**LAPORAN CODELAB 1 PEMROGRAMAN LANJUT**

**Nama: Naufal Arkaan**

**Nim: 202410370110020**

**Mata Kuliah: Praktikum Pemrograman Lanjut B**

**1. Judul**

Refactoring Program Pengelolaan Data Buku dan Perpustakaan

**2. Tujuan**

Mempraktikkan penerapan prinsip **refactoring** pada pemrograman berorientasi objek menggunakan Java untuk meningkatkan keterbacaan, efisiensi, dan struktur kode.

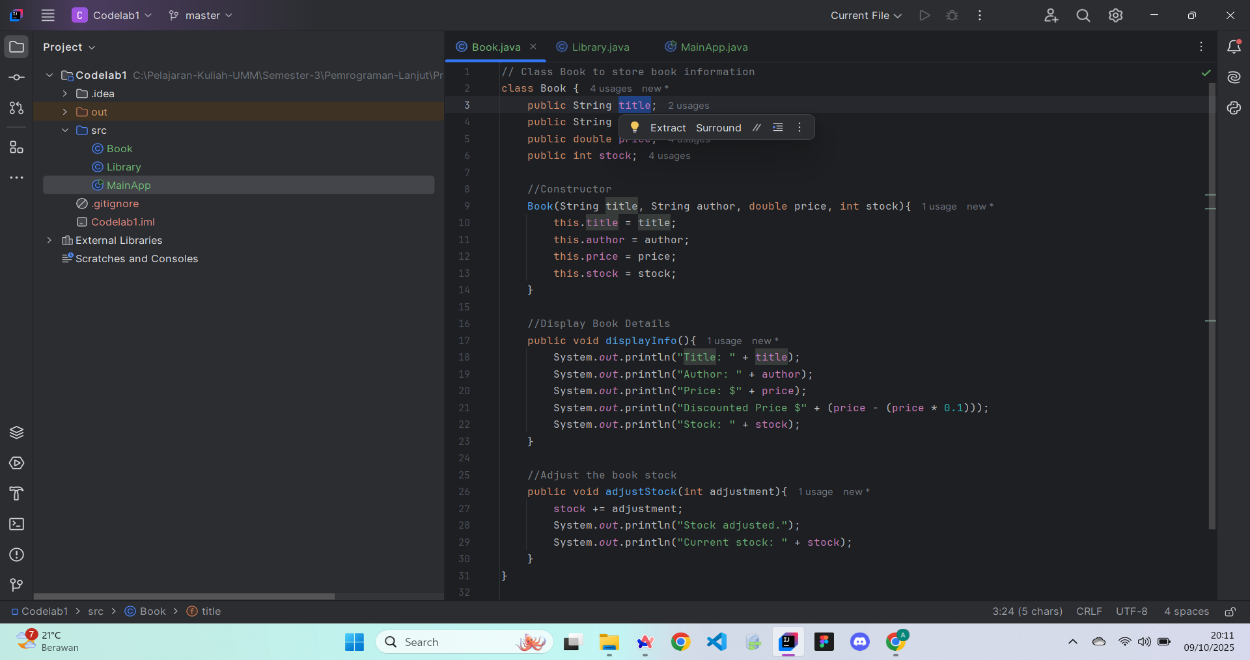
**3. Dasar Teori**

* **Encapsulation**: Melindungi data agar hanya bisa diakses melalui metode khusus (getter dan setter).
* **Constant**: Nilai yang tetap selama program berjalan, dideklarasikan dengan final.
* **Method Extraction**: Pemisahan bagian logika agar lebih modular dan mudah diuji.
* **Refactoring**: Proses memperbaiki struktur internal kode tanpa mengubah perilaku eksternal program.

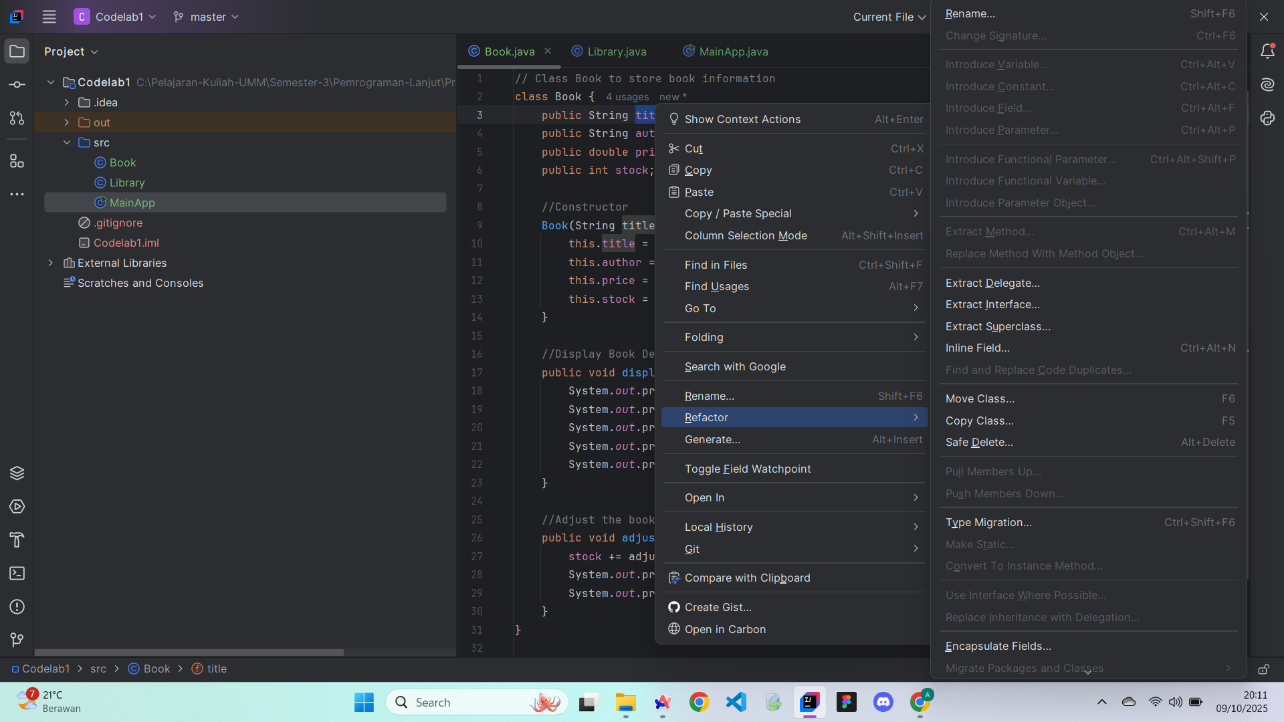
**4. Langkah-Langkah**

1. **Menambahkan getter dan setter untuk setiap atribut pada class Book dan Library.**

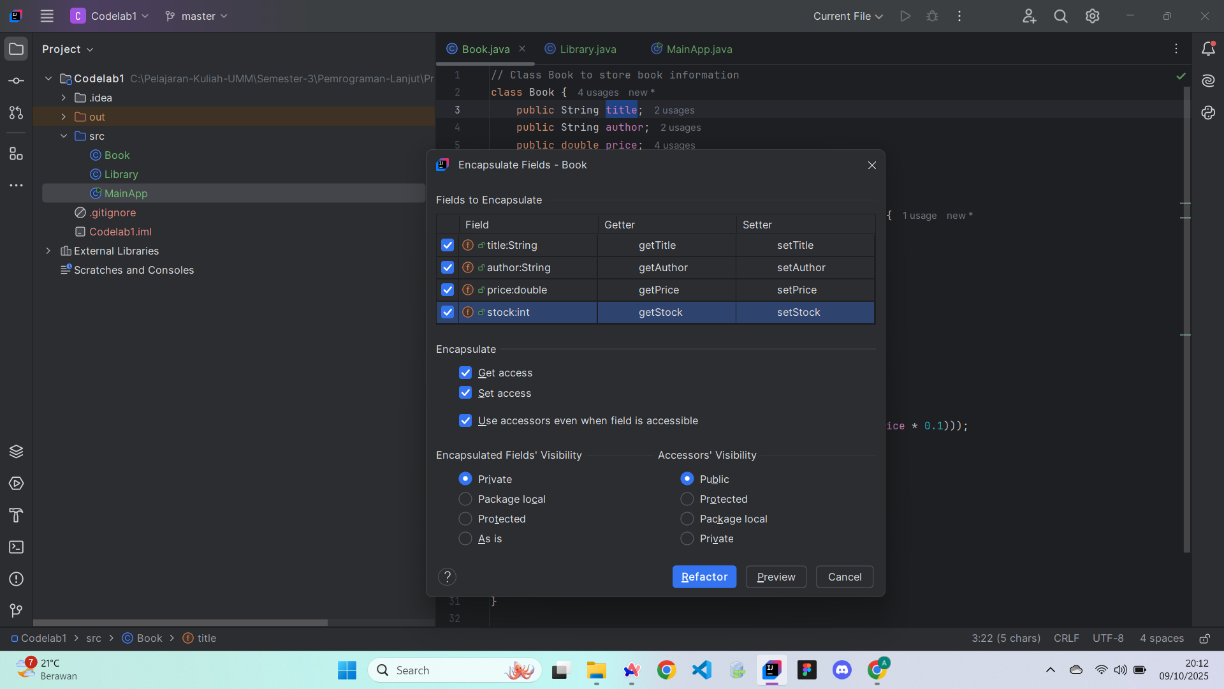
* Langkah awal block code/atribut yang akan dilakukan refactoring pada class **Book**



* Setelah itu klik kanan pada mousepad, pilih refactor dan pilih Encapsulate Field



* Selanjutnya centang atribut yang ingin di Encapsulate Field



* Berikut hasil code setelah di refactoring dan sebelum di refactoring pada class **Book**:

**Sebelum:**

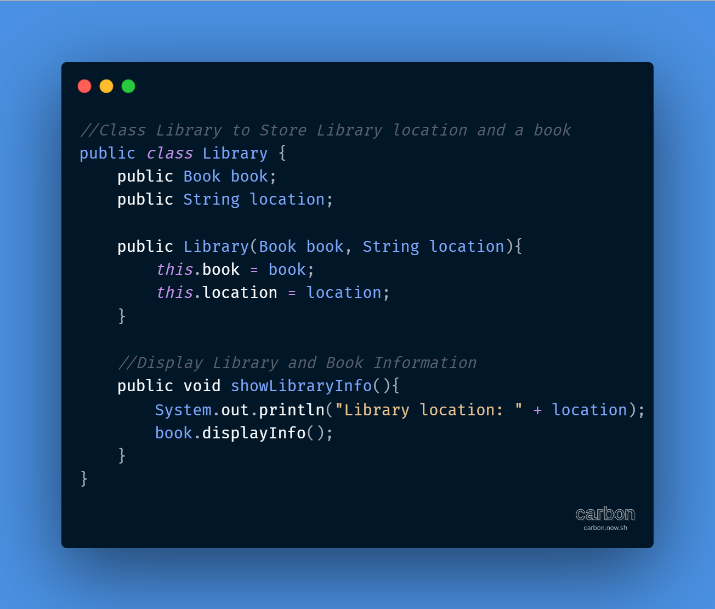


**Sesudah:**

****

* Pada class **Library** kita isa melakukan refactoring Encapsulate Field dengan cara yang sama dengan class **Book**, Saya akan tunjukan code sebelum di refactoring dan sesudah di refactoring:

**Sebelum:**

****

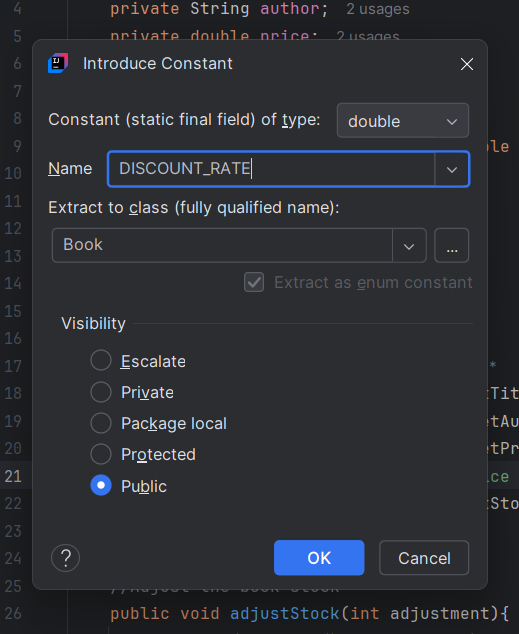
**Sesudah:**

****

setter untuk field book dan location pada Class **Library** otomatis langsung ditambahkan, saya menggunakan exstension tambahan yaitu **carbon** agar code bisa di upload secara jelas dan lengkap.

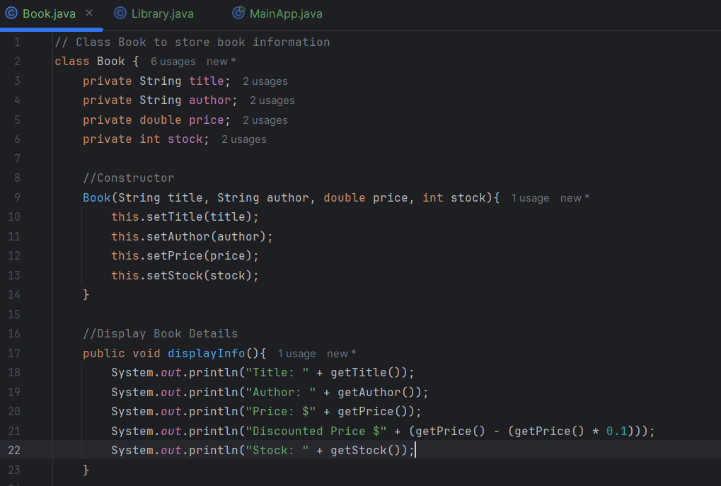
1. **Menambahkan konstanta DISCOUNT\_RATE = 0.1 di class Book.**

* Cara melakukan refactor introduce Constant yaitu Pilih salah satu nilai atau value yang ingin dijadian sebuah konstanta, Klik kanan → pilih Refactor → lalu Introduce Constant, Beri nama konstanta.

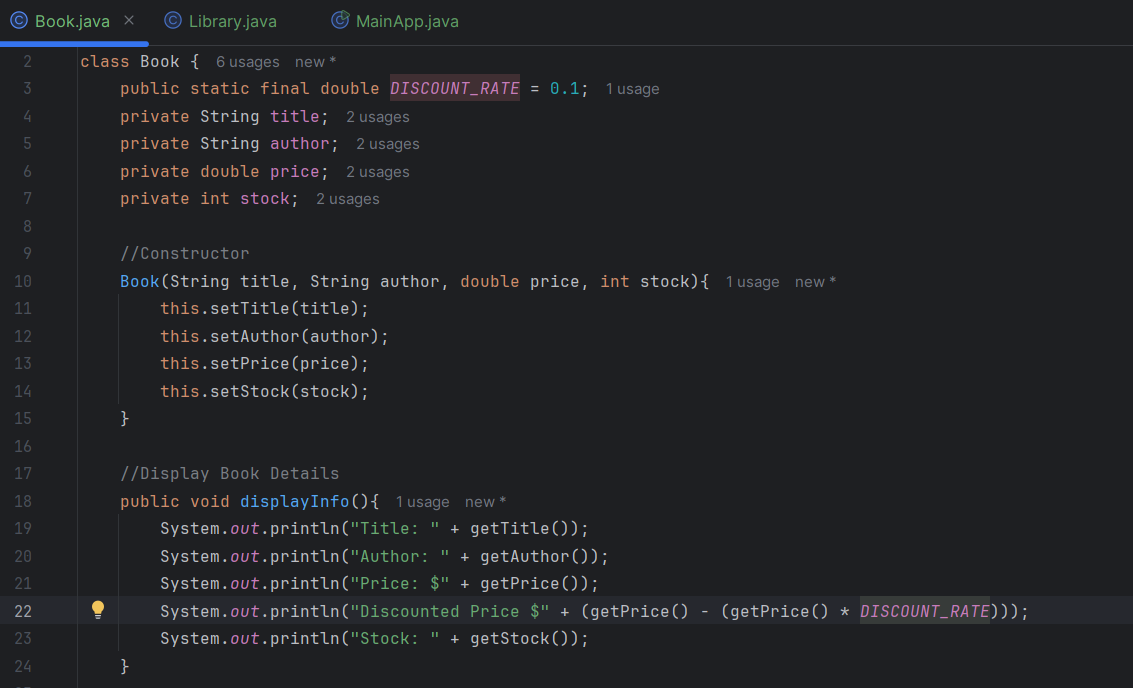
****

* Berikut ada code sebelum dan sesudah di refactoring pada class Book:

**Sebelum:**

****

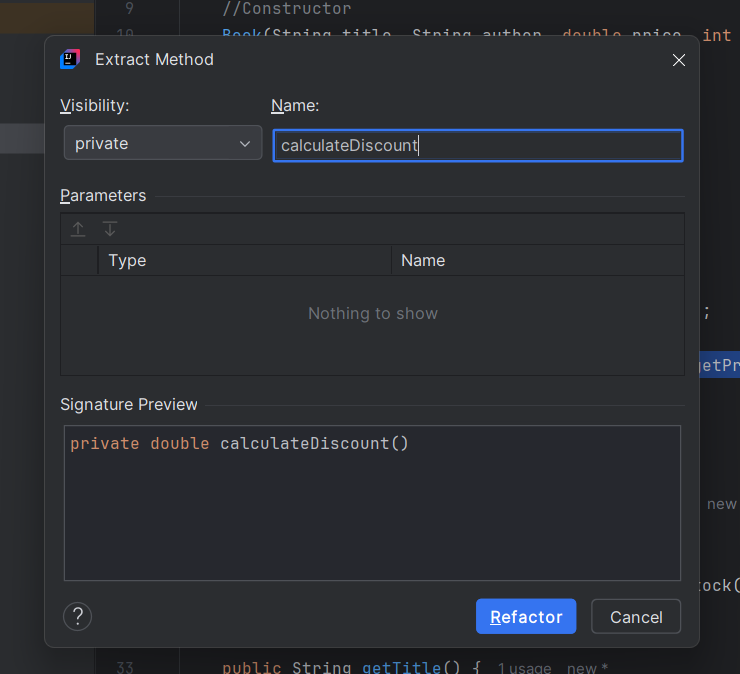
**Sesudah:**



Code sudah otomatis berubah jadi constant/nilai tetap yang ga bisa diubah lagi, sehingga kita lebih mudah menggunakannya berulang kali.

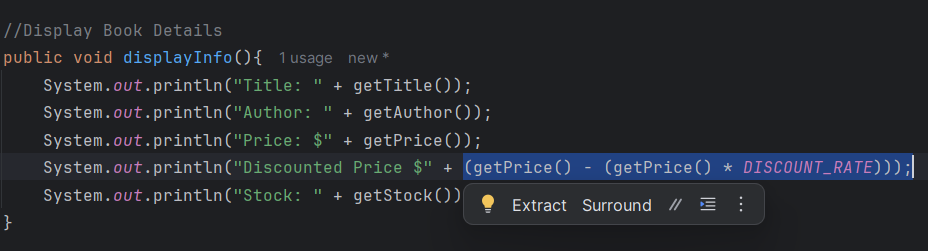
1. **Memisahkan perhitungan diskon dari displayInfo() menjadi metode baru calculateDiscount().**

* Cara melakukan extract method adalah Blok bagian kode yang ingin dipisahkan, Klik kanan pada kode tersebut, pilih Refactor → Extract Method, Berikan nama metode sesuai dengan fungsi atau tujuan dari kode tersebut.

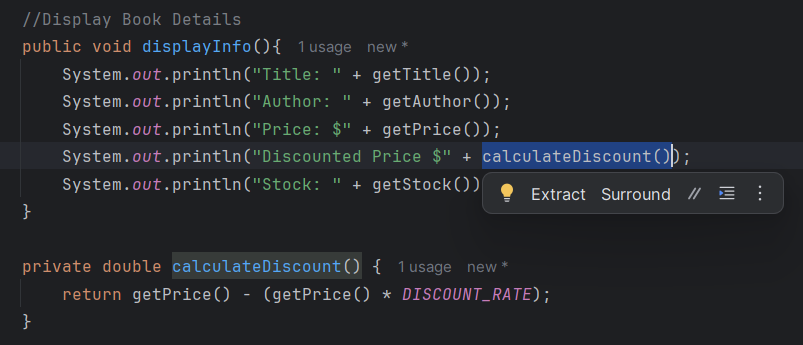


* Berikut saya akan berikan code sebelum dan sesudah di refactor:

**Sebelum:**

****

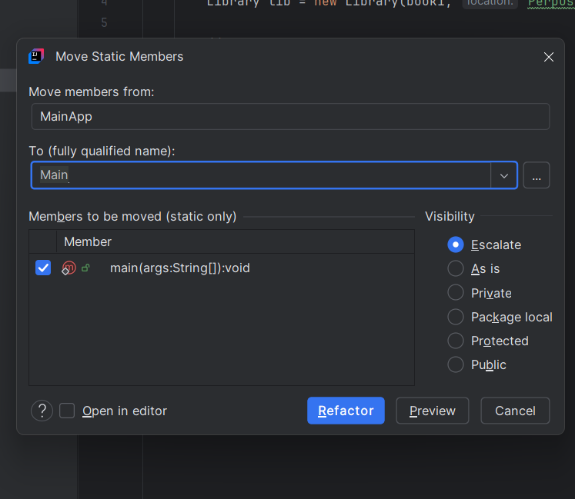
**Sesudah:**



Secara otomatis akan generate sebuah method baru yaitu **calculateDiscount()**.

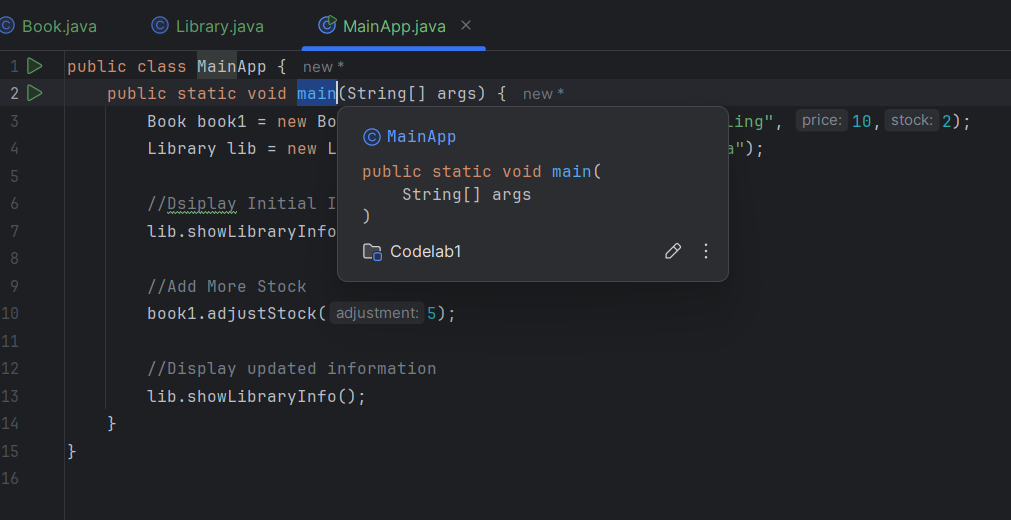
1. **Memindahkan method main() dari class MainApp ke class baru Main.**

* Cara melakukan move method/move members adalah Blok bagian kode/method yang ingin dipindahkan, Klik kanan pada kode tersebut, pilih Refactor → Move Members, Sebelum itu buatkan class Main terlebih dahulu, setelah class main dibuat maka pilihlah class Main sebagai tempat code yang akan dipindahkan.

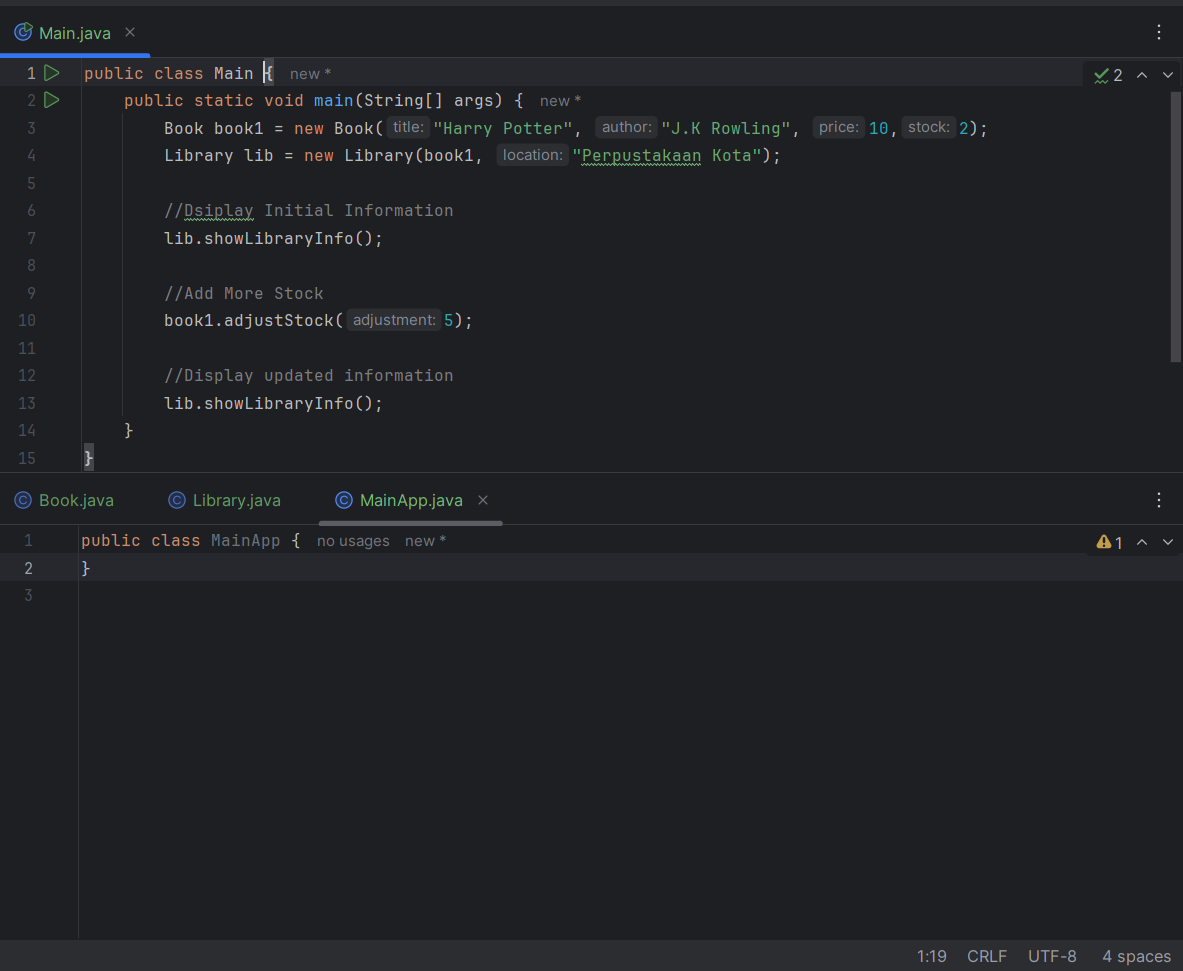


* Berikut saya akan berikan code sebelum dan sesudah di refactor:

**Sebelum:**

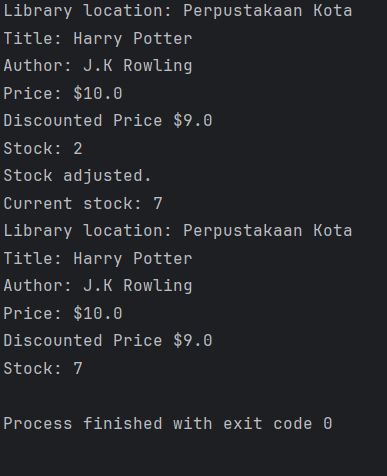


**Sesudah:**



Method **main()** yang semula di class **MainApp** pindah ke class **Main**.

**5. Hasil Program (Output)**

****

**6. Kesimpulan**

Setelah dilakukan refactoring, kode menjadi:

* Lebih mudah dibaca dan dikelola,
* Mengikuti prinsip OOP (Encapsulation, Abstraction),
* Modular dan siap dikembangkan lebih lanjut tanpa mengubah struktur utama.