Laporan Praktikum - CODELAB

Muhammad Radya Iftikhar

202410370110370

Pemrograman Lanjut C

1. Pada class Book, tambahkan setter dan getter untuk field title, author, stock, dan price. Selain itu, buat juga setter untuk field book dan location pada Class Library.

BEFORE

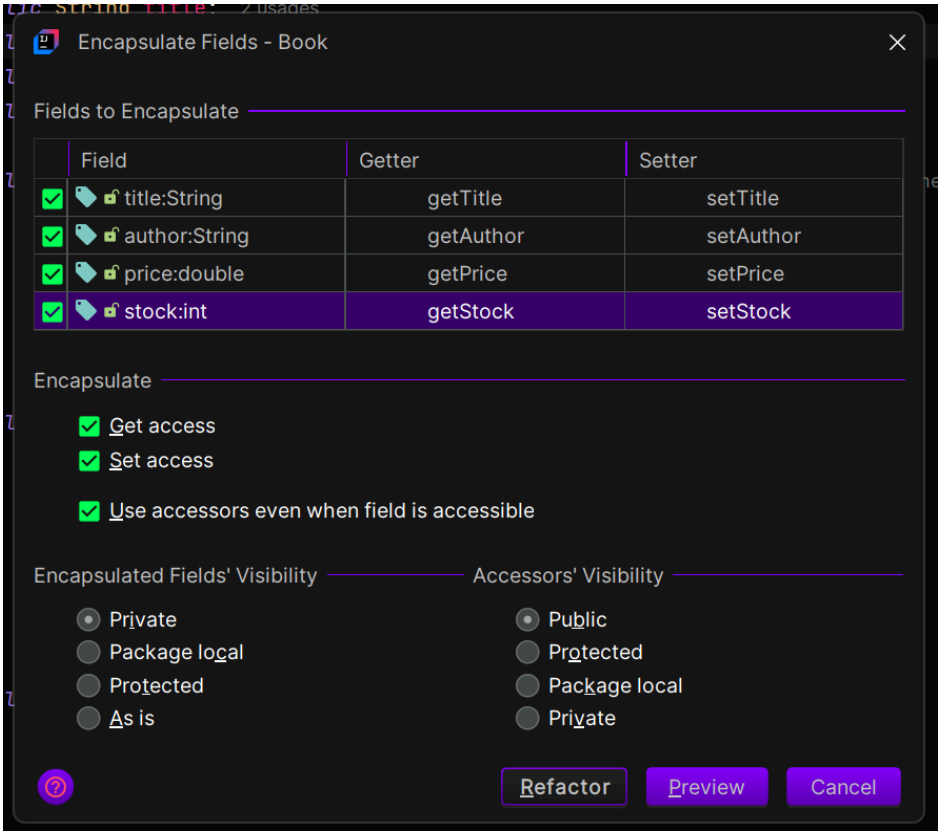
MainApp.java | Book.java | Library.java ×

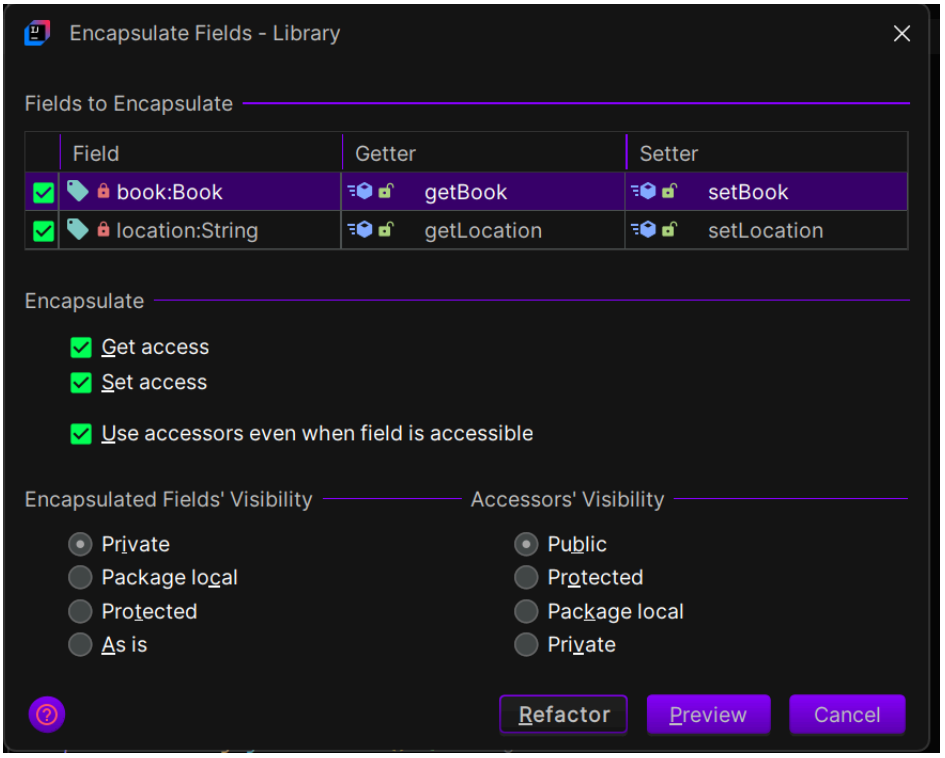
```java
public class Library {  2 usages
    public Book book;  2 usages
    public String location;  2 usages

    public Library(Book book, String location) {  1 usage
        this.book = book;
        this.location = location;
    }

    public void showLibraryInfo() {  2 usages
        System.out.println("Library Location: " + location);
        book.displayInfo();
    }
}
```

MainApp.java | Book.java × | Library.java

```java
public class Book {  4 usages  new *
    public String title;  2 usages
    public String author;  2 usages
    public double price;  4 usages
    public int stock;  4 usages

    public Book(String title, String author, double price, int stock) {  1 usage  new *
        this.title = title;
        this.author = author;
        this.price = price;
        this.stock = stock;
    }

    public void displayInfo() {  1 usage  new *
        System.out.println("Title: " + title);
        System.out.println("Author: " + author);
        System.out.println("Price: $" + price);
        System.out.println("Discounted Price $" + (price - (price * 0.1)));
        System.out.println("Stock: " + stock);
    }

    public void adjustStock(int adjustment) {  1 usage  new *
        stock += adjustment;
        System.out.println("Stock adjusted.");
        System.out.println("Current stock: " + stock);
    }
}
```

Bisa dilihat kode di atas yaitu pada atribut **title**, **author**, **price**, **stock, book, location** belum memiliki **setter** dan **getter**. Untuk menambahkan **setter** dan **getter,** saya menggunakan refactoring **Encapsulate Fields.**





AFTER

Book.java

```java
public class Book {
    private String title;
    private String author;
    private double price;
    private int stock;

    public Book(String title, String author, double price, int stock) {
        this.setTitle(title);
        this.setAuthor(author);
        this.setPrice(price);
        this.setStock(stock);
    }

    public void displayInfo() {
        System.out.println("Title: " + getTitle());
        System.out.println("Author: " + getAuthor());
        System.out.println("Price: $" + getPrice());
        System.out.println("Discounted Price $" + (getPrice() - (getPrice() *
0.1)));
        System.out.println("Stock: " + getStock());
    }

    public void adjustStock(int adjustment) {
        setStock(getStock() + adjustment);
        System.out.println("Stock adjusted.");
        System.out.println("Current stock: " + getStock());
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getAuthor() {
        return author;
    }

    public void setAuthor(String author) {
        this.author = author;
    }

    public double getPrice() {
        return price;
    }

    public void setPrice(double price) {
        this.price = price;
    }

    public int getStock() {
        return stock;
    }

    public void setStock(int stock) {
        this.stock = stock;
    }
}
```

MainApp.java × Book.java Library.java ×

```java
public class Library {  2 usages
    private Book book;  2 usages
    private String location;  2 usages

    public Library(Book book, String location) {  1 usage
        this.setBook(book);
        this.setLocation(location);
    }

    public void showLibraryInfo() {  2 usages
        System.out.println("Library Location: " + getLocation());
        getBook().displayInfo();
    }

    public Book getBook() {  1 usage
        return book;
    }

    public void setBook(Book book) {  1 usage
        this.book = book;
    }

    public String getLocation() {  1 usage
        return location;
    }

    public void setLocation(String location) {  1 usage
        this.location = location;
```

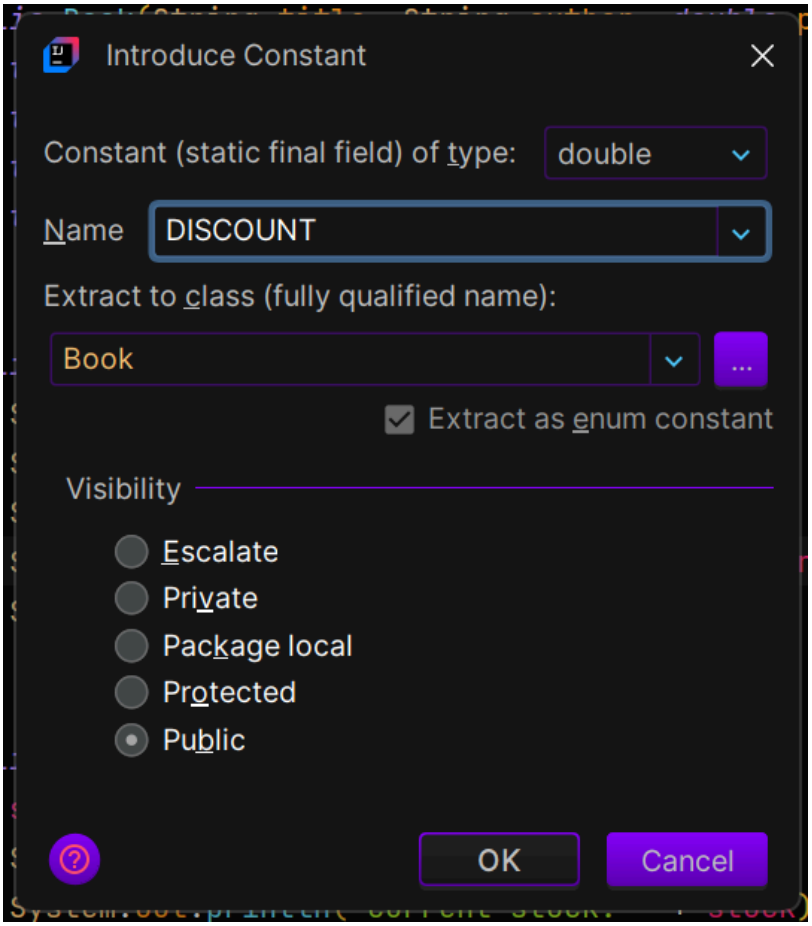
Setter dan **getter** secara otomatis ditambahkan. (**Book.java** karena kode terlalu panjang saya menggunakan bantuan tools untuk screenshot kode: Carbon)

2. Perkenalkan sebuah konstanta baru di Class Book untuk menyimpan nilai diskon (misalnya DISCOUNT_RATE = 0.1).

BEFORE

```java
public void displayInfo() {  1 usage  new *
    System.out.println("Title: " + title);
    System.out.println("Author: " + author);
    System.out.println("Price: $" + price);
    System.out.println("Discounted Price $" + (price - (price * 0.1)));
    System.out.println("Stock: " + stock);
}
```

Pada kode ini, terdapat nilai diskon **0.1** secara **hardcoded (ditulis langsung pada source code)** sehingga tidak efisien ketika ingin menggunakannya berulang kali. Untuk mengatasi ini saya menggunakan refactoring **Introduce Constant** yaitu membuat sebuah variabel constant untuk menampung nilai tersebut.



Saya beri nama **DISCOUNT**

AFTER

```java
MainApp.java   Book.java ×   Library.java

public class Book {  4 usages  new *
    public static final double DISCOUNT = 0.1;  1 usage
    public String title;  2 usages
    public String author;  2 usages
    public double price;  4 usages
    public int stock;  4 usages

    public Book(String title, String author, double price, int stock) {  1 usage  new *
        this.title = title;
        this.author = author;
        this.price = price;
        this.stock = stock;
    }

    public void displayInfo() {  1 usage  new *
        System.out.println("Title: " + title);
        System.out.println("Author: " + author);
        System.out.println("Price: $" + price);
        System.out.println("Discounted Price $" + (price - (price * DISCOUNT)));
        System.out.println("Stock: " + stock);
    }

    public void adjustStock(int adjustment) {  1 usage  new *
```

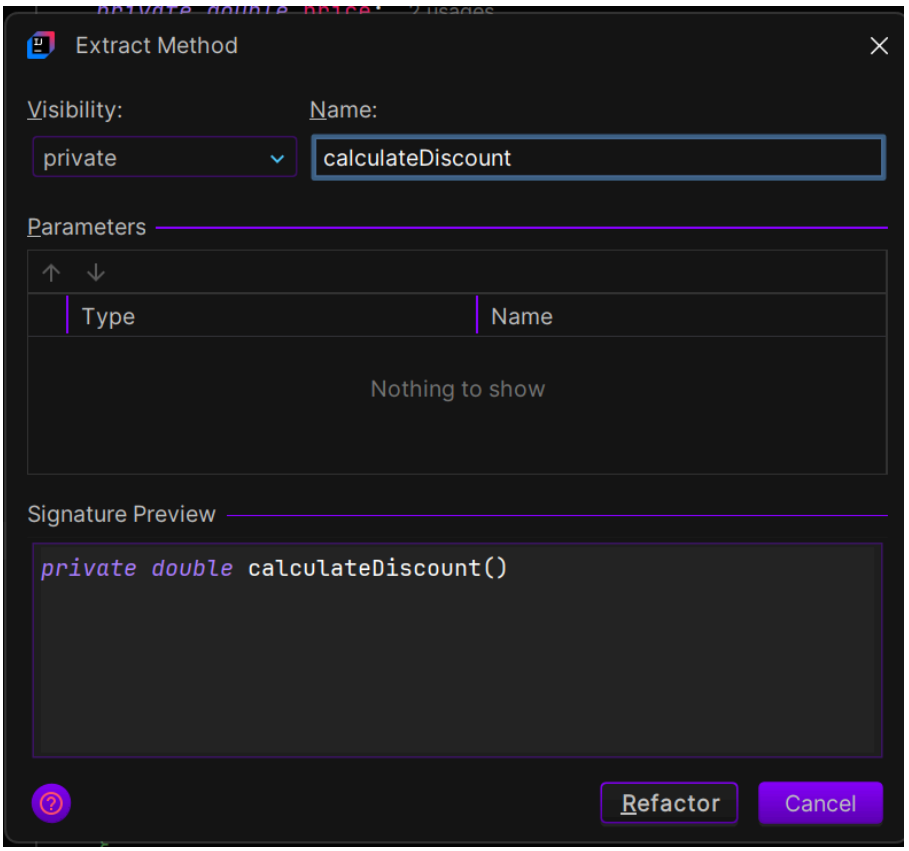
Sudah terdapat sebuah variabel **constant** (nilai yang tidak bisa diubah), sehingga menjadi lebih mudah ketika ingin menggunakannya berulang kali.

3. Pisahkan perhitungan harga diskon dari displayInfo() menjadi sebuah metode baru di kelas Book dengan nama calculateDiscount().

BEFORE

```java
public void displayInfo() {  1 usage  new *
    System.out.println("Title: " + title);
    System.out.println("Author: " + author);
    System.out.println("Price: $" + price);
    System.out.println("Discounted Price $" + (price - (price * DISCOUNT)));
    System.out.println("Stock: " + stock);
}
```

Pada kode di atas, untuk menghitung diskon masih langsung di method **displayInfo(),** untuk membuat method tersendiri khusus menghitung diskon saya menggunakan refactoring **Extract Method.**



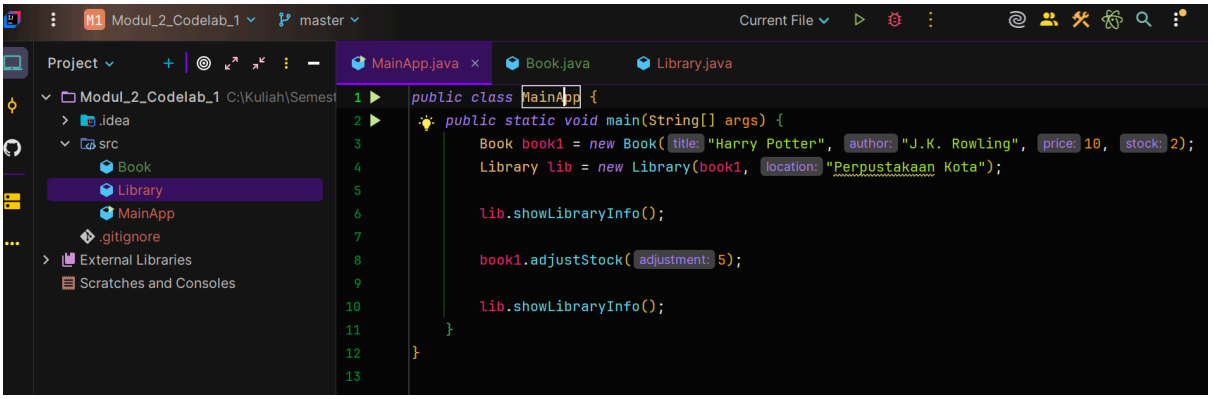
AFTER

```
public void displayInfo() {  1 usage  new *
    System.out.println("Title: " + getTitle());
    System.out.println("Author: " + getAuthor());
    System.out.println("Price: $" + getPrice());
    System.out.println("Discounted Price $" + calculateDiscount());
    System.out.println("Stock: " + getStock());
}

private double calculateDiscount() {  1 usage  new *
    return getPrice() - (getPrice() * DISCOUNT);
}
```

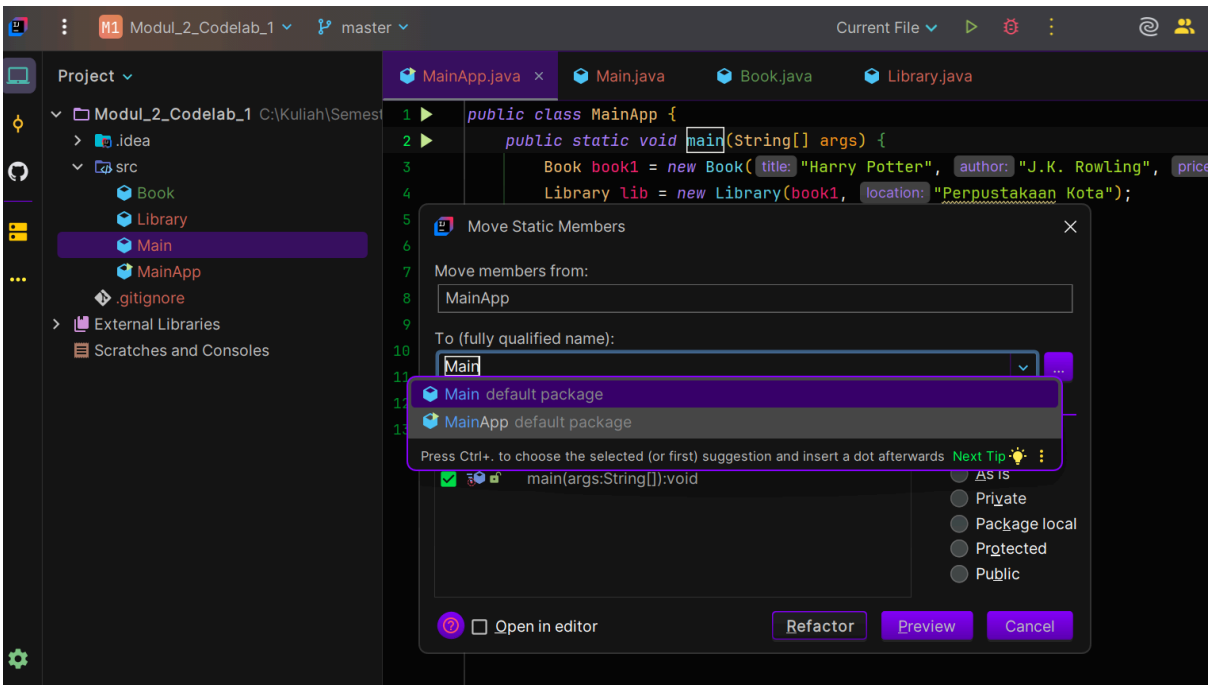
Secara otomatis akan membuat sebuah method baru bernama **calculateDiscount().**

4. Pindahkan method main() dari class MainApp ke dalam kelas baru bernama Main (buat baru) dan pastikan bahwa kelas MainApp dihapus setelahnya.

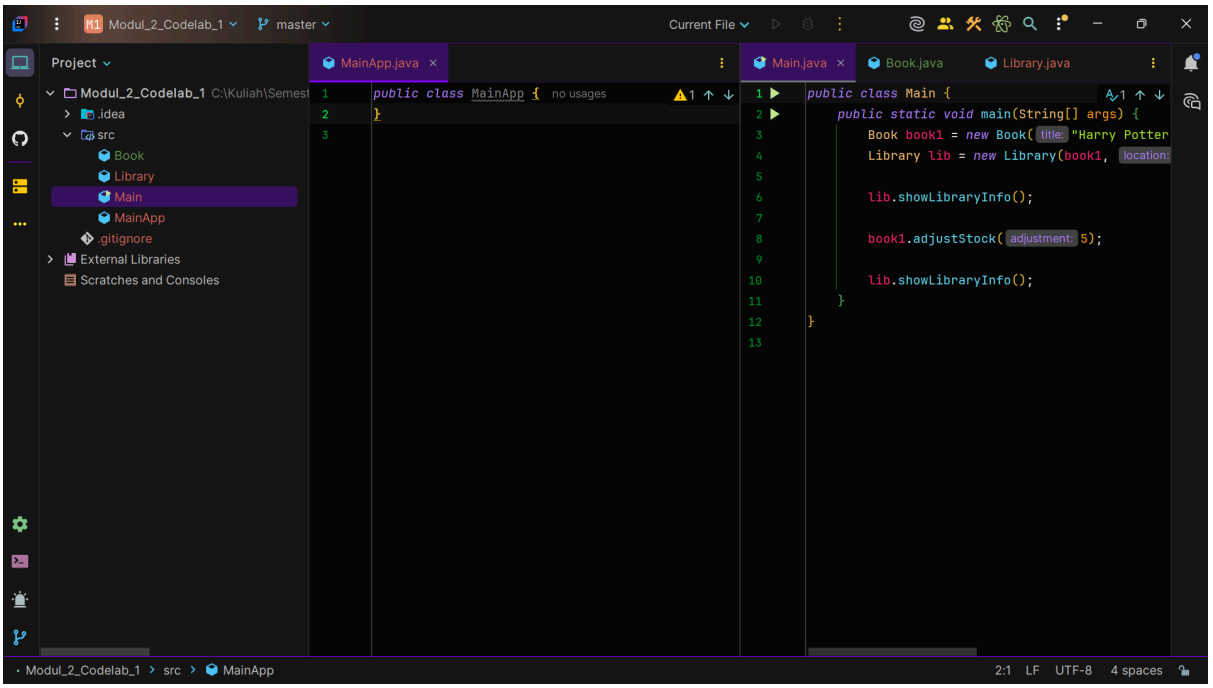
BEFORE



Method **main()** masih berada pada file/class MainApp, untuk memindahkan method **main()** saya menggunakan refactoring **Move Method / Move Member.**



AFTER



Method **main()** yang semula ada di class **MainApp** sekarang sudah berpindah ke class/file **Main.**