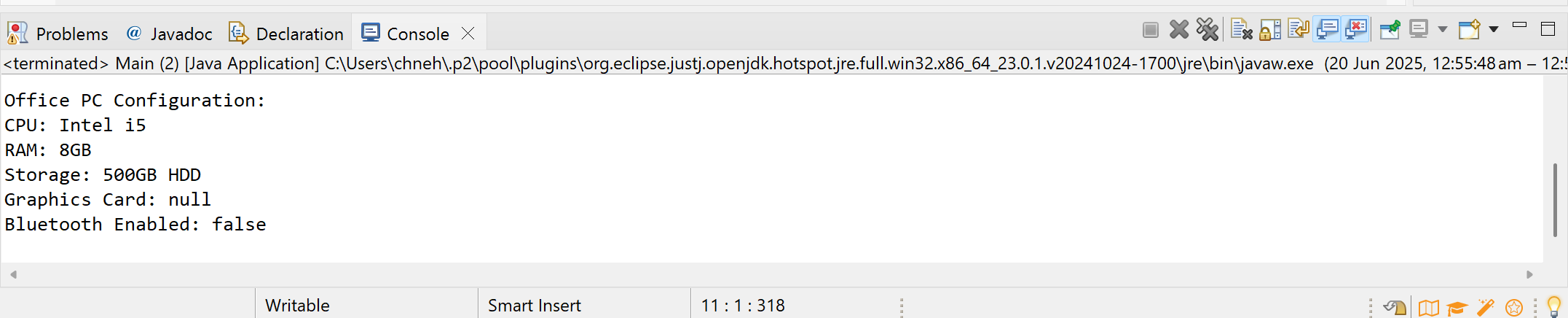
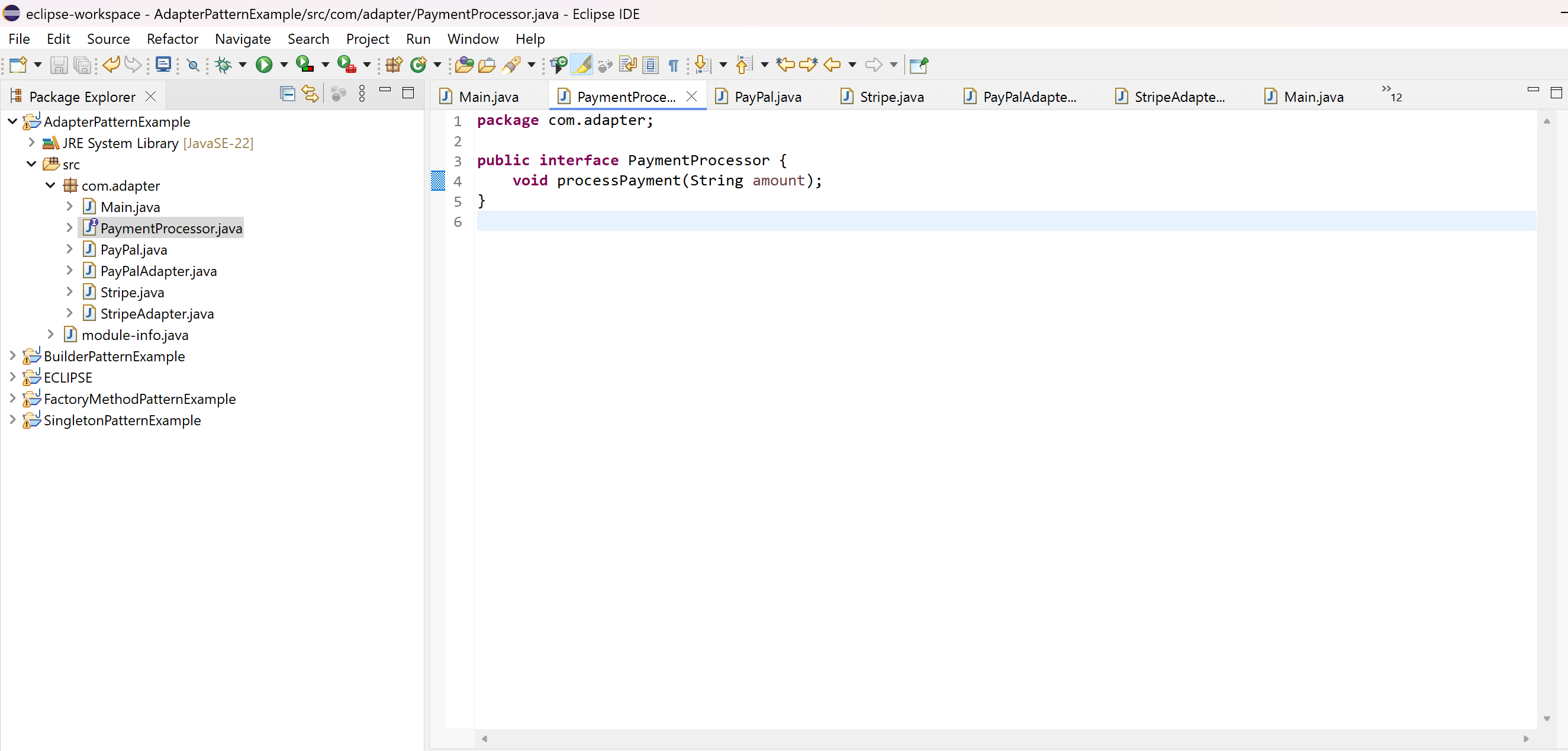
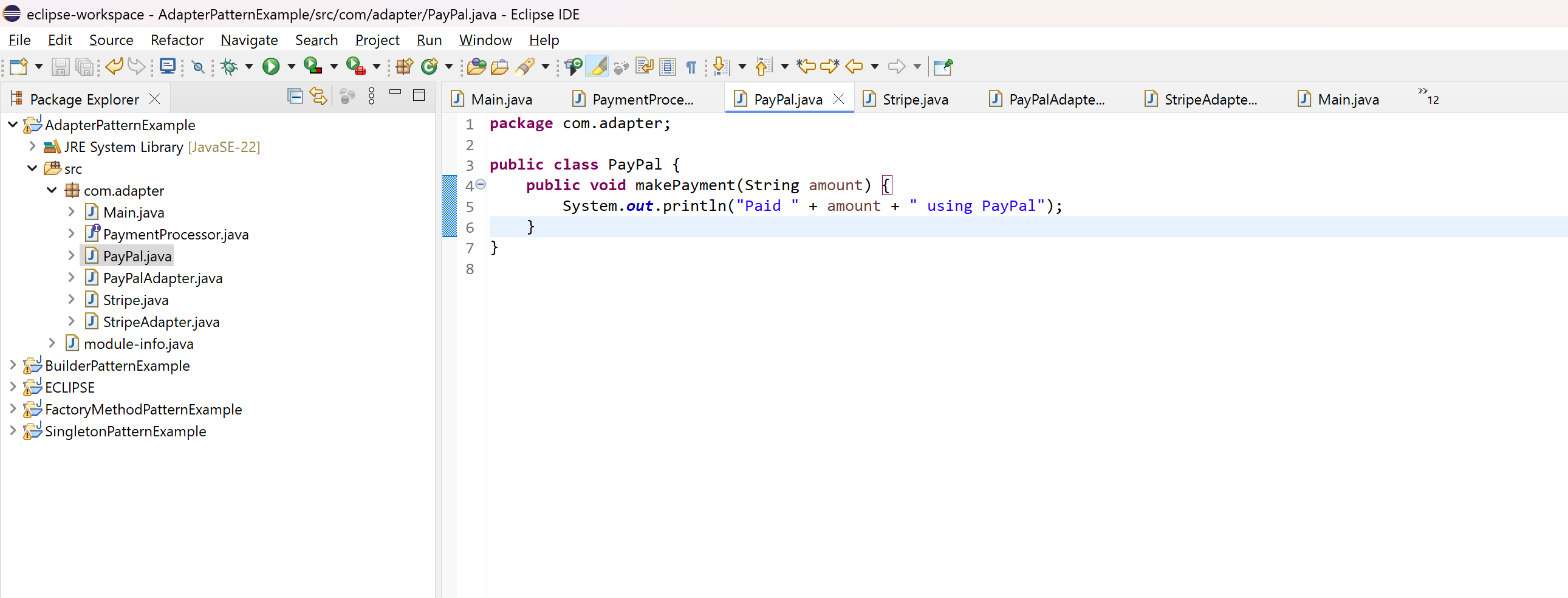
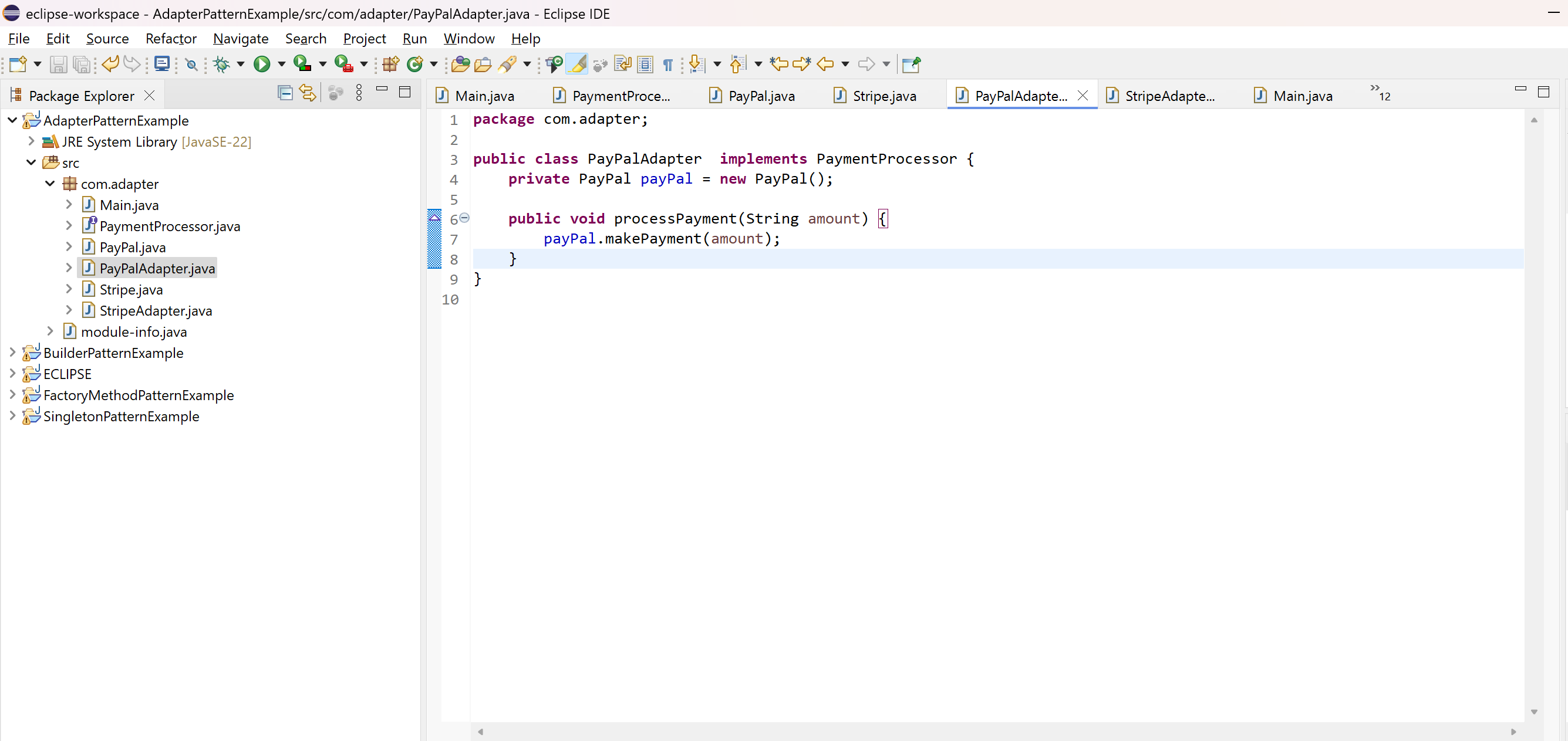
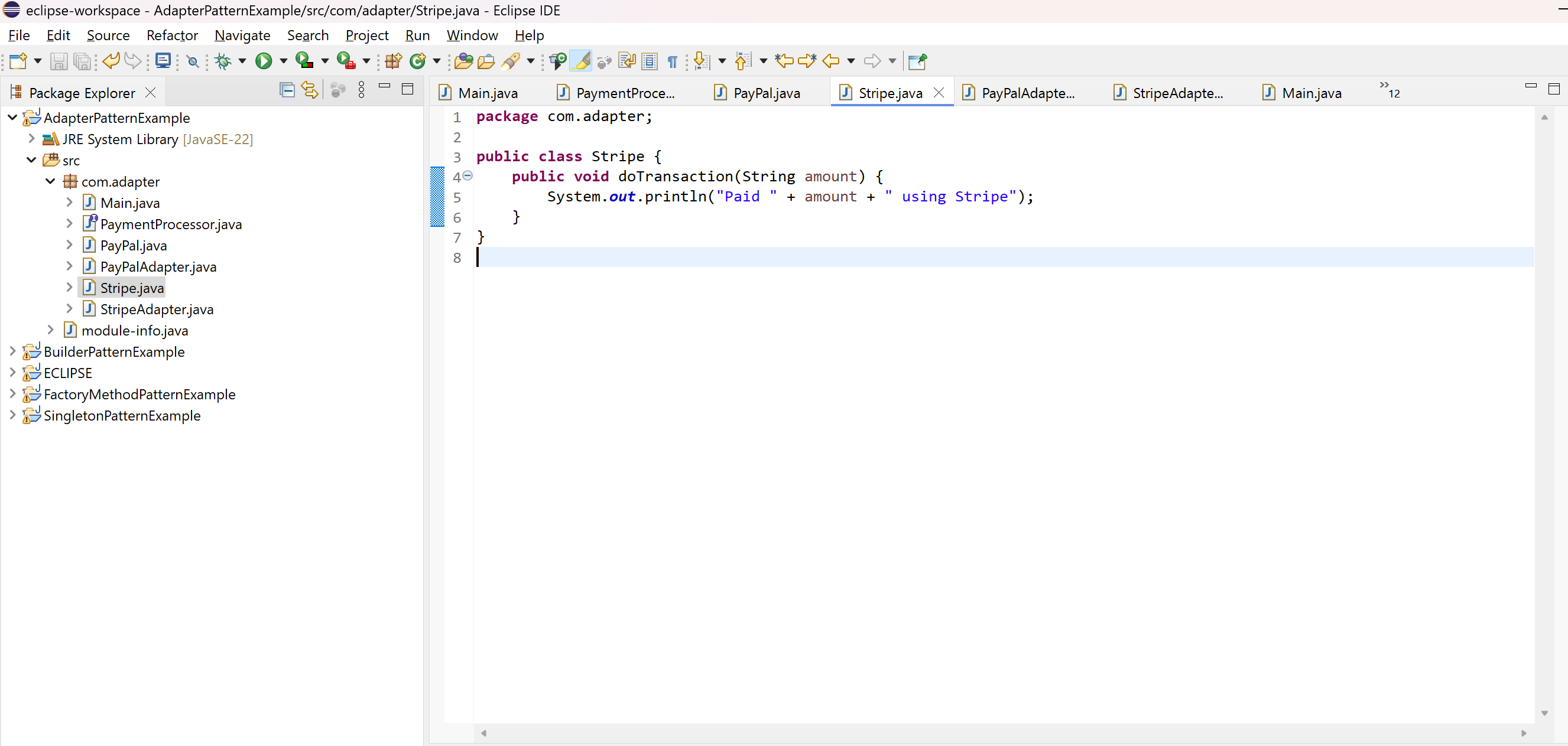
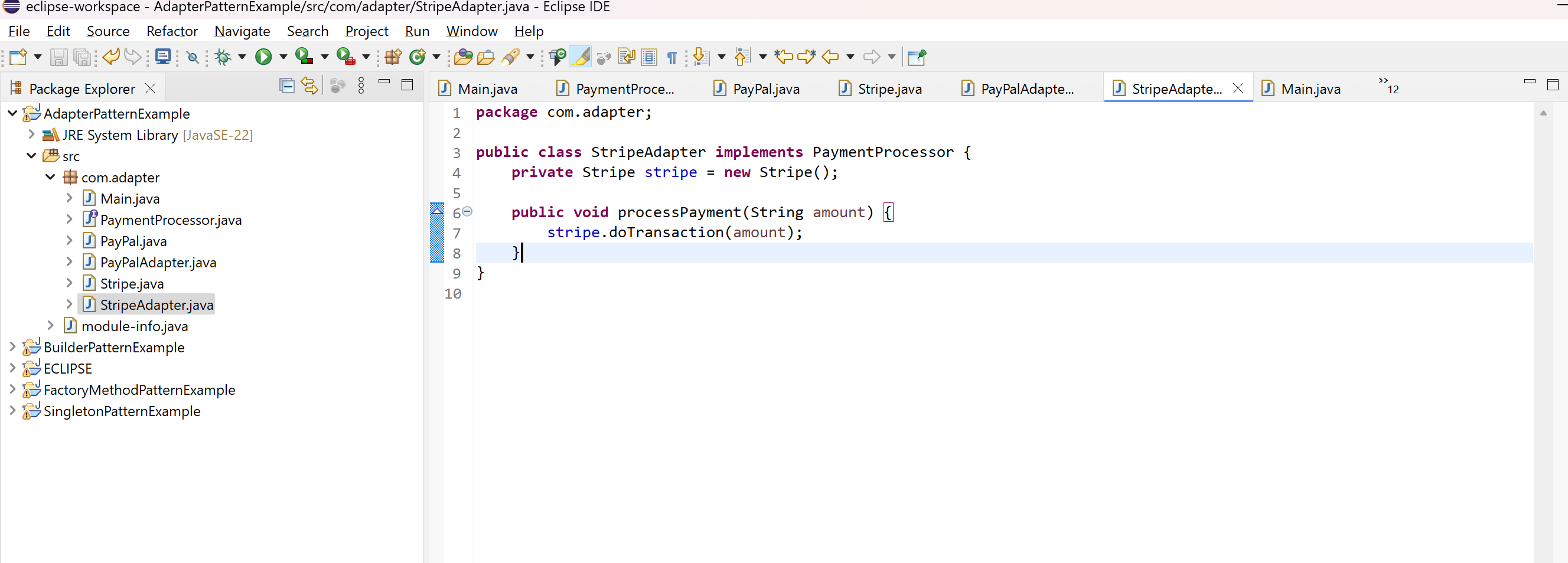
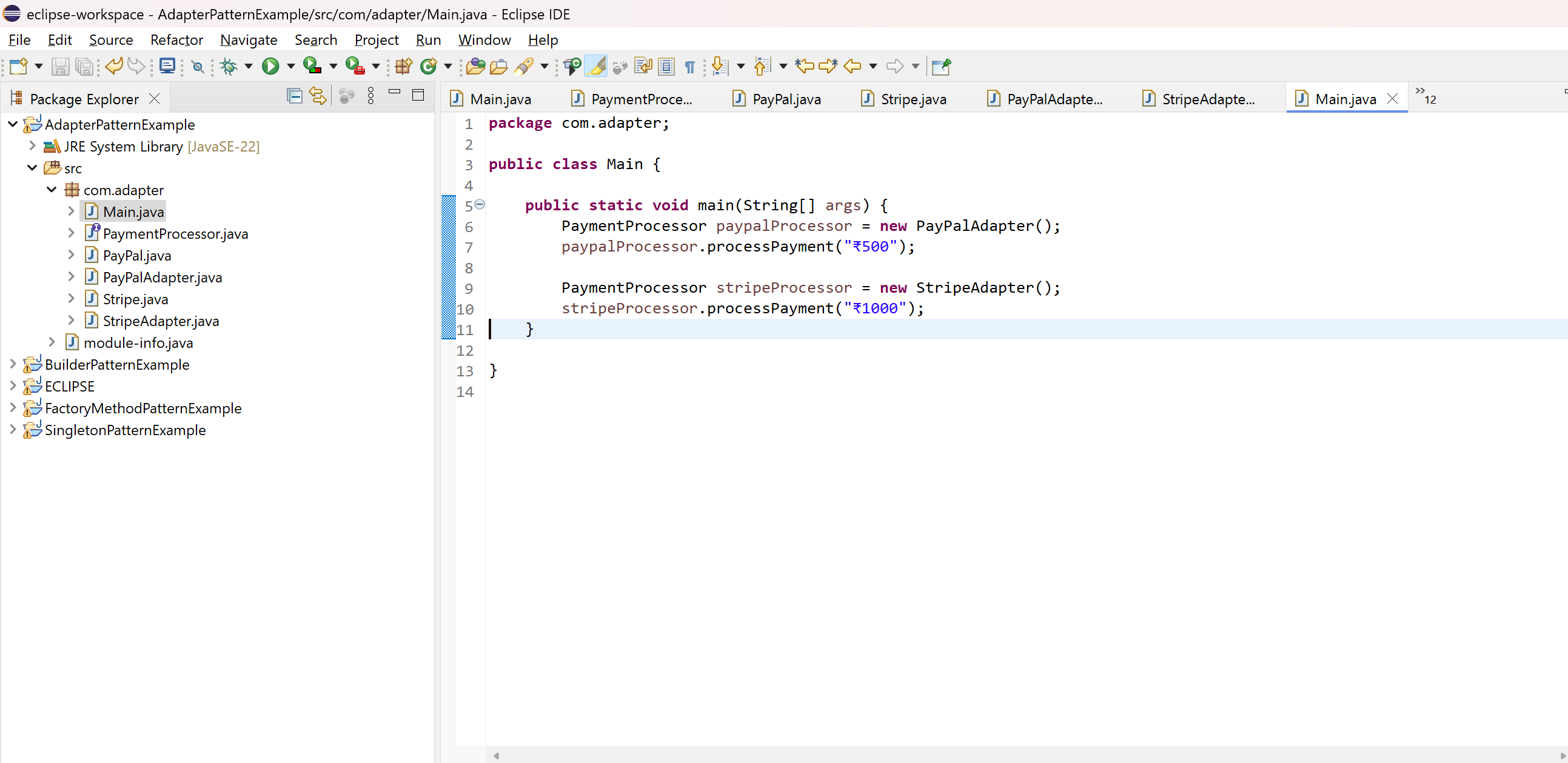
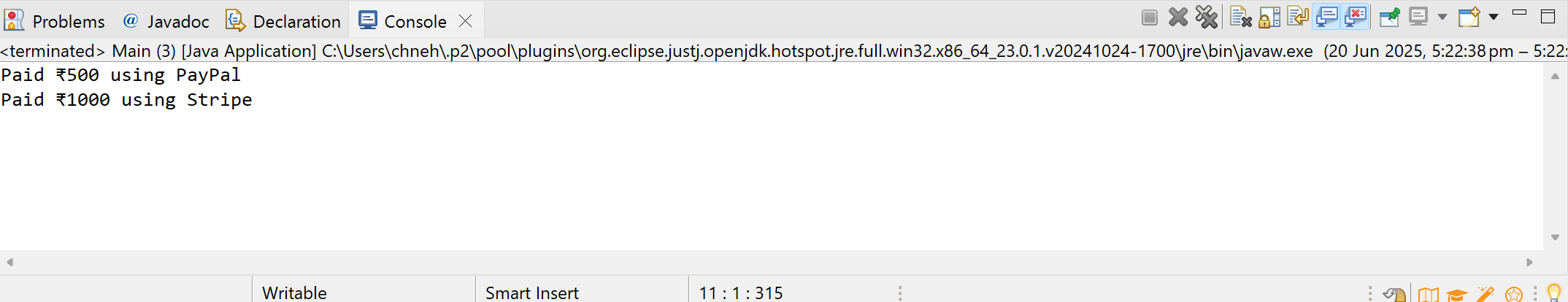
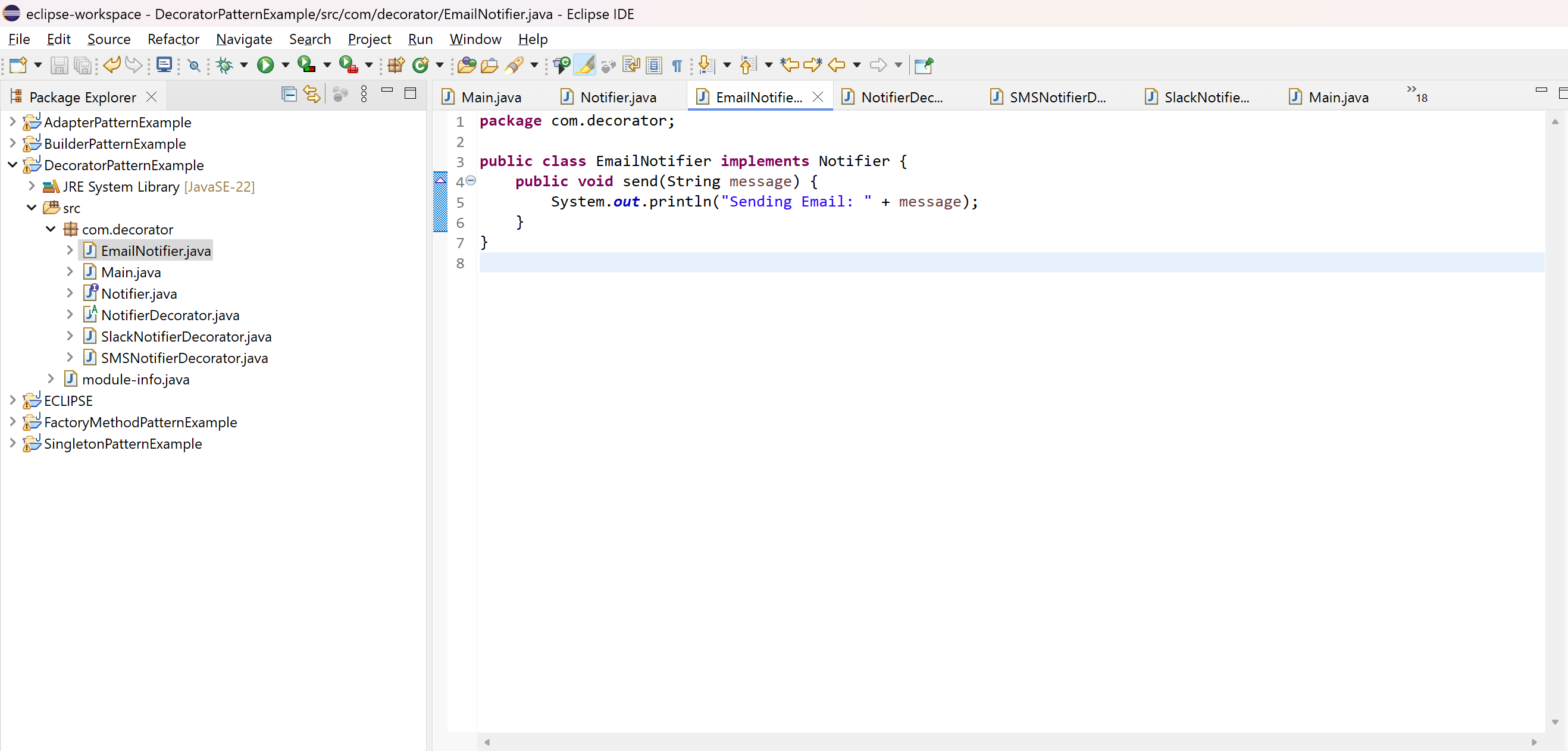
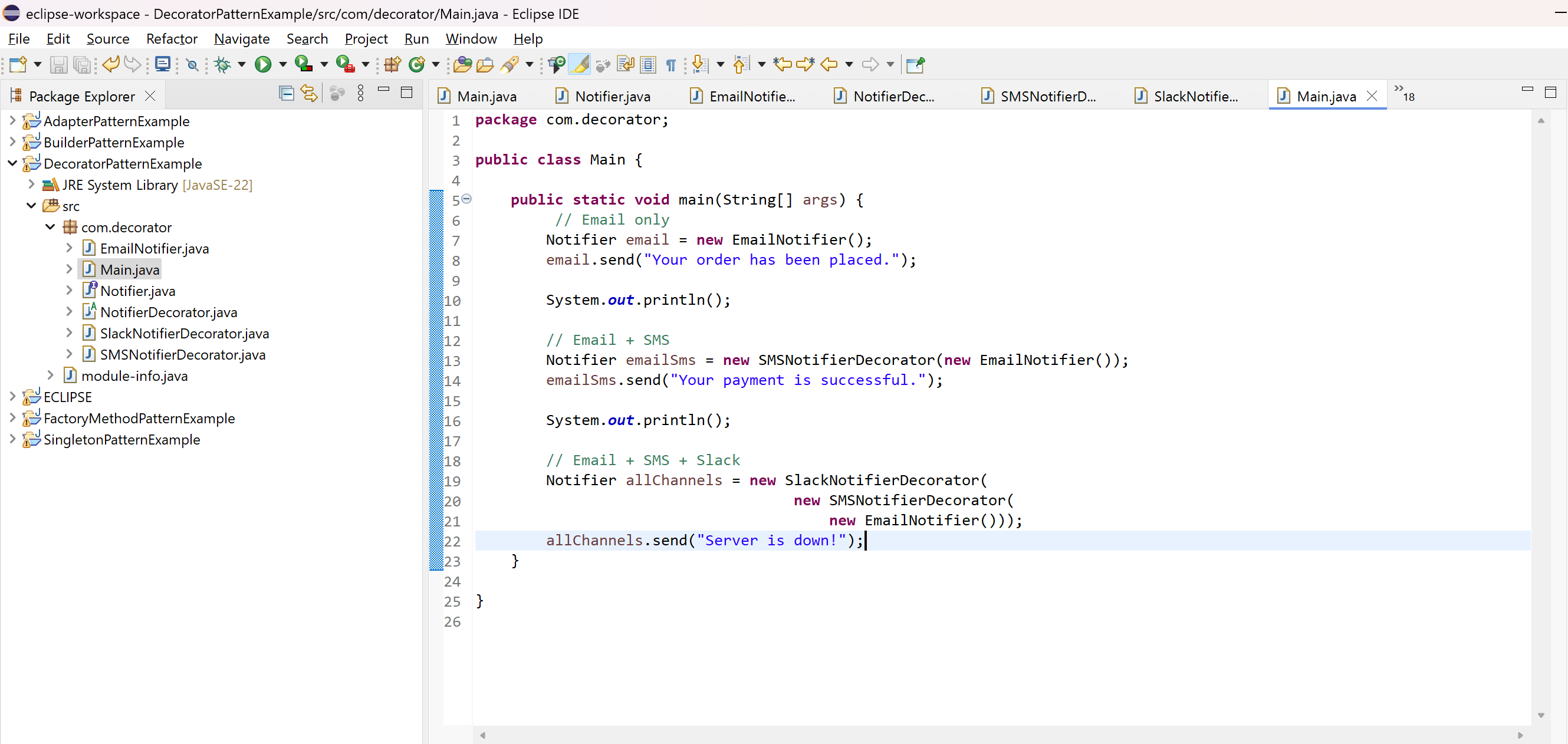
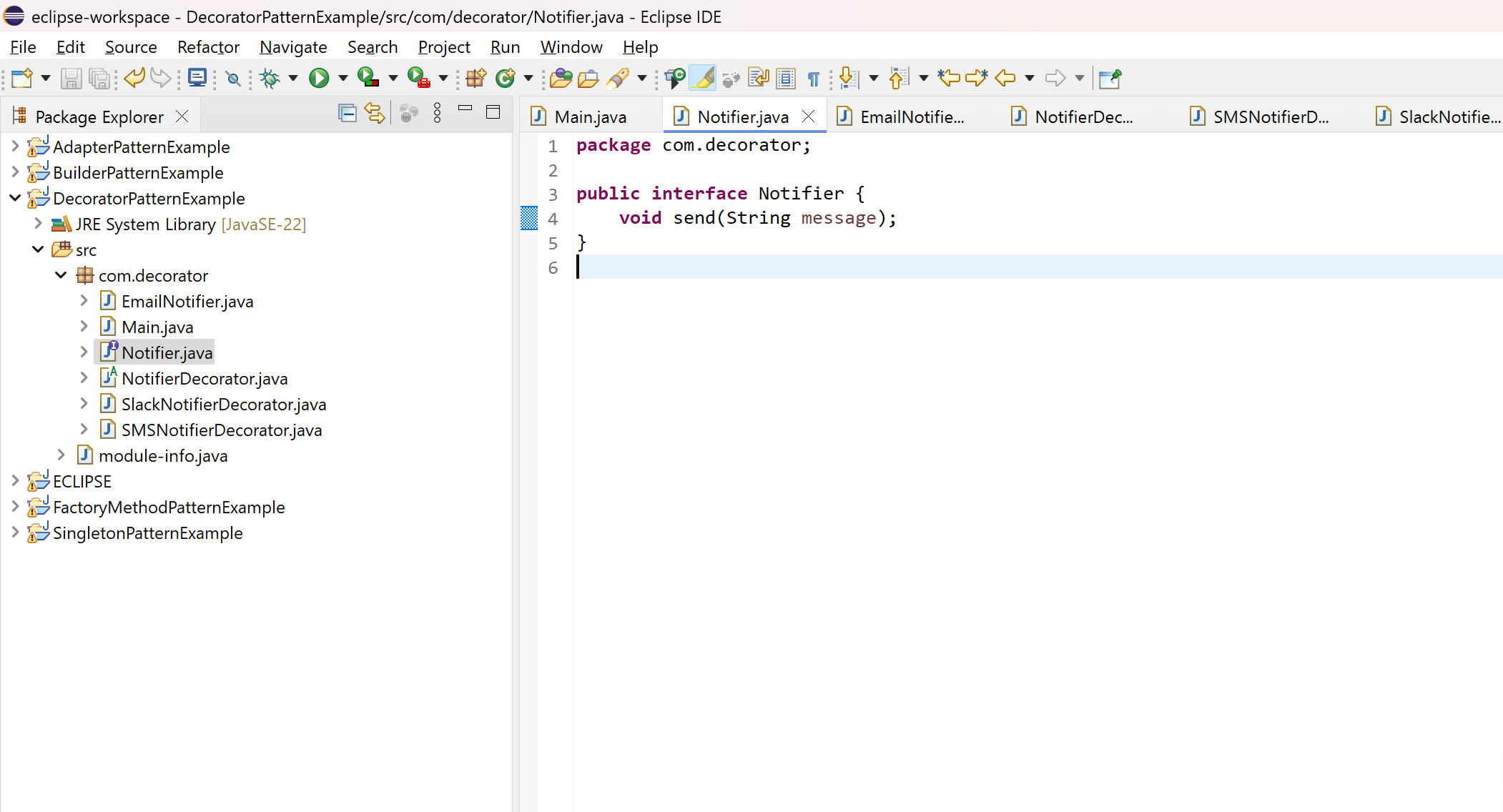
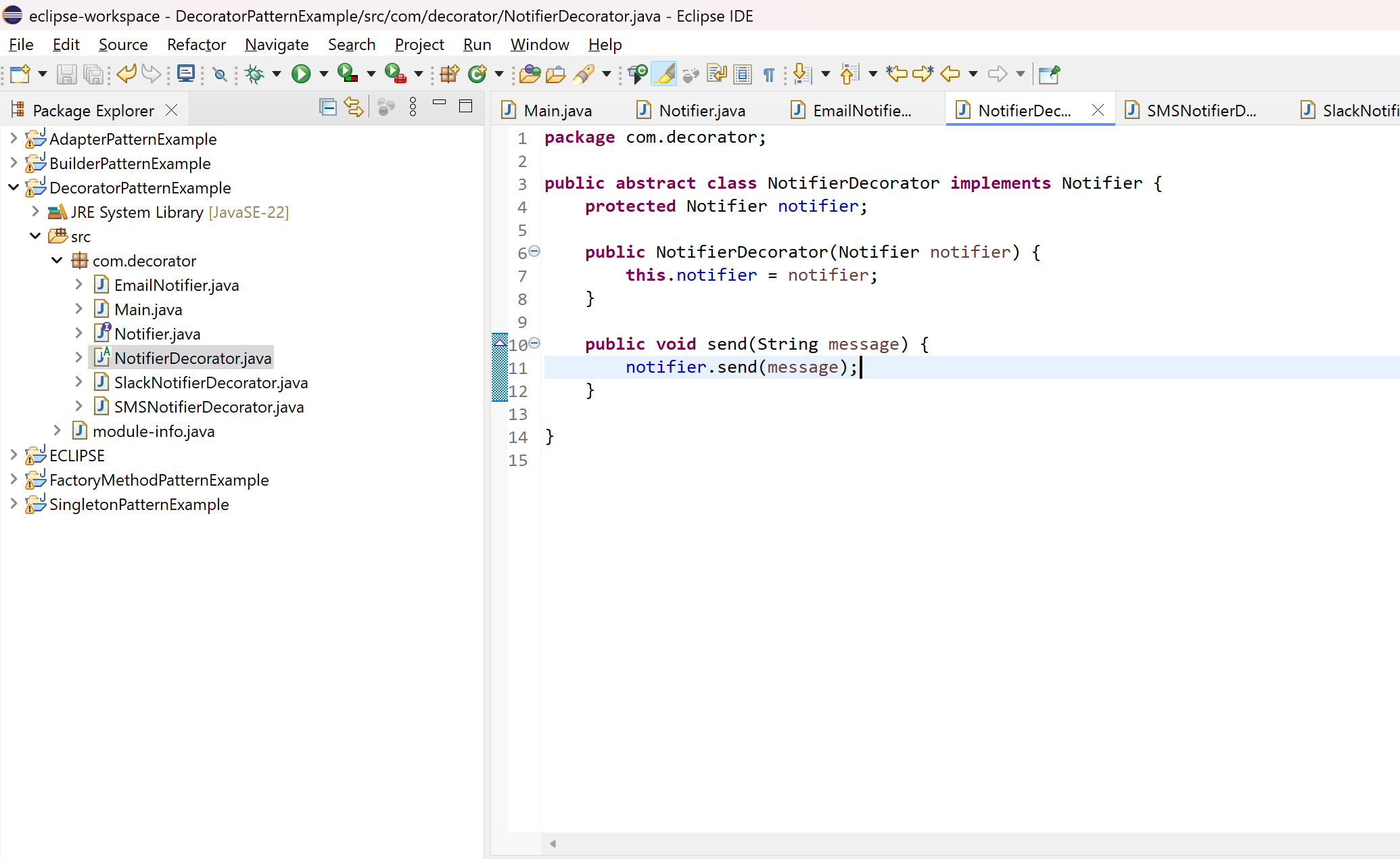
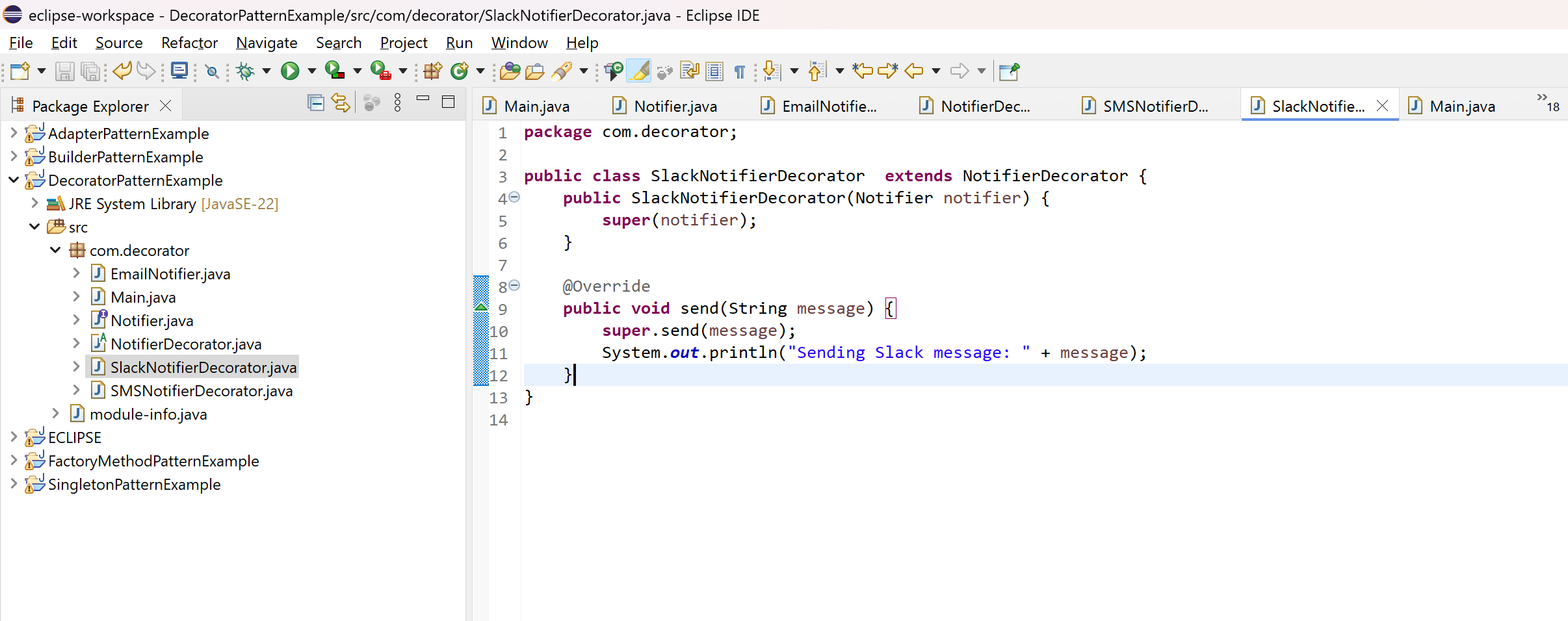
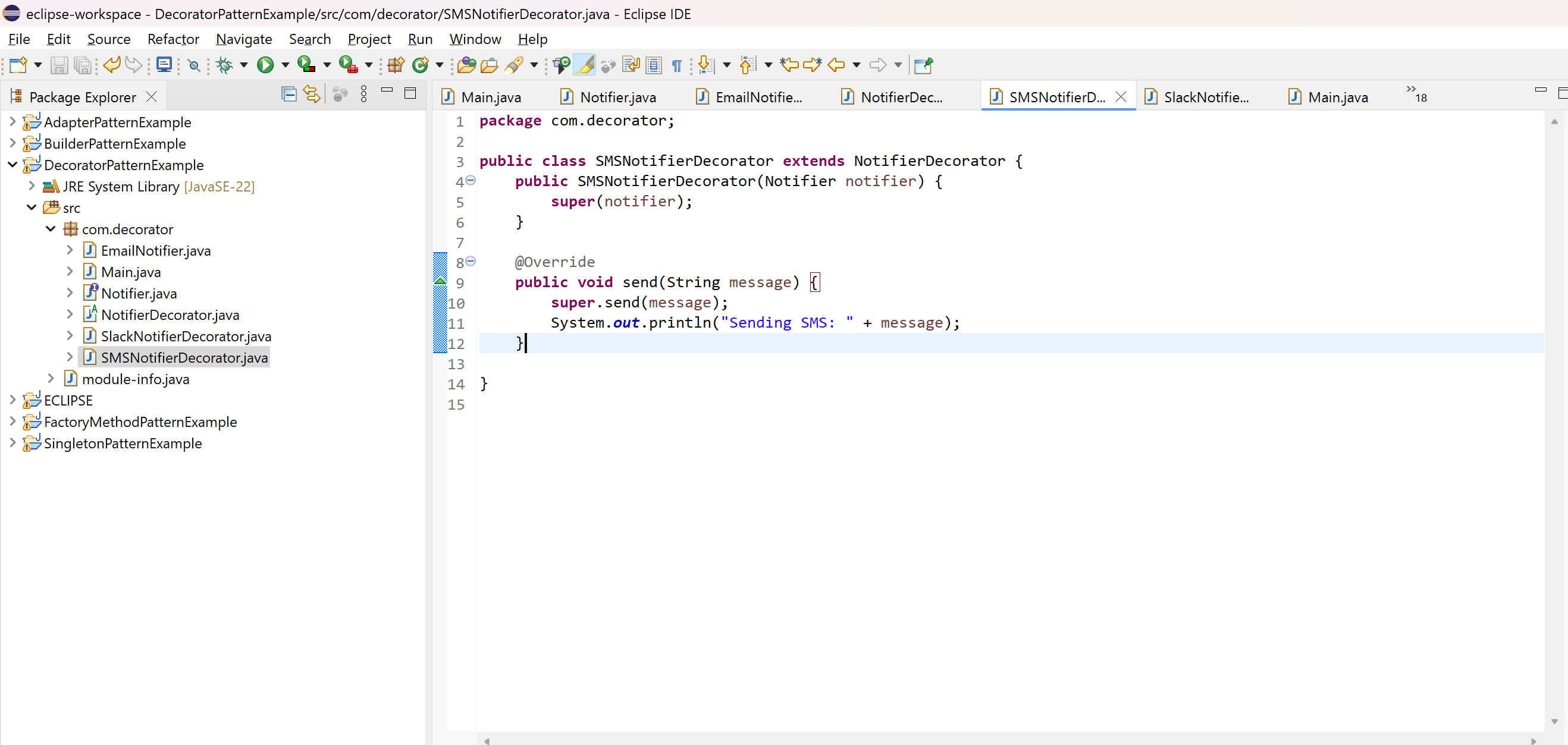
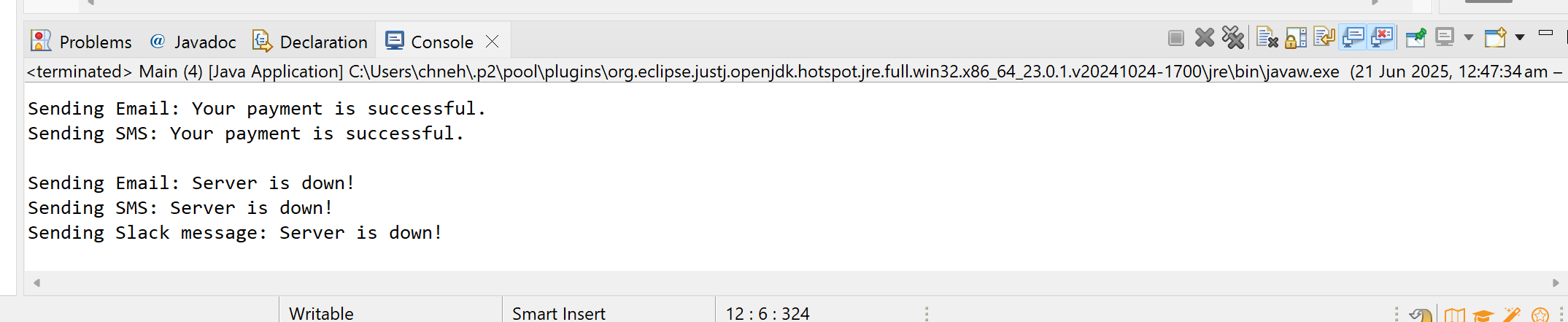
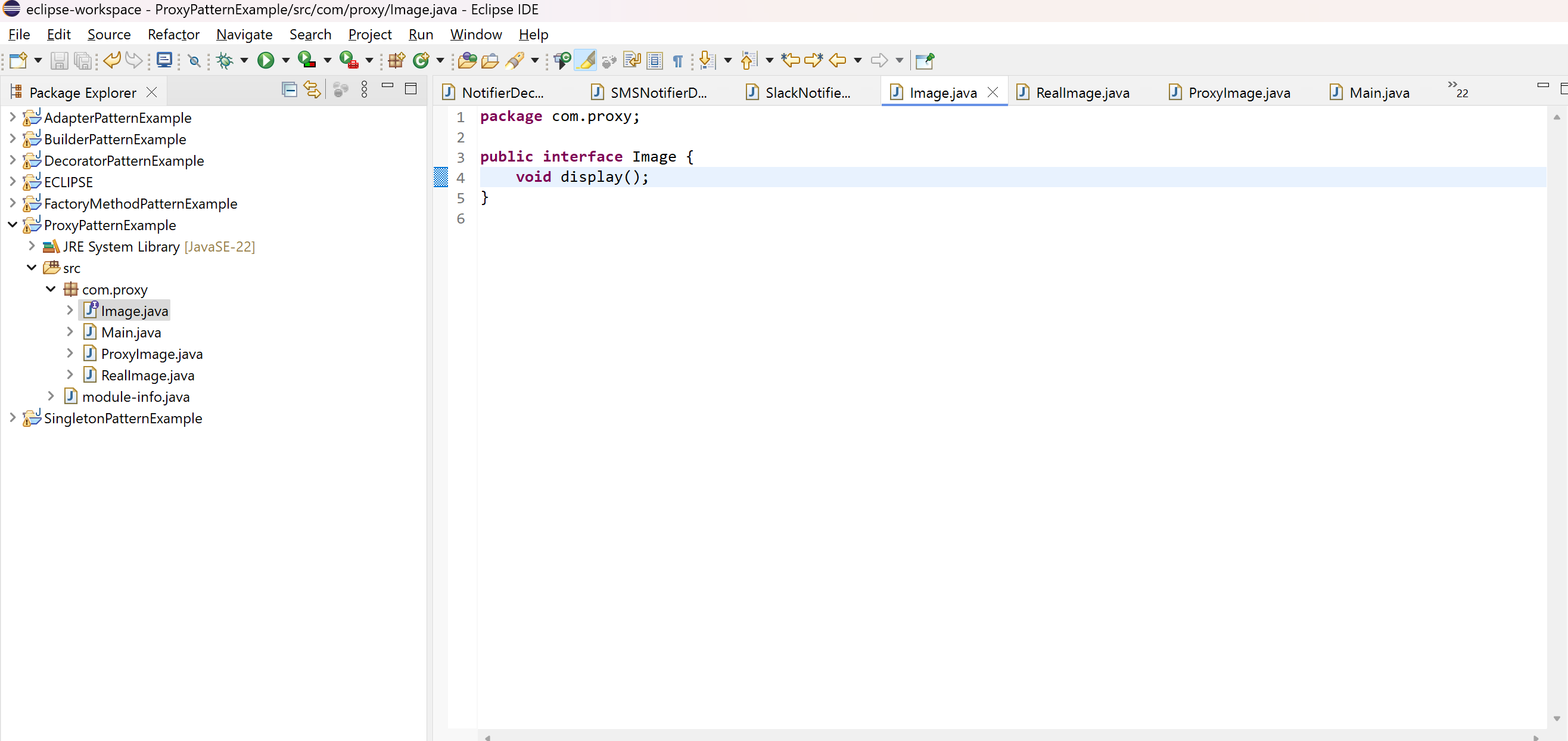
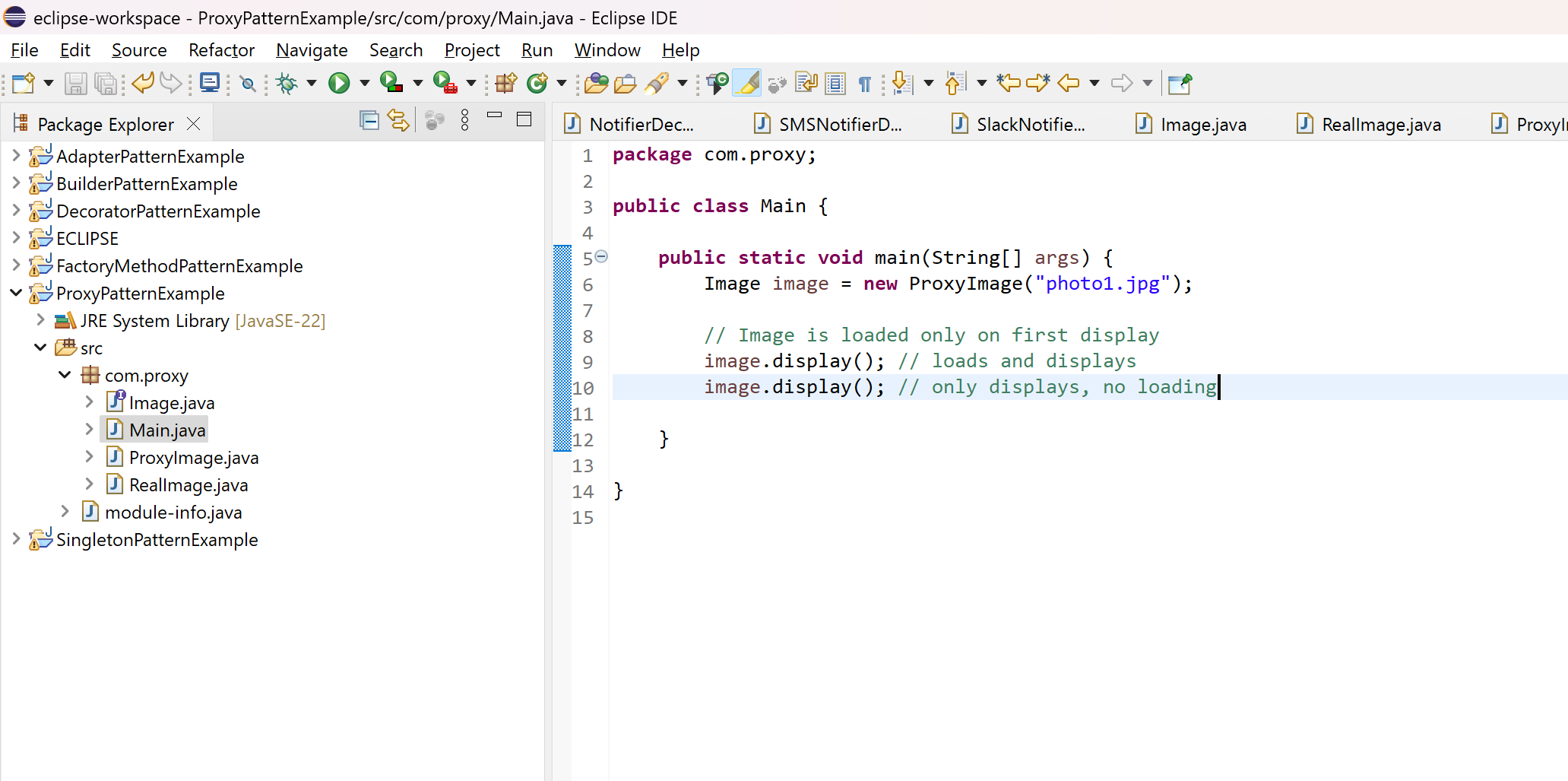
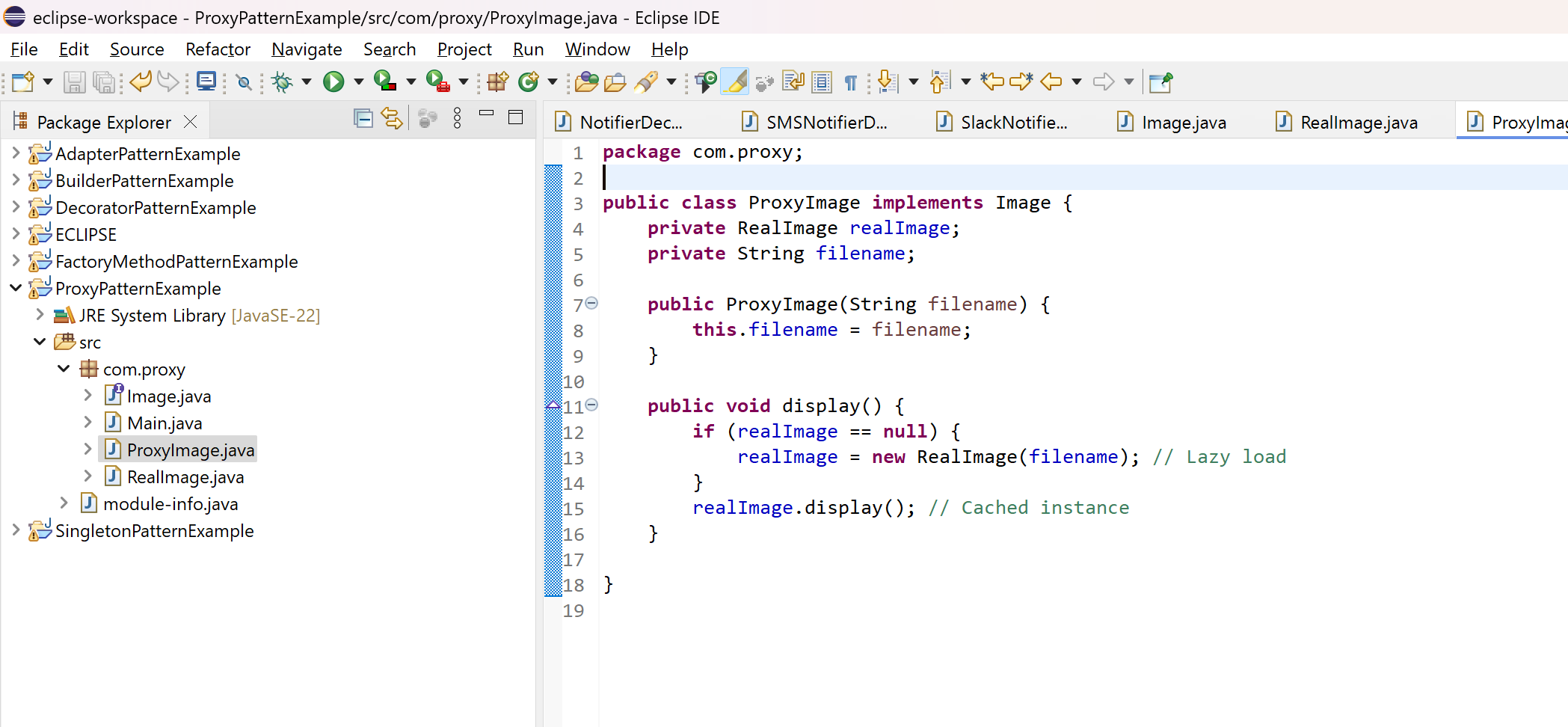
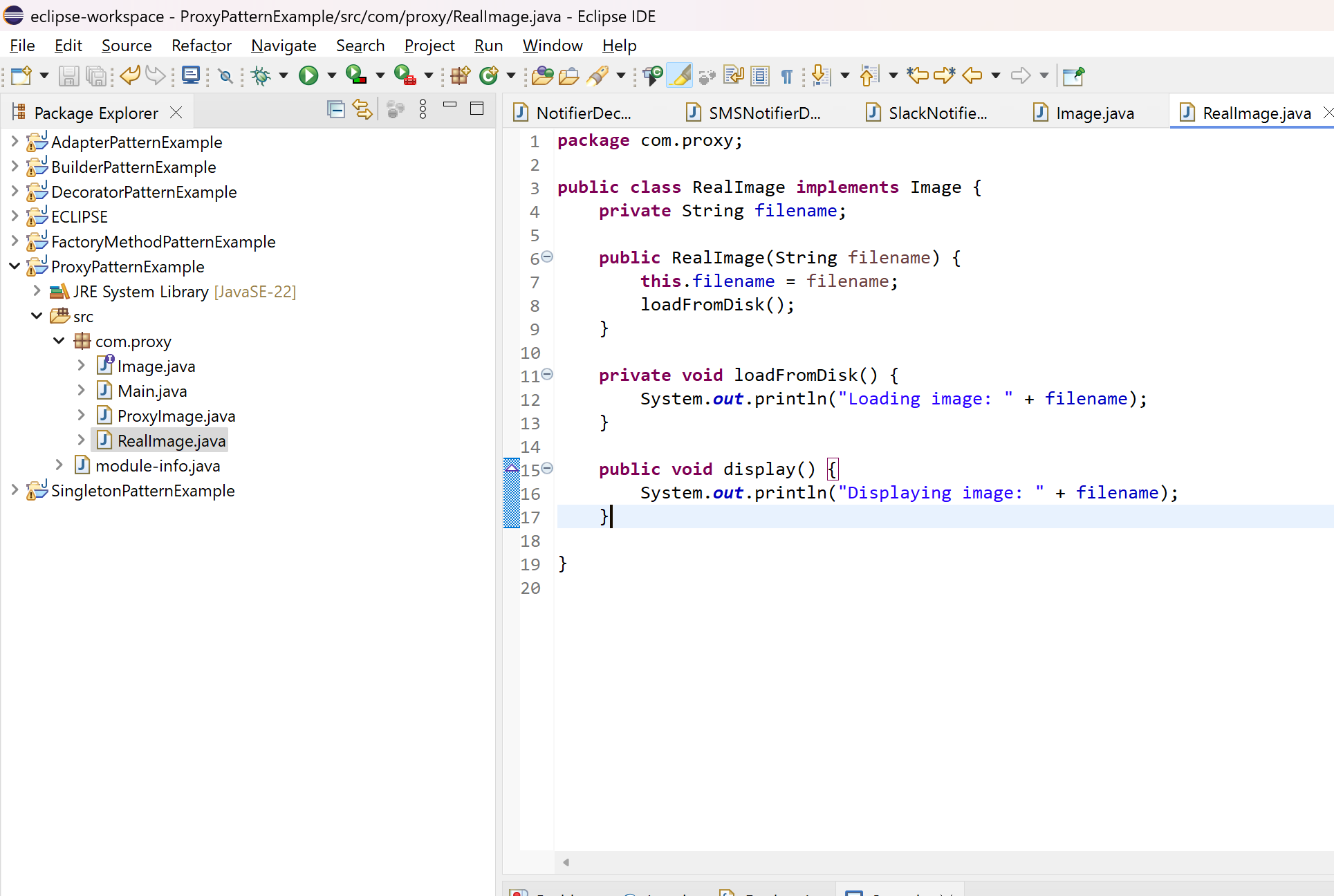
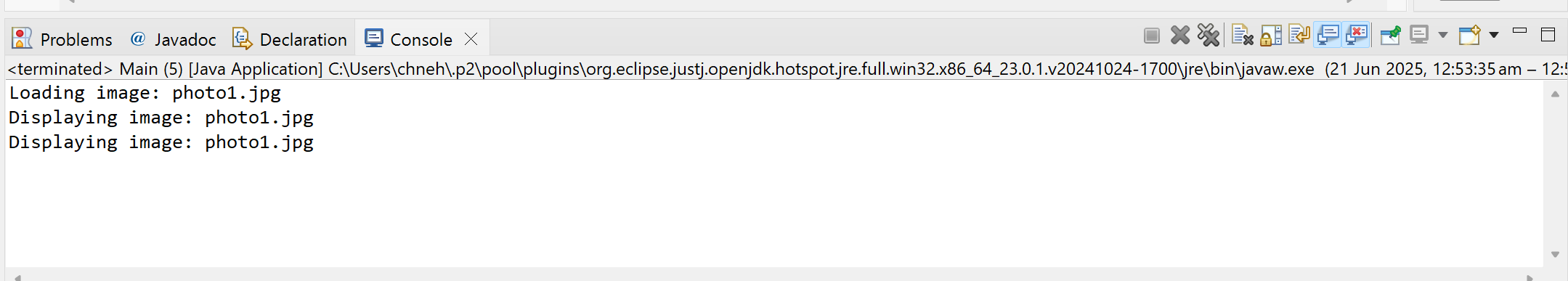
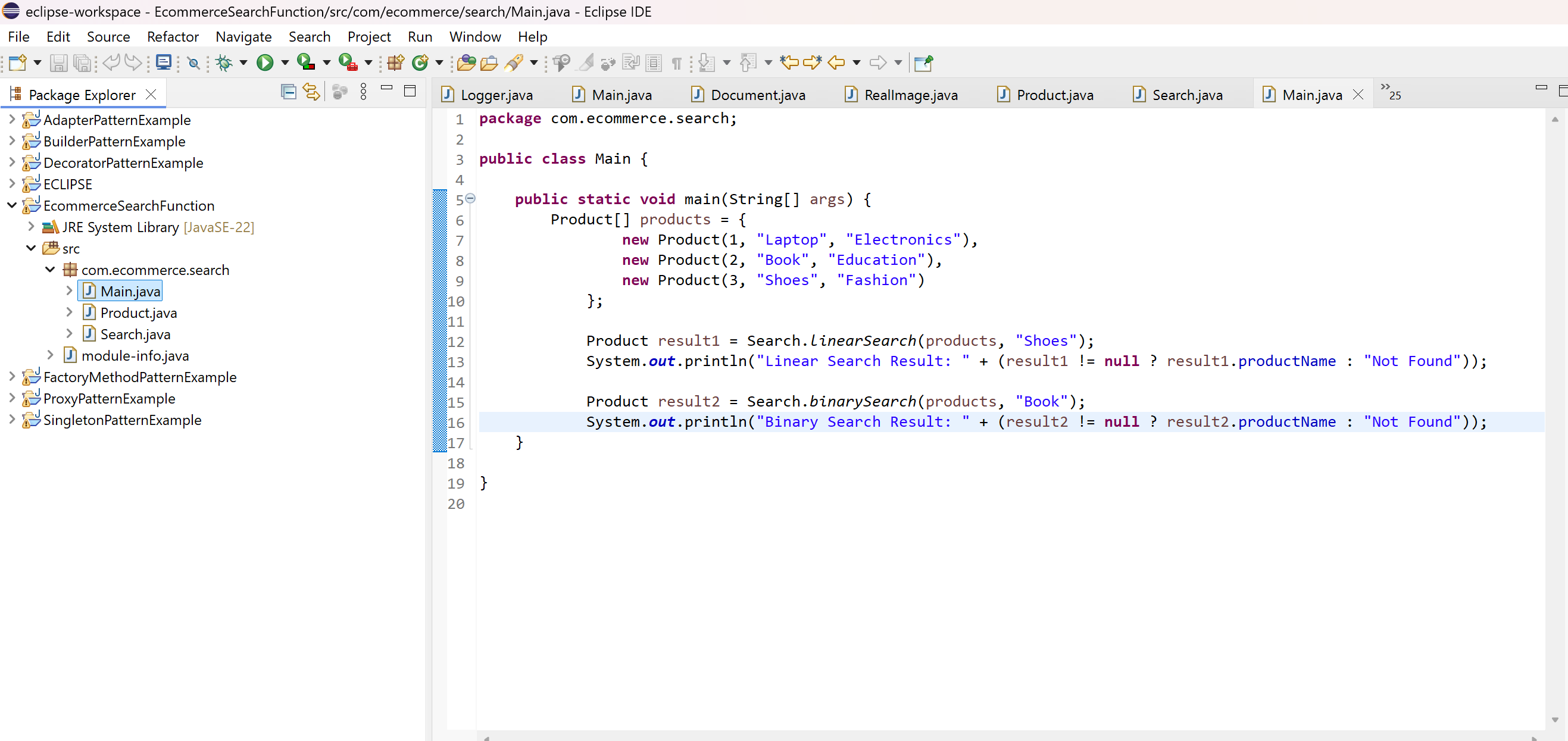
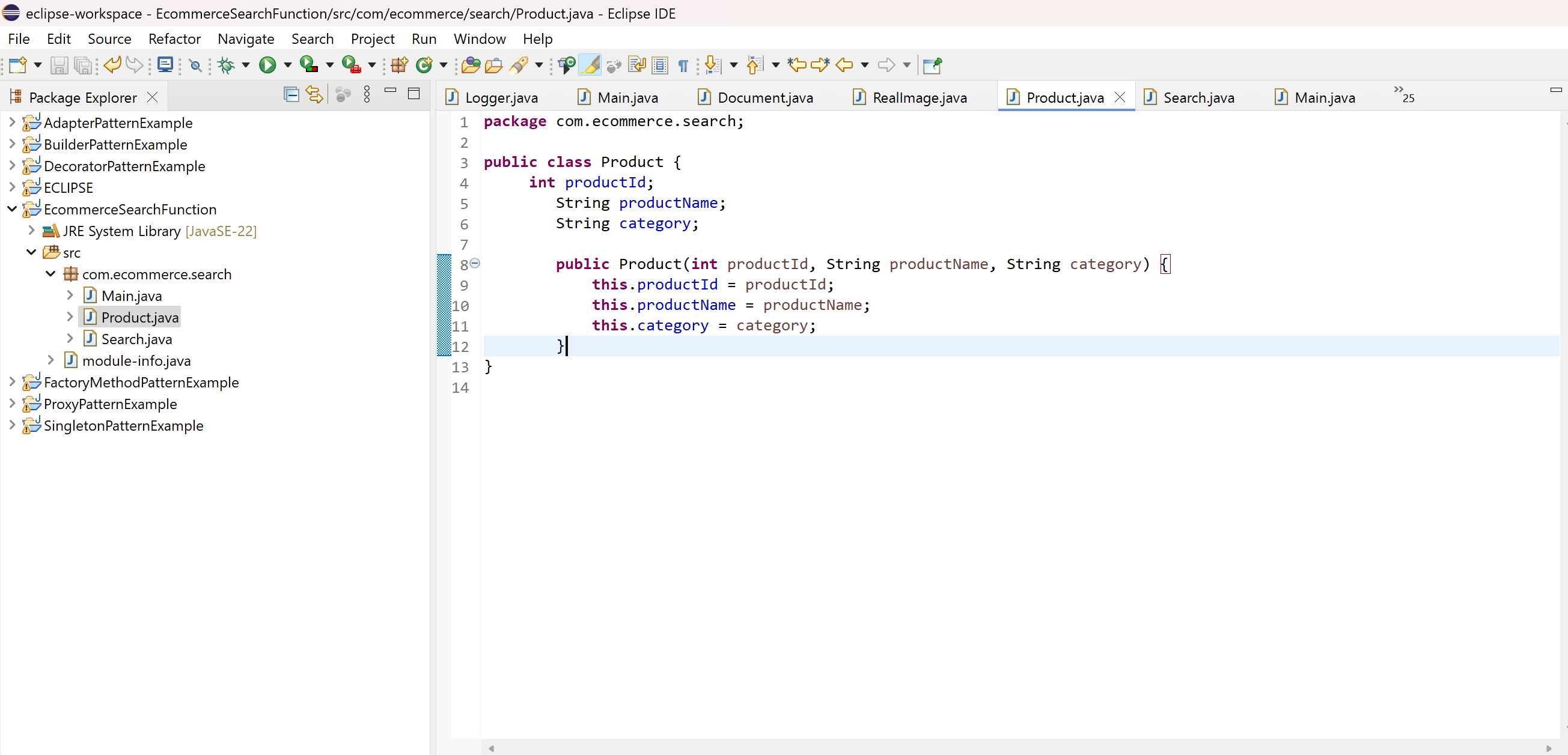
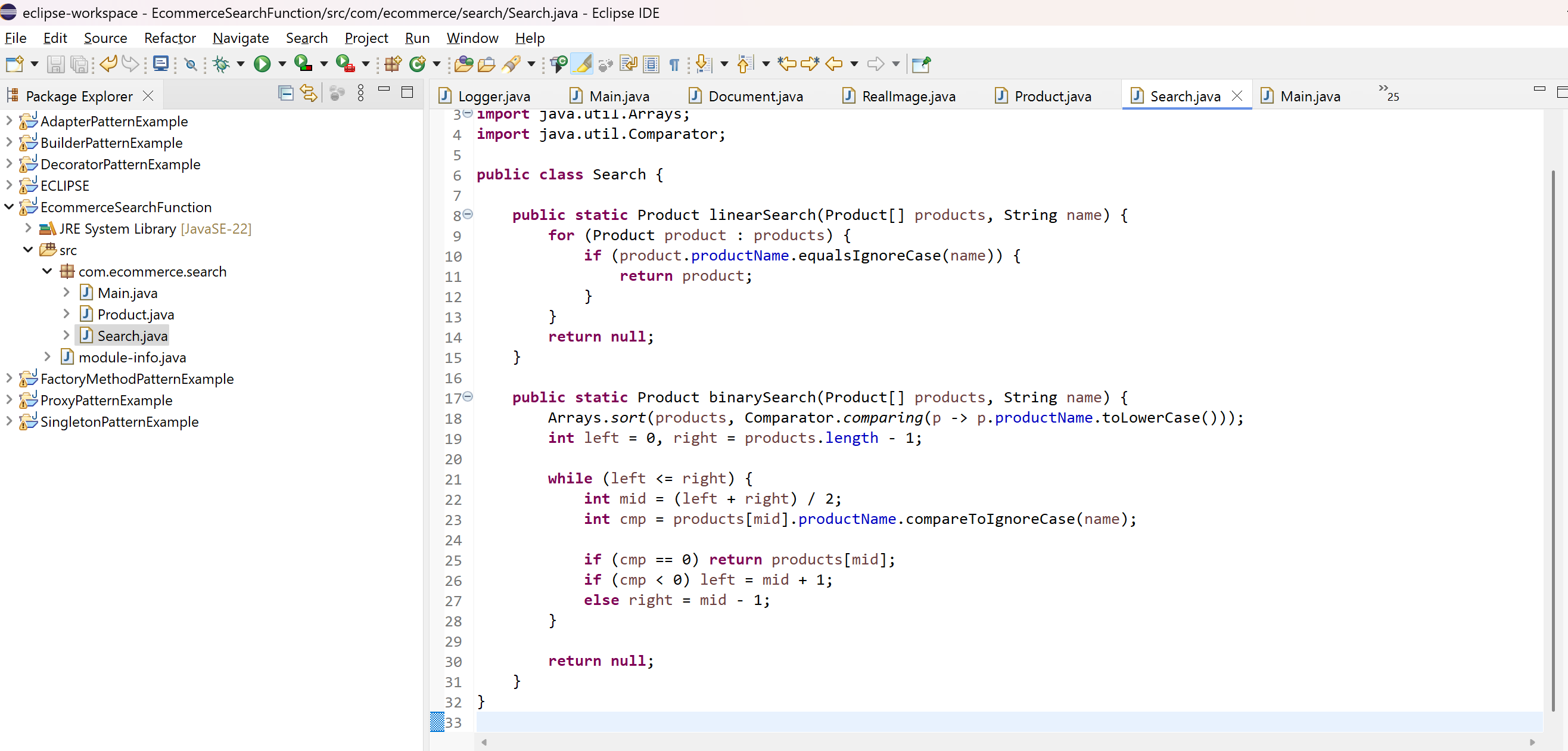
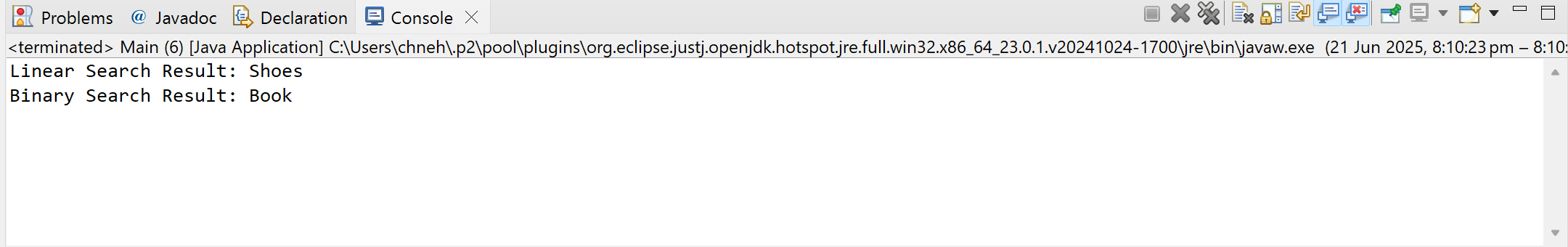
**DESIGN PATTERNS AND PRINCIPLES**  
**1) Implementing the Singleton Pattern**  
  
  
  
  
OUTPUT:  
  
  
  
**2) Implementing the Factory Method Pattern**  
  
  
  
  
  
  
  
  
  
  
  
OUTPUT  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
**3) Implementing the Builder Pattern**  
  
  
  
OUTPUT:  
  
  
**4) Implementing the Adapter Pattern**  
  
  
  
  
  
  
  
OUTPUT:  
  
  
**5) Implementing the Decorator Pattern**  
  
  
  
  
  
  
  
OUTPUT:  
  
  
**6) Implementing the Proxy Pattern**  
  
  
  
  
  
OUTPUT:  


**ALGORITHMNS\_DATA STRUCTURES**  
  
**Exercise 2: E-commerce Platform Search Function**

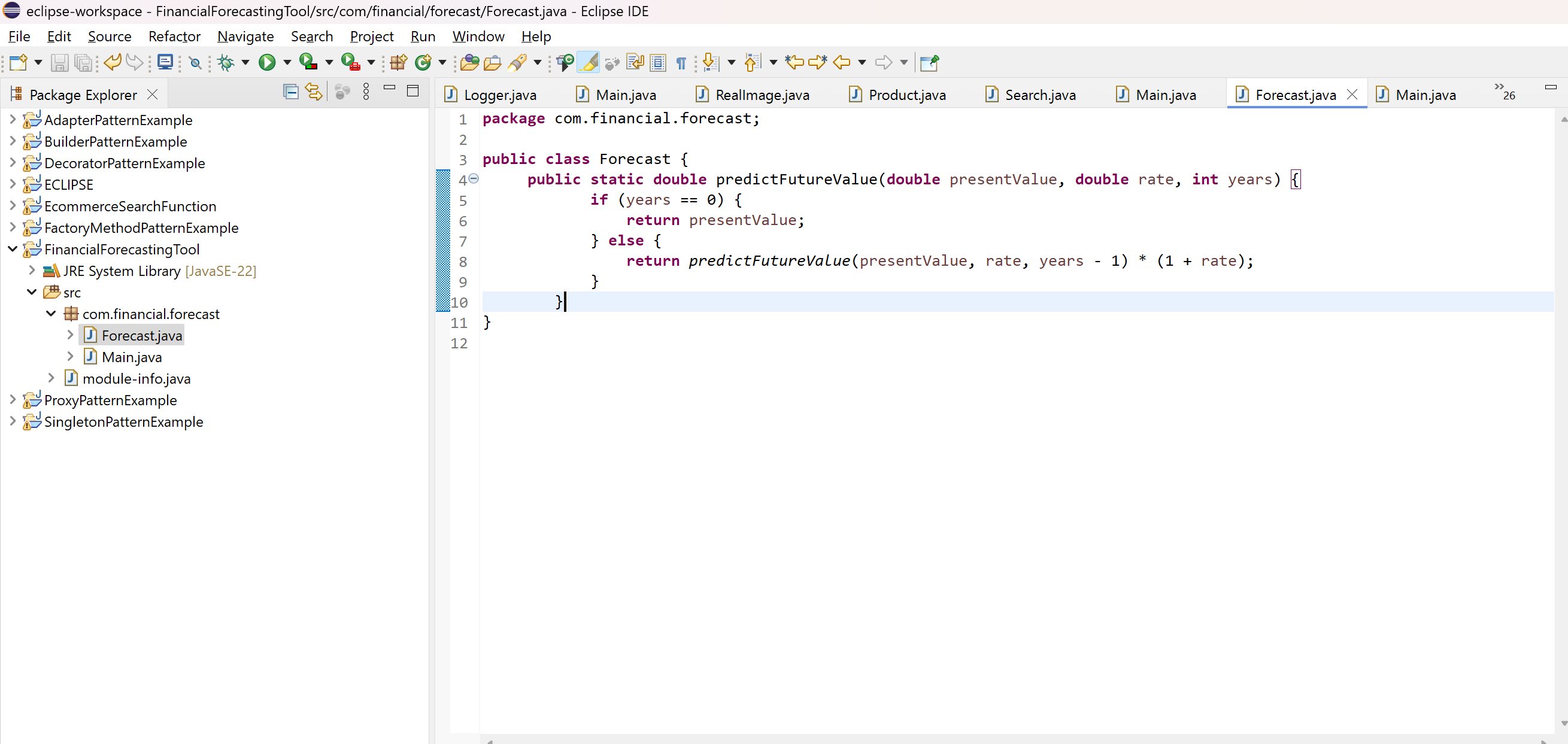
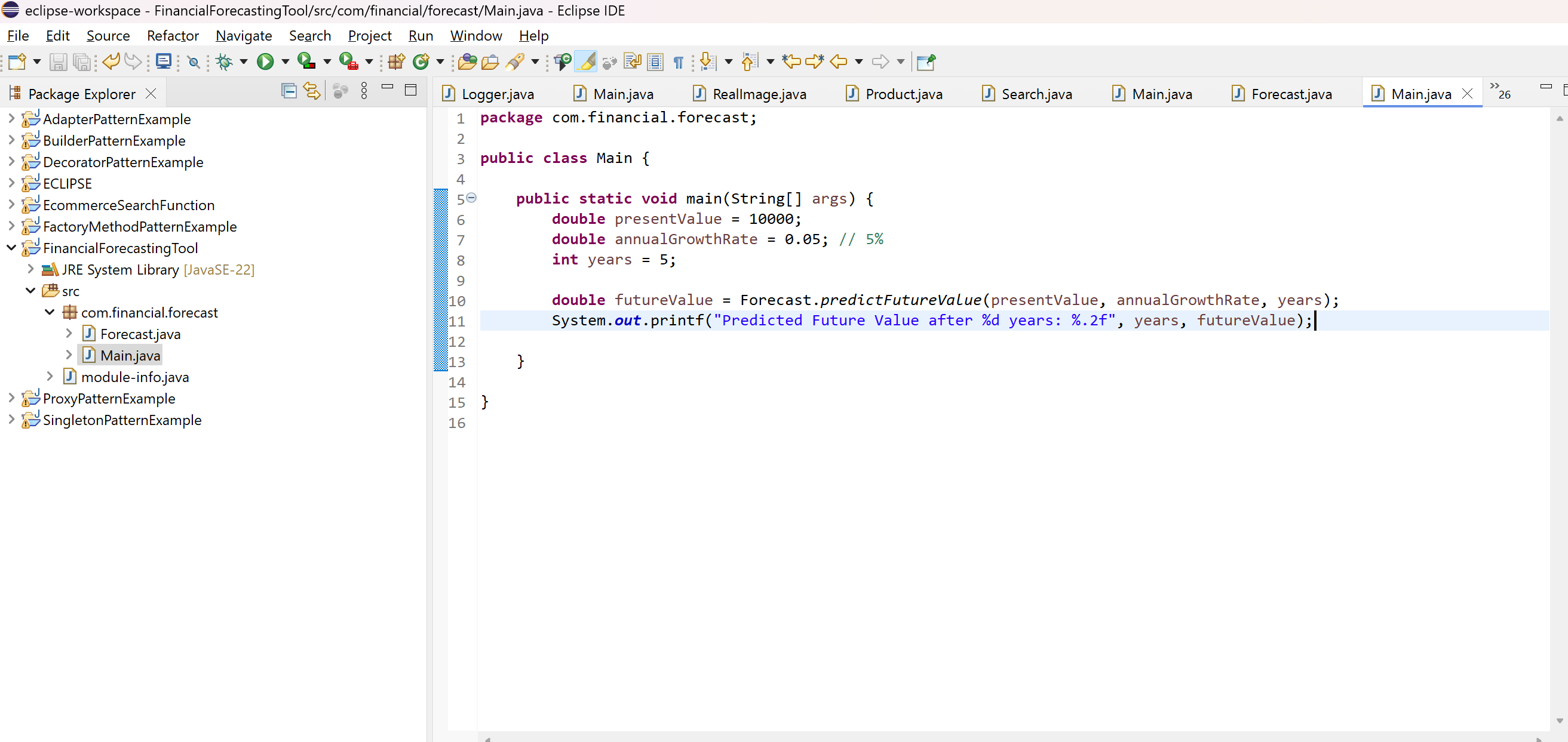
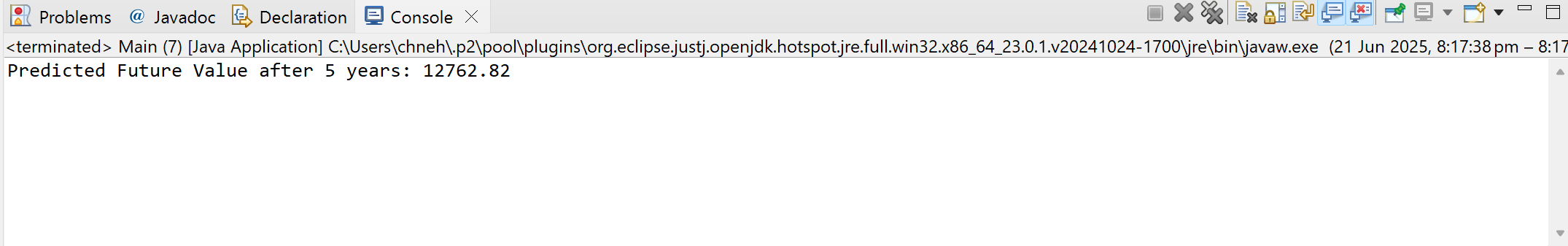
Big O notation is used to describe the performance of an algorithm as the input size increases. It helps in understanding how the algorithm scales.

- Best Case: The minimum time required (e.g., item is at the beginning).

- Average Case: Expected time for typical inputs.

- Worst Case: Maximum time required (e.g., item not found or at the end).  
  
  
  
  
  
  
OUTPUT:  
  
  
  
**Exercise 7: Financial Forecasting**

Recursion is a technique where a function calls itself to solve a problem. It is particularly useful in scenarios where a problem can be divided into subproblems of the same type.

Example: Predicting future value based on compound growth — where each year's value depends on the previous year's.  
  
  
  
  
  
  
OUTPUT:  
  
  
time complexity = **O(n)**