LAPORAN PRAKTIKUM CODELAB PEMROGRAMAN LANJUT

MODUL 2

Nama: Muhammad Ibrahim Al Ayubi
NIM: 202410370110123
Kelas: Pemrograman Lanjut E

CODELAB:

CODELAB 1

```java
// Class Book to store book information
class Book {
    public String title;
    public String author;
    public double price;
    public int stock;

    // Constructor
    Book(String title, String author, double price, int stock) {
        this.title = title;
        this.author = author;
        this.price = price;
        this.stock = stock;
    }

    // Display book details
    public void displayInfo() {
        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);
    }

    // Adjust the book stock
    public void adjustStock(int adjustment) {
        stock += adjustment;
        System.out.println("Stock adjusted.");
        System.out.println("Current stock: " + stock);
    }
}
```

```java
// Class Library to store library location and a book
class Library {
    public Book book;
    public String location;

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

    // Display library and book information
    public void showLibraryInfo() {
        System.out.println("Library Location: " + location);
        book.displayInfo();
    }
}
```

```java
class MainApp {
    public static void main(String[] args) {
        Book book1 = new Book("Harry Potter", "J.K. Rowling", 10,2);
        Library lib = new Library(book1, "Perpustakaan Kota");

        // Display initial information
        lib.showLibraryInfo();

        // Add more stock
        book1.adjustStock(5);

        // Display updated information
        lib.showLibraryInfo();
    }
}
```

Program ini digunakan untuk mengelola data buku dan perpustakaan. Namun, ada beberapa bagian kode yang masih perlu diperbaiki supaya lebih jelas dan mudah dipahami. Berikut rencana refactoring yang akan dilakukan :

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. (**Clue: Encapsulate Field**)

2. Perkenalkan sebuah konstanta baru di Class Book untuk menyimpan nilai diskon (misalnya DISCOUNT_RATE = 0.1). (Clue: **Introduce Constant**)

3. Pisahkan perhitungan harga diskon dari displayInfo() menjadi sebuah metode baru di kelas Book dengan nama calculateDiscount(). (**Clue: Extract Method**)
    a. Before

```
// Class Book
// Display book details
  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() * DISCOUNT_RATE)));
    System.out.println("Stock: " + getStock());
  }
```

    b. After

```
// Class Book
// Display book details
public void displayInfo() {
  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());
}
```

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

LANGKAH-LANGKAH:

1. Salin kode ke intellij IDE

   Class Book

```java
public class Book {  4 usages
    public String title;  2 usages
    public String author;  2 usages
    public double price;  4 usages
    public int stock;  4 usages
    Book(String title, String author, double price, int stock) {  1 usag
        this.title = title;
        this.author = author;
        this.price = price;
        this.stock = stock;
    }
    public void displayInfo(){  1 usage
        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+"\n");
    }
    public void adjustStock(int adjustment){  1 usage
        stock += adjustment;
        System.out.println("Stock adjusted.");
        System.out.println("Current Stock: "+stock+"\n");
    }
}
```

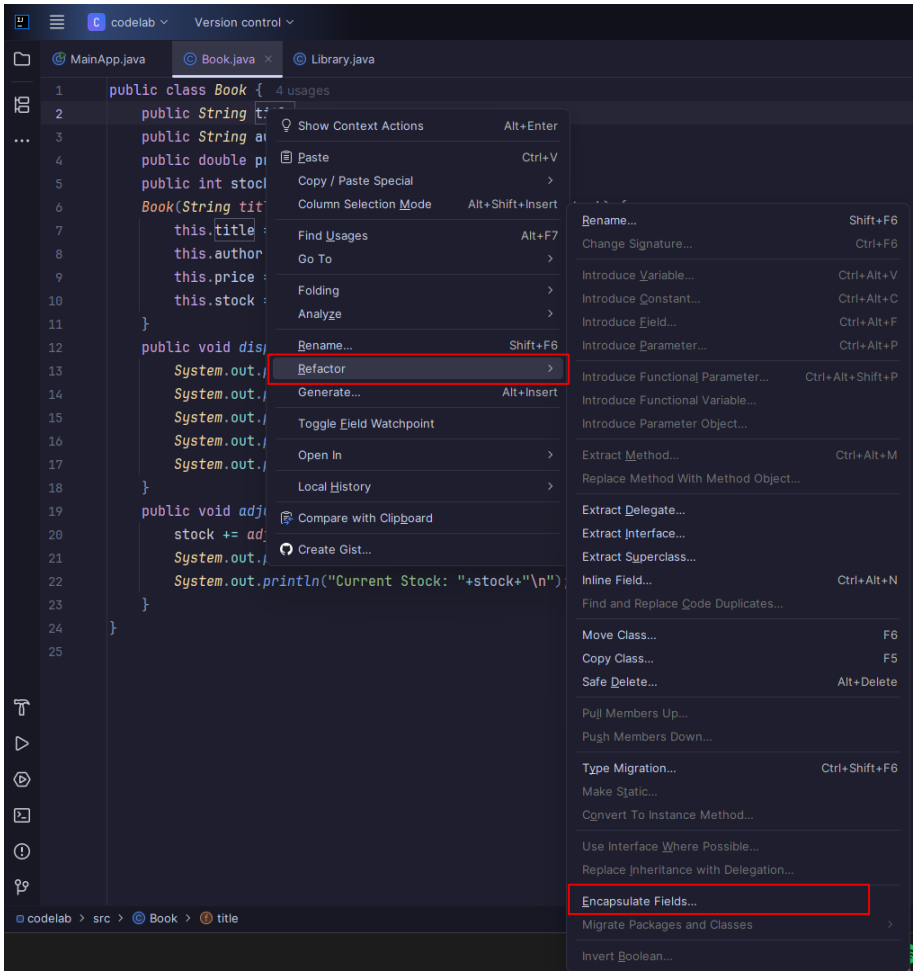
   Class Library

```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();
    }
}
```

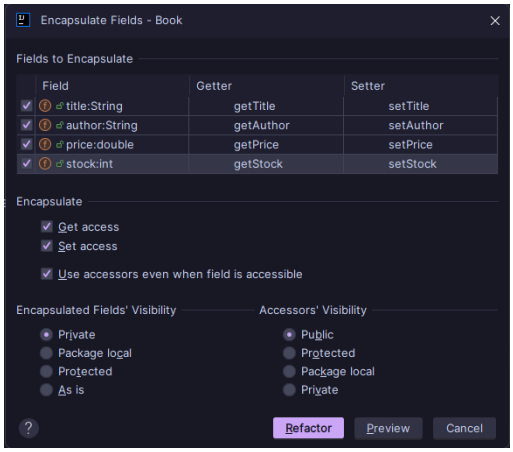
   Class MainApp

```java
public class MainApp {
    public static void main(String[] args) {
        Book book1 = new Book( title: "Harry Potter", author: "J.K Rowling", price: 10, stock: 2);
        Library lib = new Library(book1, location: "Perpustakaan Kota");
        lib.showLibraryInfo();
        book1.adjustStock( adjustment: 5);
        lib.showLibraryInfo();
    }
}
```

2. Selanjutnya kita akan melakukan encapsulate field di Class Book dan Class Library
    - Class Book
        - Klik kanan disalah satu attribut pilih Refactor lalu klik Encapsulate field



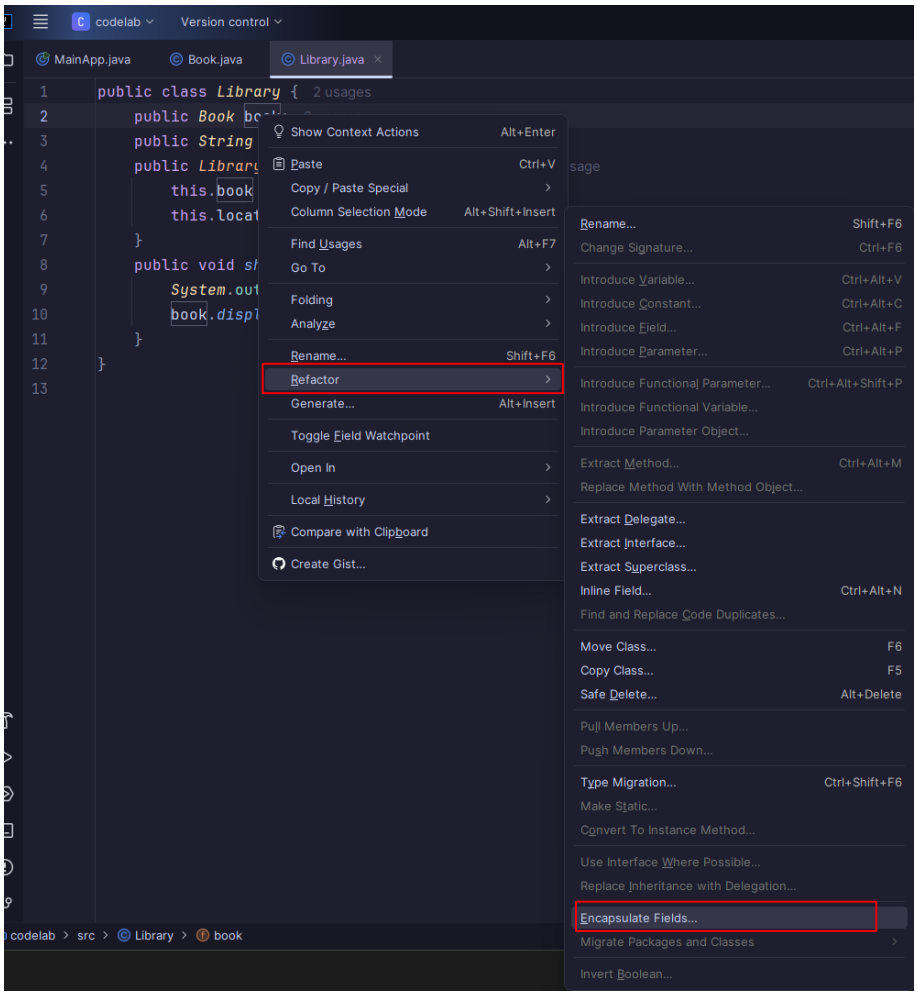
        - Ceklist field yang mau di encapsulate lalu klik refactor



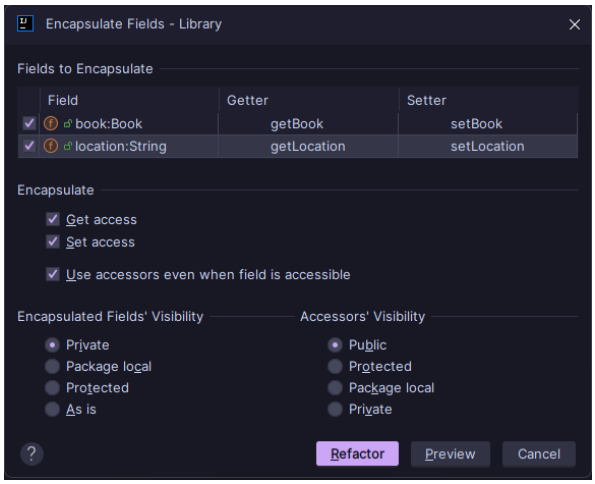
- Maka kode akan berubah menjadi seperti ini

```
public class Book {  4 usages
    private String title;  2 usages
    private String author;  2 usages
    private double price;  2 usages
    private int stock;  2 usages
    Book(String title, String author, double price, int stock) {  1 usage
        this.setTitle(title);
        this.setAuthor(author);
        this.setPrice(price);
        this.setStock(stock);
    }
    public void displayInfo(){  1 usage
        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() +"\n");
    }
    public void adjustStock(int adjustment){  1 usage
        setStock(getStock() + adjustment);
        System.out.println("Stock adjusted.");
        System.out.println("Current Stock: "+ getStock() +"\n");
    }
    public String getTitle() {  1 usage
        return title;
    }
    public void setTitle(String title) {  1 usage
        this.title = title;
    }
    public String getAuthor() {  1 usage
        return author;
    }
    public void setAuthor(String author) {  1 usage
        this.author = author;
    }
    public double getPrice() {  3 usages
        return price;
    }
    public void setPrice(double price) {  1 usage
        this.price = price;
    }
    public int getStock() {  3 usages
        return stock;
    }
    public void setStock(int stock) {  2 usages
        this.stock = stock;
    }
}
```

- Class Library
  - Klik kanan disalah satu attribut pilih Refactor lalu klik Encapsulate field



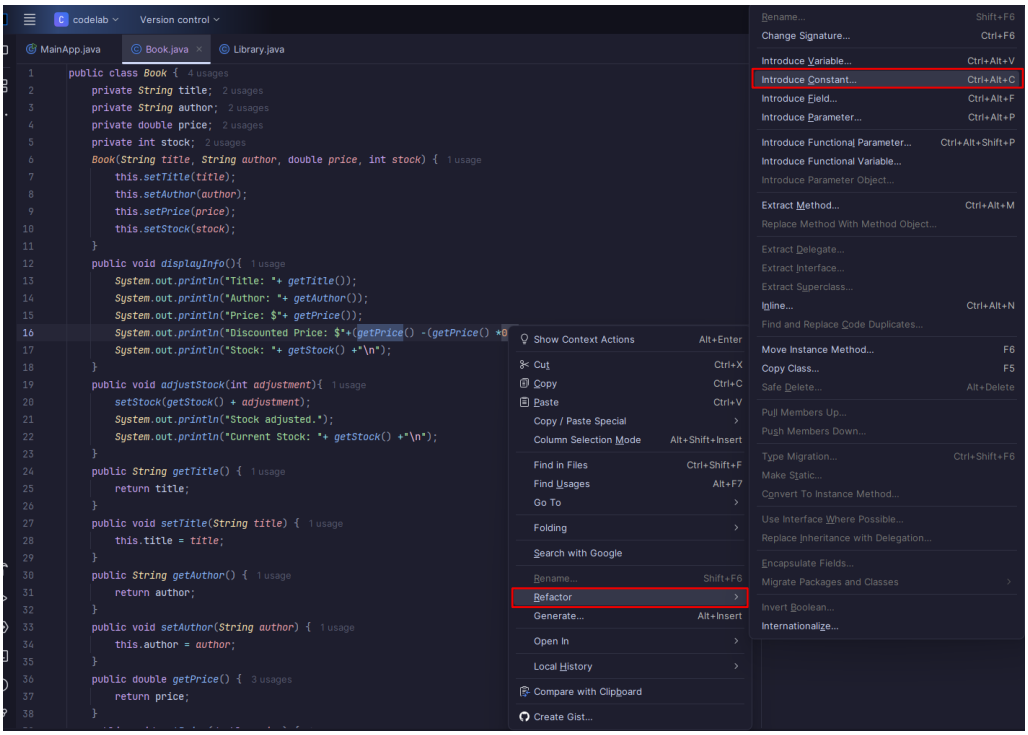
  - Ceklist field yang mau di encapsulate lalu klik refactor



- Maka kode akan berubah menjadi seperti ini

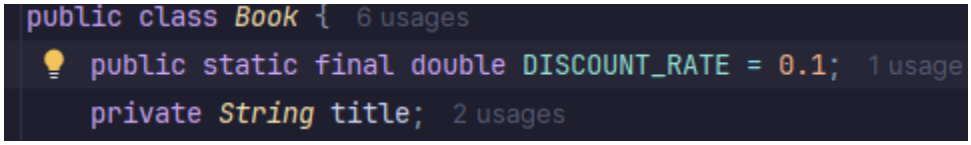
```java
public class Library {  2 usages
    private Book book;  2 usages
    private String location;  2 usages
    public Library(Book book, String location) {  1 usage
        this.setBook(book);
        this.setLocation(location);
    }
    public void showLibraryInfo(){  2 usages
        System.out.println("Library 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;
    }
}
```

3. Lalu kita akan mengubah perhitungan diskon yang ada di class buku dengan menggunakan Introduce Constant untuk menyimpan nilai diskon
    - Blok angka diskon (0.1) untuk menghitung diskon > klik kanan > refactor > introduce constant

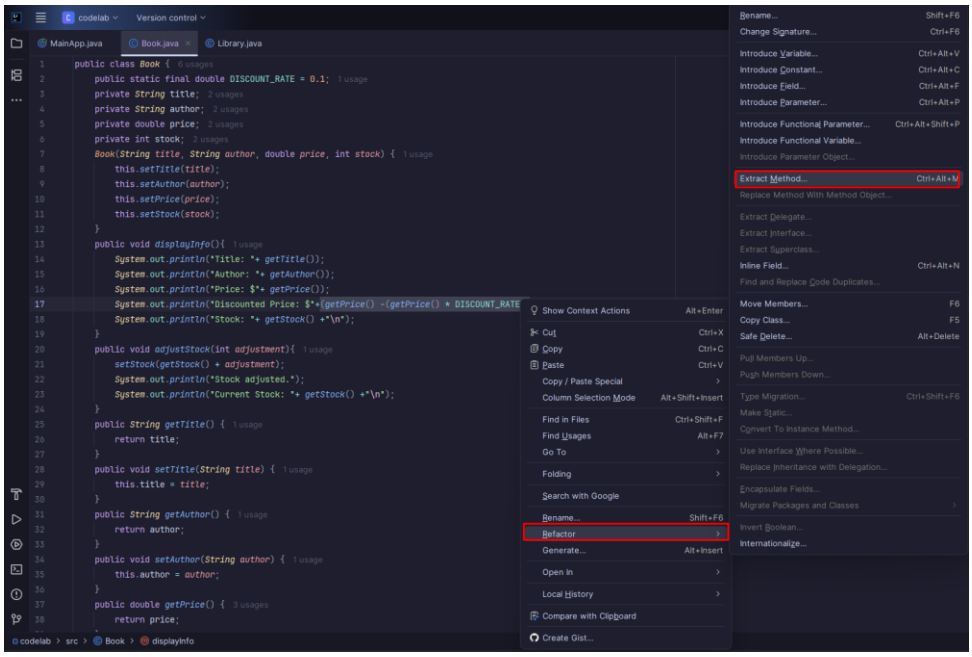


- Ubah menjadi DISCOUNT_RATE

- Secara otomatis akan terbuat variable constant



4. Lalu kita akan memisahkan perhitungan harga diskon dari displayInfo () menjadi metode baru di kelas book
    - Blok perhitungan diskon > klik kanan > refactor > Extract Method

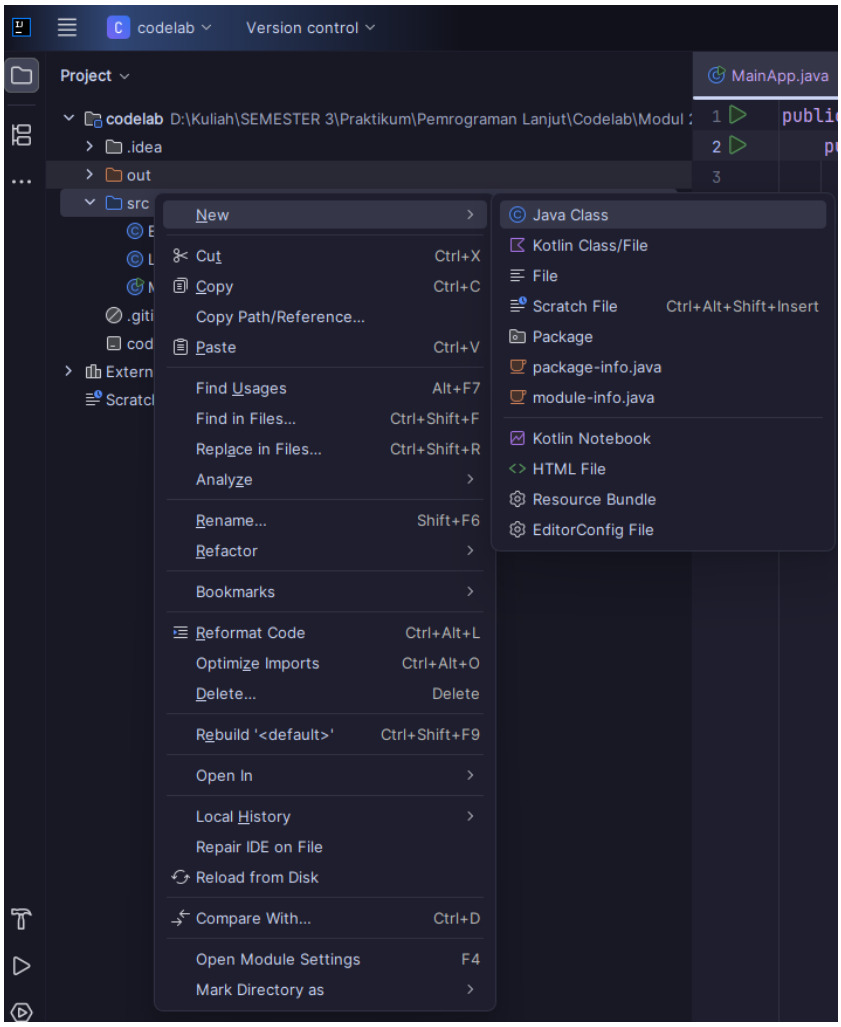


- Ubah menjadi calculateDiscount

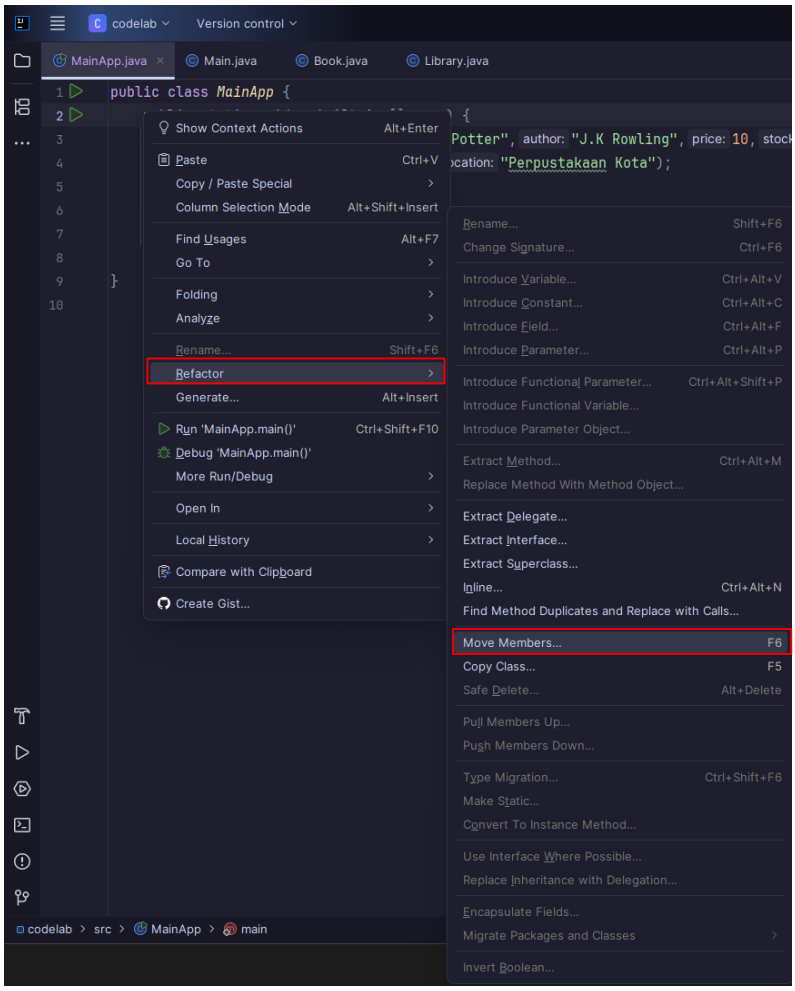
- Secara otomatis akan membuat method baru

```
private double calculateDiscount() {
    return getPrice() - (getPrice() * DISCOUNT_RATE);
}
```

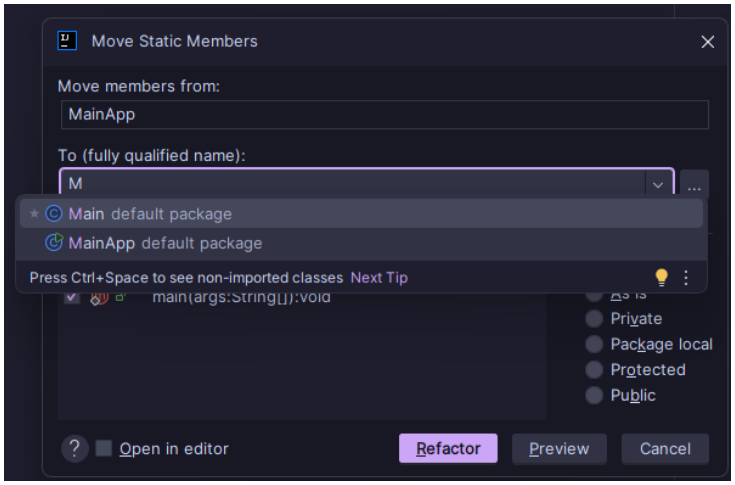
5. Terakhir kita akan memindahkan method main () dari class MainApp ke dalam kelas baru Main ()
    - Buat kelas main terlebih dahulu



- Lalu balik lagi ke class MainApp () klik kanan method main > refactor > move members



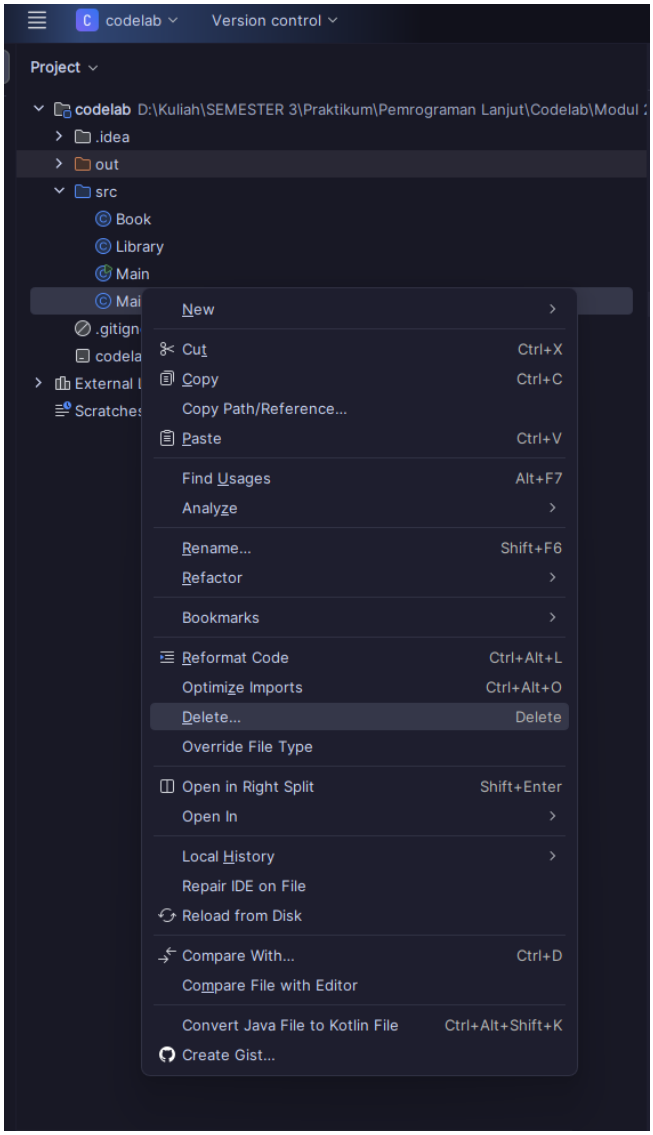
- Ketik nama Class tujuan “Main” lalu klik refactor



- Maka secara otomatis akan pindah ke class Main

```
public class Main {
    public static void main(String[] args) {
        Book book1 = new Book( title: "Harry Potter", author: "J.K Rowling", price: 10, stock: 2);
        Library lib = new Library(book1, location: "Perpustakaan Kota");
        lib.showLibraryInfo();
        book1.adjustStock( adjustment: 5);
        lib.showLibraryInfo();
    }
}
```

- Hapus Class MainApp



6. Maka setelah melakukang refactoring maka kode akan terlihat seperti ini
   Class Book

```java
public class Book {  6 usages
    public static final double DISCOUNT_RATE = 0.1;  1 usage
    private String title;  2 usages
    private String author;  2 usages
    private double price;  2 usages
    private int stock;  2 usages
    Book(String title, String author, double price, int stock) {  1 usage
        this.setTitle(title);
        this.setAuthor(author);
        this.setPrice(price);
        this.setStock(stock);
    }
    public void displayInfo(){  1 usage
        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() +"\n");
    }
    private double calculateDiscount() {  1 usage
        return getPrice() - (getPrice() * DISCOUNT_RATE);
    }
    public void adjustStock(int adjustment){  1 usage
        setStock(getStock() + adjustment);
        System.out.println("Stock adjusted.");
        System.out.println("Current Stock: "+ getStock() +"\n");
    }
    public String getTitle() {  1 usage
        return title;
    }
    public void setTitle(String title) {  1 usage
        this.title = title;
    }
    public String getAuthor() {  1 usage
        return author;
    }
    public void setAuthor(String author) {  1 usage
        this.author = author;
    }
    public double getPrice() {  3 usages
        return price;
    }
    public void setPrice(double price) {  1 usage
        this.price = price;
    }
    public int getStock() {  3 usages
        return stock;
    }
    public void setStock(int stock) {  2 usages
        this.stock = stock;
    }
}
```

Class Library

```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;
    }
}
```

Class Main

```java
public class Main {
    public static void main(String[] args) {
        Book book1 = new Book( title: "Harry Potter", author: "J.K Rowling", price: 10, stock: 2);
        Library lib = new Library(book1, location: "Perpustakaan Kota");
        lib.showLibraryInfo();
        book1.adjustStock( adjustment: 5);
        lib.showLibraryInfo();
    }
}
```

Output

```
Run  Main
C:\Users\macho\.jdks\openjdk-24.0.1\bi
Library Location: Perpustakaan Kota
Title: Harry Potter
Author: J.K Rowling
Price: $10.0
Discounted Price: $9.0
Stock: 2

Stock adjusted.
Current Stock: 7

Library Location: Perpustakaan Kota
Title: Harry Potter
Author: J.K Rowling
Price: $10.0
Discounted Price: $9.0
Stock: 7
```

Kesimpulan

Pada praktikum ini, kegiatan refactoring dilakukan untuk meningkatkan struktur dan keterbacaan kode program tanpa mengubah perilaku fungsionalnya. Proses refactoring diterapkan pada kode sederhana yang terdiri dari tiga kelas utama: Book, Library, dan MainApp.

Langkah-langkah refactoring yang dilakukan meliputi:

1. Encapsulation Field
   Penerapan encapsulation pada kelas Book dan Library untuk melindungi data (atribut) dari akses langsung, serta meningkatkan keamanan dan modularitas kode.
2. Introduce Constant
   Penggunaan konstanta DISCOUNT_RATE menggantikan nilai diskon numerik (magic number 0.1) agar kode lebih mudah dipelihara dan dimengerti.
3. Extract Method
   Pemisahan logika perhitungan diskon dari metode displayInfo() menjadi metode baru calculateDiscount() untuk meningkatkan keteraturan dan prinsip single responsibility.
4. Move Method / Move Members
   Pemindahan fungsi main() dari MainApp ke kelas baru Main, sehingga struktur proyek menjadi lebih bersih dan terorganisasi.

Secara keseluruhan, hasil refactoring menghasilkan kode yang:

- Lebih terstruktur dan mudah dibaca.
- Memenuhi prinsip OOP (Object-Oriented Programming) seperti Encapsulation dan Modularity.
- Lebih mudah dikelola, diuji, dan dikembangkan di masa mendatang.