

## **LAPORAN PRAKTIKUM CODELAB PROMLAN MODUL 2**



**NAMA : MUHAMMAD ARDHI PONTOH**  
**NIM : 202410370110290**  
**KELAS: 3D**

## CODELAB:

CODELAB 1

```

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);
    }
}

```

```

// 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();
    }
}

// Main Method
public class MainApp {
    public static void main(String[] args) {
        Book book1 = new Book("Harry Potter", "J.K. Rowling", 10.9, 5);
        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)
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)

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

a. Before

```

// Class Book
// Extract the discount
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());
}

```

## LANGKAH-LANGKAH:

## 1. Code awal

```

1 class MainApp {
2     public static void main(String[] args) {
3         Book book1 = new Book("Harry Potter", "J.K. Rowling", 10.9, 5);
4         Library lib = new Library(book1, "Perpustakaan Kota");
5
6         // Display Initial Information
7         System.out.println("--- Initial Book Information ---");
8         lib.showLibraryInfo();
9         System.out.println();
10
11        // Add more stock
12        book1.adjustStock(5);
13        System.out.println();
14
15        // Display updated information
16        System.out.println("--- Updated Book Information ---");
17        lib.showLibraryInfo();
18    }
19 }

```

```

1 class Library {
2     public Book book;
3     public String location;
4
5     public Library(Book book, String location) {
6         this.book = book;
7         this.location = location;
8     }
9
10    // Display library and book information
11    public void showLibraryInfo() {
12        System.out.println("Library Location: " + location);
13        book.displayInfo();
14    }
15 }

```

```

1 class Book {
2     public String title;
3     public String author;
4     public double price;
5     public int stock;
6
7     // Constructor
8     public Book(String title, String author, double price, int stock) {
9         this.title = title;
10        this.author = author;
11        this.price = price;
12        this.stock = stock;
13    }
14
15    // Display book details with hardcoded discount
16    public void displayInfo() {
17        System.out.println("Title: " + title);
18        System.out.println("Author: " + author);
19        System.out.println("Price: $" + price);
20        // "magic number" 0.1 for discount calculation
21        System.out.println("Discounted Price: $" + (price - (price * 0.1)));
22        System.out.println("Stock: " + stock);
23    }
24
25    // Adjust the book stock
26    public void adjustStock(int adjustment) {
27        stock += adjustment;
28        System.out.println("Stock adjusted.");
29        System.out.println("Current stock: " + stock);
30    }
31 }

```

## 2. Menerapkan Encapsulate Field

Field pada kelas Book (title, author, price, stock) dan Library (book, location) diubah dari public menjadi private. Kemudian, setter dan getter dibuat untuk setiap field agar akses terhadap data lebih terkontrol.

```
private String title; 2 usages
private String author; 2 usages
private double price; 2 usages
private int stock; 2 usages
```

```
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; }
```

```
private Book book; 2 usages
private String location; 2 usages
```

```
public Book getBook() { return book; }
public void setBook(Book book) { this.book = book; }
public String getLocation() { return location; }
public void setLocation(String location) { this.location = location; }
```

### 3. Menerapkan Introduce Constant

Nilai diskon 0.1 yang sebelumnya ditulis langsung (hardcoded) di dalam method displayInfo, diekstrak menjadi sebuah konstanta DISCOUNT\_RATE. Hal ini membuat kode lebih mudah dibaca.

```
public static final double DISCOUNT_RATE = 0.1;
```

```
System.out.println("Discounted Price: $" + (getPrice() - (getPrice() * DISCOUNT_RATE)));
```

### 4. Menerapkan Extract Method

Logika perhitungan harga diskon dipisahkan dari metode displayInfo() ke dalam metode baru bernama calculateDiscount(). Ini membuat metode displayInfo() lebih ringkas dan fokus pada tugasnya, yaitu menampilkan informasi.

```
System.out.println("Discounted Price: $" + calculated_discount());
System.out.println("Stock: " + getStock());
}

private double calculated_discount() { 1 usage
    return getPrice() - (getPrice() * DISCOUNT_RATE);
}
```

### 5. Menerapkan Move Method

Metode main() dipindahkan dari kelas MainApp ke kelas baru bernama Main. Kelas MainApp kemudian dihapus. Langkah ini bertujuan untuk mengorganisasi kode dengan lebih baik, di mana kelas Main secara khusus berfungsi sebagai titik masuk (entry point) program.

```
1 public class Main {
2     public static void main(String[] args) {
3         Book book1 = new Book( title: "Harry Potter", author: "J.K. Rowling", price: 10.0, stock: 2);
4         Library lib = new Library(book1, location: "Perpustakaan Kota");
5
6         // Display initial information
7         System.out.println("--- Initial Book Information ---");
8         lib.showLibraryInfo();
9         System.out.println();
10
11        // Add more stock
12        book1.adjustStock( adjustment: 5);
13        System.out.println();
14
15        // Display updated information
16        System.out.println("--- Updated Book Information ---");
17        lib.showLibraryInfo();
18    }
19 }
```

## 6. Jalankan Program

Setelah semua proses refactoring selesai, program dijalankan dari kelas Main. Output yang dihasilkan tetap sama seperti sebelum refactoring, yang membuktikan bahwa perubahan yang dilakukan tidak mengubah fungsionalitas program.

```
"C:\Program Files\Java\jdk-17\bin\java.exe" "-javaagent:C:\Users\Alinno faza pratama\IntelliJ IDEA Community Editio
--- Initial Book Information ---
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

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

Process finished with exit code 0
```

## 7. Kesimpulan

Program ini berhasil di-refactor dengan menerapkan empat teknik utama untuk meningkatkan kualitas kode:

- **Encapsulate Field:** Meningkatkan keamanan data dan fleksibilitas dengan menyembunyikan implementasi internal.
- **Introduce Constant:** Meningkatkan keterbacaan dan kemudahan pemeliharaan dengan menghilangkan magic numbers.
- **Extract Method:** Membuat kode lebih bersih dan mudah dipahami dengan memecah metode yang kompleks menjadi bagian-bagian yang lebih kecil.
- **Move Method:** Meningkatkan struktur dan kohesi program dengan menempatkan fungsionalitas pada kelas yang paling sesuai.