

# **LAPORAN PRATIUM CODELAB PBO 2E**

## **MODUL 4**



Nama: Maulana Rayhan Zulkarnaen

NIM: 202410370110170

Kelas: PBO 2E

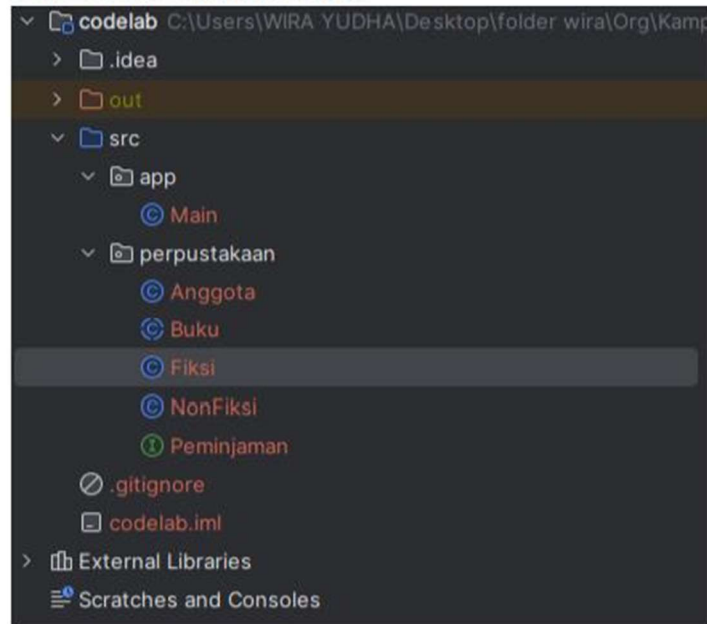
Github: [https://github.com/RAYHAN2006/PBO\\_2E\\_RAYHAN\\_170](https://github.com/RAYHAN2006/PBO_2E_RAYHAN_170)

## CODELAB:

### CODELAB

Buatlah sistem manajemen perpustakaan sederhana yang menerapkan konsep-konsep dasar pemrograman berorientasi objek dalam Java, yaitu **Package**, **Polymorphism**, **Overloading**, **Interface**, dan **Abstraction**.

1. Semua kelas harus disimpan dalam package perpustakaan (kecuali **Main.java**). Berikut adalah struktur foldernya:



2. Kelas **Buku** harus dibuat sebagai kelas **abstrak** dengan atribut **judul** dan **penulis**, serta memiliki method abstrak **displayInfo()**.
3. Kelas **Buku** harus memiliki dua subclass: **Fiksi** dan **NonFiksi**, di mana masing-masing subclass mengimplementasikan method **displayInfo()** dengan cara yang berbeda.
4. Buatlah interface **Peminjaman** yang memiliki dua method: **pinjamBuku()** dan **kembalikanBuku()**. Kelas **Anggota** harus mengimplementasikan interface ini untuk mencetak keterangan **peminjaman** atau **pengembalian**.
5. Dalam kelas **Anggota**, buatlah method **pinjamBuku()** yang memiliki dua versi, satu menerima parameter berupa **judul buku**, dan satu lagi menerima parameter berupa **judul** dan **durasi peminjaman**. Kemudian buat 2 atribut yaitu:
  - o **String** : nama
  - o **String** : idAnggota

## 6. Contoh output yang diharapkan:

```

Buku Non-Fiksi: Madilog oleh Tan Malaka (Bidang: Sejarah & Ilmu Pengetahuan )
Buku Fiksi: Hainuwele: Sang Putri Kelapa oleh Lilis Hu (Genre: Dongeng)

Anggota: Wahyu Andika (ID: B075)
Anggota: Ega Faiz (ID: A047)

Wahyu Andika meminjam buku berjudul: Madilog
Ega Faiz meminjam buku "Hainuwele: Sang Putri Kelapa" selama 7 hari.

Wahyu Andika mengembalikan buku berjudul: Madilog
Ega Faiz mengembalikan buku berjudul: Hainuwele: Sang Putri Kelapa

Process finished with exit code 0

```

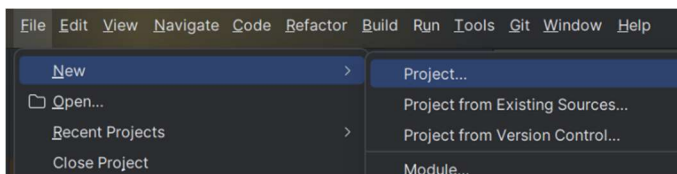
## 7. Catatan:

- **idAnggota** merupakan **kelas kalian** dan **3-digit nim terakhir** kalian dan teman kalian
- **Nama** menggunakan **nama kalian** dan **teman kalian**
- Bagi yang tidak mengumpulkan codelab pada saat praktikum, kumpulkan di llab dalam bentuk laporan (pdf). Laporan harus menjelaskan step by step pengerjaan codelab. Cantumkan link repository kalian yang sudah kalian list di Spreadsheet modul 1 kemarin. Contoh laporan dapat dilihat [disini](#).

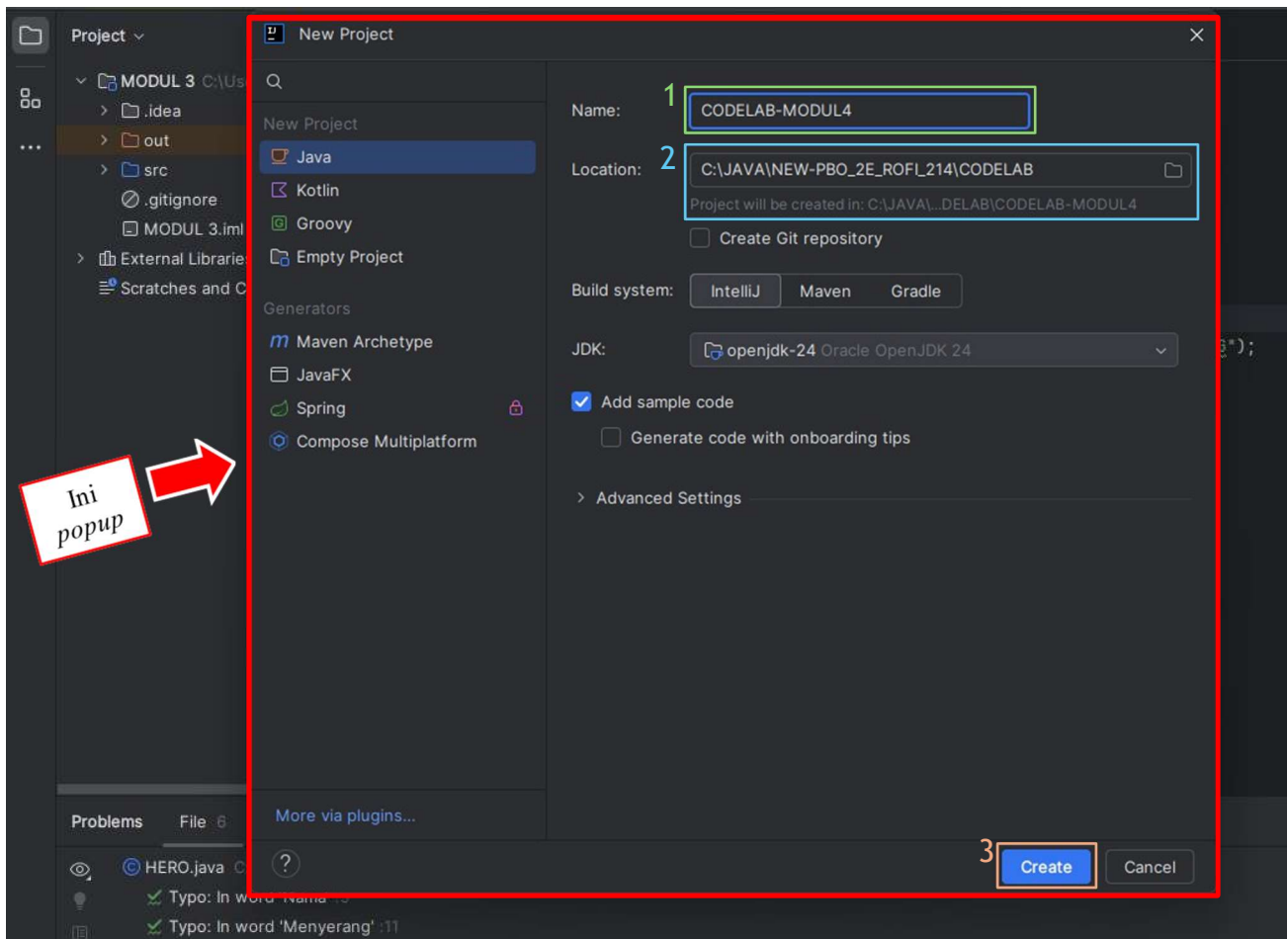
## LANGKAH-LANGKAH:

1. Membuat *New Project*

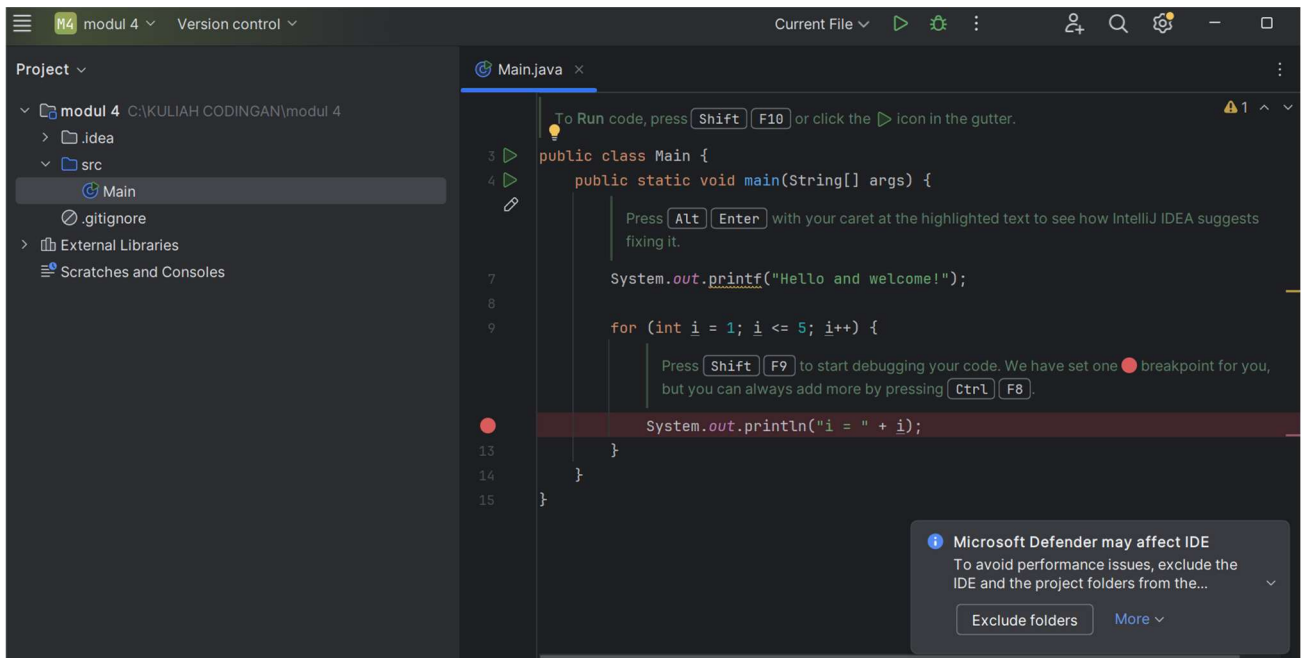
- Langkah pertama yang harus di lakukan ialah membuat *Project* Baru. Caranya dengan menekan tombol *New Project* seperti contoh gambar di bawah ini:



- Setelah itu, muncul *Popup* silakan menamai *project* anda di **Name: [1]**, menyimpan project anda di **Location: [2]**, untuk penyimpanan project silakan ikuti contoh yang ada di Modul 1, lalu tekan **Create [3]**:

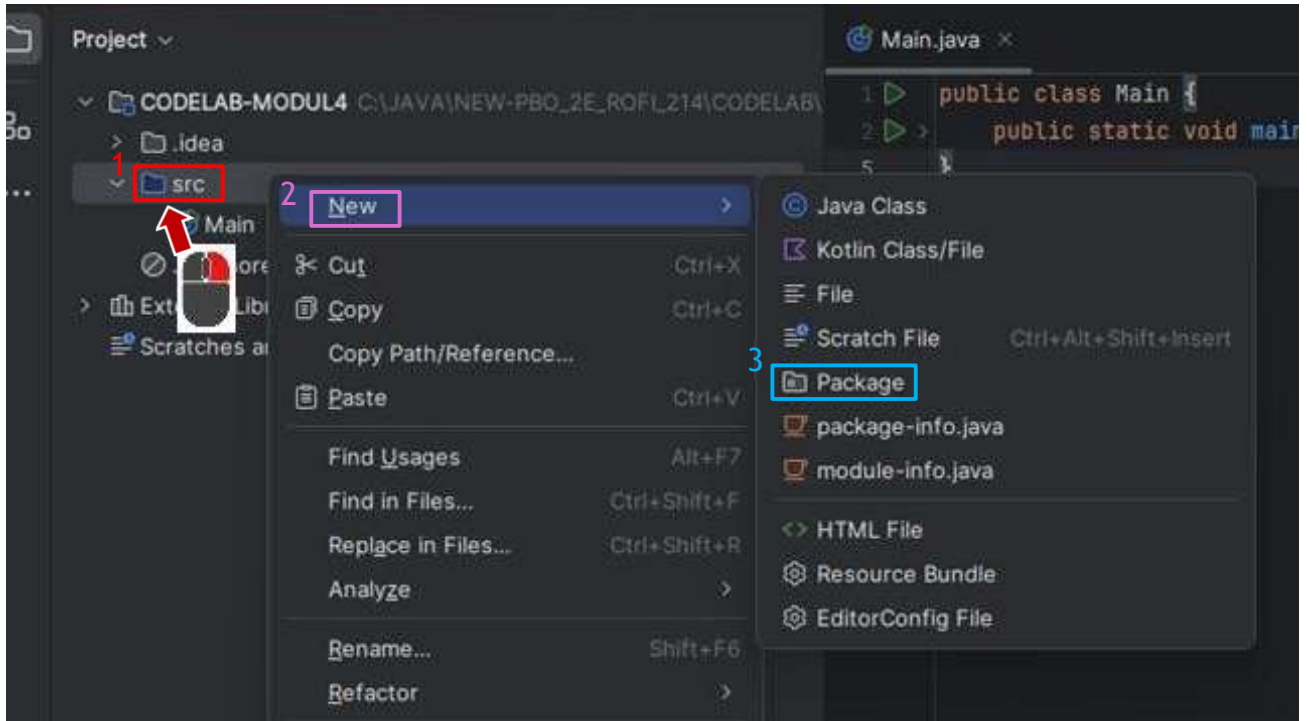


- Tampilan setelah membuat *project* baru:

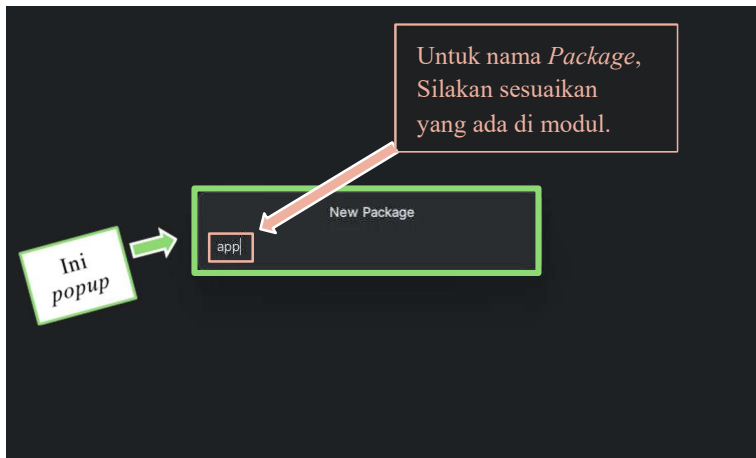


## 2. Membuat Package dan Class

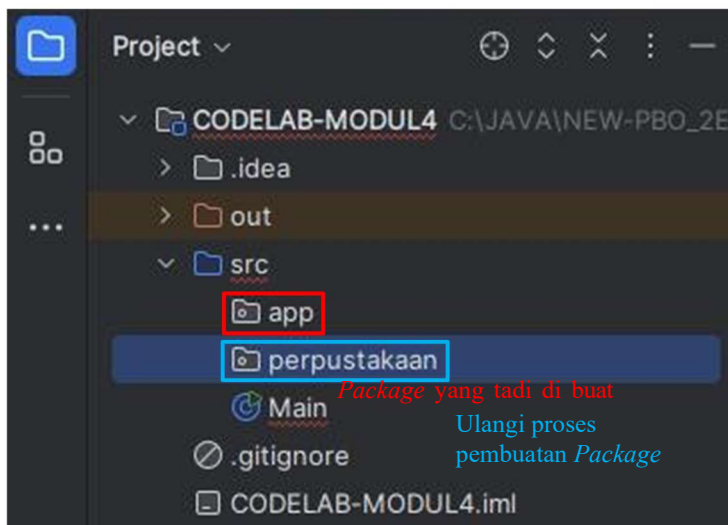
- untuk membuat *Package* klik kanan pada *folder src* [1], klik **New** [2], lalu klik lagi **package** [3]. Nanti muncul *popup* untuk mengisi nama *package* tersebut. Silakan namai *package* tersebut sesuai yang ada di dalam modul:



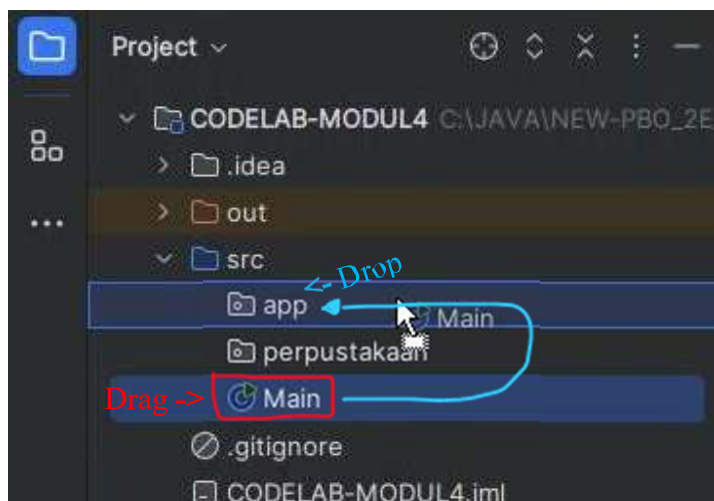
- Namain *Package* tersebut



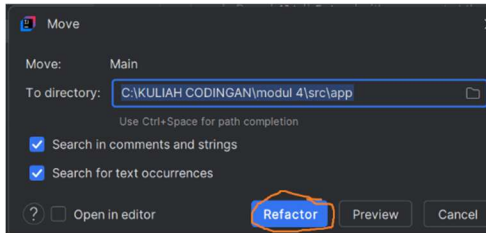
- *Package app* telah dibuat, ulangi proses ini untuk membuat *Package Perpustakaan*



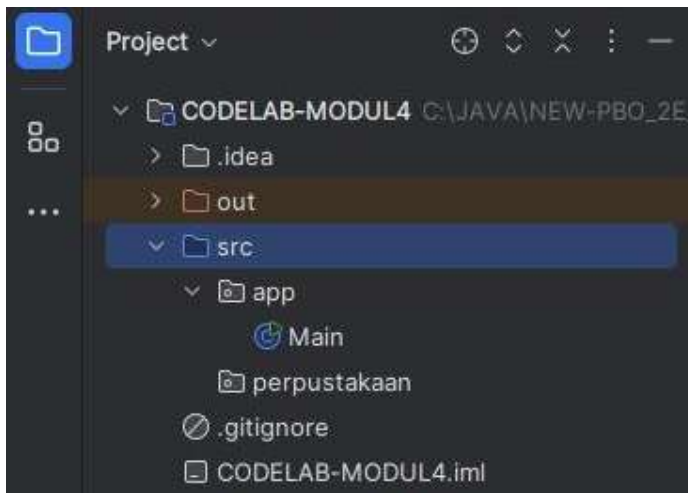
- Sebelum membuat *Class*, *Class Main* yang sudah ada dari awal, dipindahkan dulu ke *Package app*, dengan cara *drag and drop Class Main ke Package app*, nanti muncul *popup Move*, lalu klik *Refactor*, *Class Main* sudah di *Package app* seperti gambar di bawah ini :



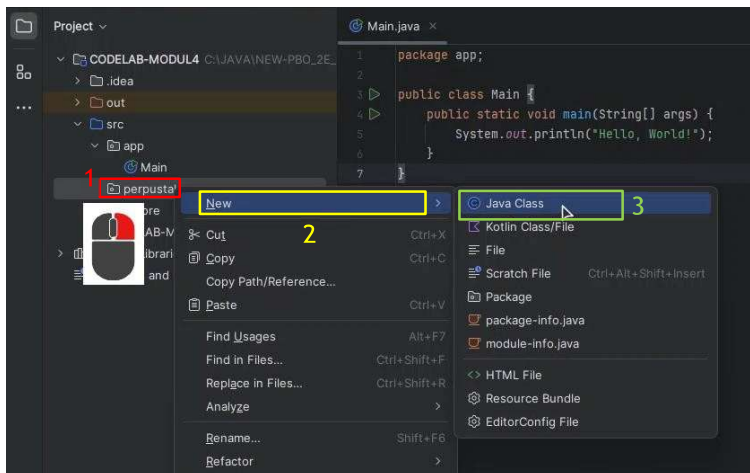
- Klik Refactor



- *Class Main* sudah di *package app*:
- Untuk membuat *class*, **klik kanan** pada *package* yang mau di isi class **[1]**, misalnya pada *package*

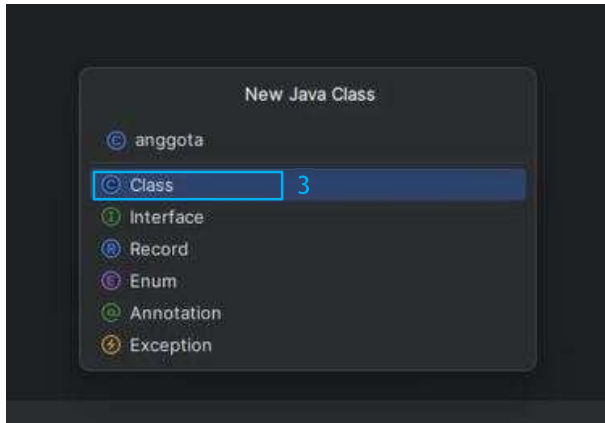


*perpustakaan*, klik **New [2]**, klik **Java Class [3]**, muncul popup lalu pilih **Class [4]** dan namai *class* tersebut sesuai yang ada di modul, lalu tekan **Enter**:

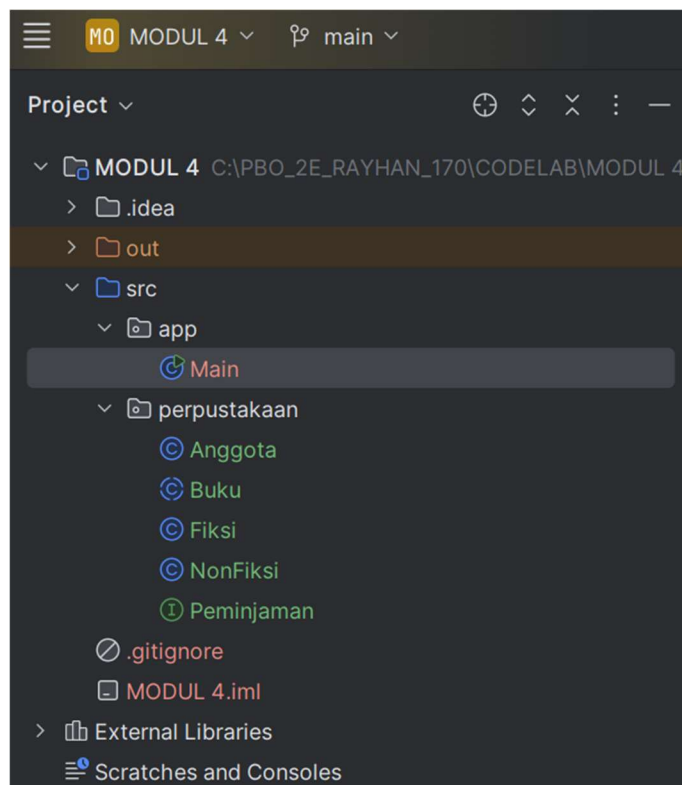




- Lalu tekan **ENTER**



- Ulangi Langkah tadi sampai seperti gambar dibawah ini:





### 3. Implementasi codelab

#### a. Class buku

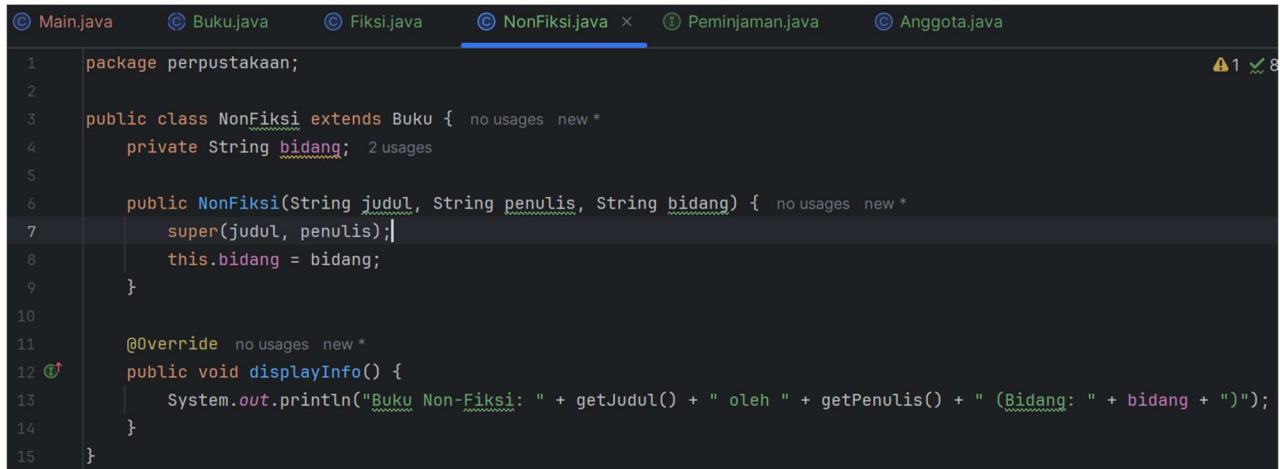
- 1) Buat kelas abstrak `Buku` dengan atribut `judul` dan `penulis`.
- 2) Tambahkan method abstrak `displayInfo()`.

#### b. Subclass fiksi dan nonfiksi

1. Buat kelas `Fiksi` dan `NonFiksi` yang mewarisi `Buku`.
2. Tambahkan atribut khusus:  
 Fiksi: `genre`  
 NonFiksi: `bidang`
3. Override method `displayInfo()` untuk menampilkan info spesifik.

##### 1. Fiksi

## 2. nonfiksi



```

1 package perpustakaan;
2
3 public class NonFiksi extends Buku { no usages new *
4     private String bidang; 2 usages
5
6     public NonFiksi(String judul, String penulis, String bidang) { no usages new *
7         super(judul, penulis);
8         this.bidang = bidang;
9     }
10
11     @Override no usages new *
12     public void displayInfo() {
13         System.out.println("Buku Non-Fiksi: " + getJudul() + " oleh " + getPenulis() + " (Bidang: " + bidang + ")");
14     }
15 }

```

## 4. Class peminjaman jadikan interface

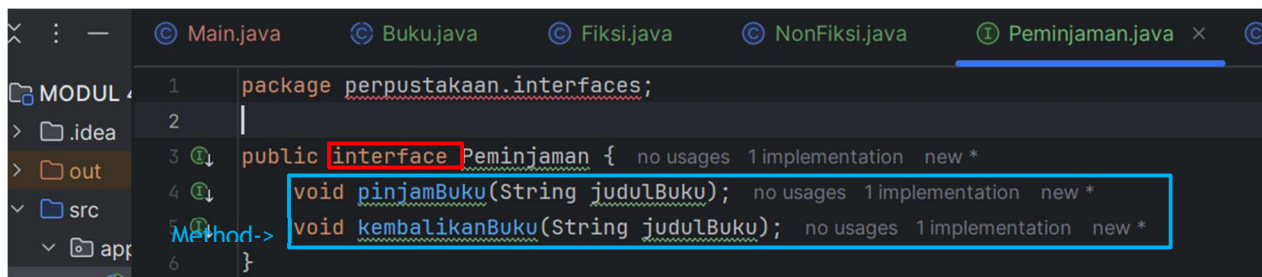


```

1 package perpustakaan.interfaces;
2
3 public interface Peminjaman { no usages 1 implementation new *
4     void pinjamBuku(String judulBuku); no usages 1 implementation new *
5     void kembalikanBuku(String judulBuku); no usages 1 implementation new *
6 }

```

- 1) kata *class* disini, diganti menjadi kata *interface*. Gunanya untuk menentukan operasi wajib untuk proses di peminjaman buku.



```

1 package perpustakaan.interfaces;
2
3 public interface Peminjaman { no usages 1 implementation new *
4     void pinjamBuku(String judulBuku); no usages 1 implementation new *
5     void kembalikanBuku(String judulBuku); no usages 1 implementation new *
6 }

```

- 2) Definisikan *method* pinjamBuku() dan kembalikanBuku()

### 5. Class **anggota**, mengimplementasi *interface* **peminjaman**

- 1) Implementasikan **interface** **peminjaman**.
- 2) Tambahkan **atribut** **nama** dan **idAnggota**
- 3) Tambahkan **Constructor** untuk inisialisasi atribut.
- 4) **Override method** dari interface dengan logika peminjaman dan pengembalian.

```

1 package perpustakaan;
2
3 import perpustakaan.interfaces.Peminjaman;
4
5 public class Anggota implements Peminjaman {
6     private String nama;
7     private String idAnggota;
8
9     public Anggota(String nama, String idAnggota) {
10         this.nama = nama;
11         this.idAnggota = idAnggota;
12     }
13
14     @Override
15     public void pinjamBuku(String judulBuku) {
16         System.out.println(nama + " meminjam buku berjudul: " + judulBuku);
17     }
18
19     @Override
20     public void kembalikanBuku(String judulBuku) {
21         System.out.println(nama + " mengembalikan buku berjudul: " + judulBuku);
22     }
23
24     public void pinjamBuku(String judulBuku, int durasiHari) {
25         System.out.println(nama + " meminjam buku \"" + judulBuku + "\" selama " + durasiHari + " hari.");
26     }
27 }
  
```

Annotations in the image:

- Interface**: Points to the `implements Peminjaman` part of the class declaration.
- Method**: Points to the private attributes `nama` and `idAnggota`.
- Constructor**: Points to the `Anggota` constructor method.
- Override method**: Points to the `@Override` annotated methods.

## 6. *Package* **app** *Class* **Main**, untuk menjalankan semua komponen di *Package* **perpustakaan**

```

1 package app;
2
3 import perpustakaan.*;
4
5 public class Main {
6     public static void main(String[] args) {
7         // Buat objek buku
8         Buku buku1 = new NonFiksi( judul: "Filosofi Teras", penulis: "Henry Manampiring", bidang: "Kesehatan mental");
9         Buku buku2 = new Fiksi( judul: "Perahu Kertas", penulis: "Dee Lestari", genre: "Romance");
10
11         // Tampilkan info buku
12         buku1.displayInfo();
13         buku2.displayInfo();
14         System.out.println();
15
16         // Buat objek anggota
17         Anggota anggota1 = new Anggota( nama: "Maulana Rayhan Zulkarnaen", idAnggota: "170");
18         Anggota anggota2 = new Anggota( nama: "Muhammad Syakhish Al Hanif", idAnggota: "189");
19
20         // Tampilkan info anggota
21         System.out.println("Anggota: " + anggota1.getNama() + " (ID: " + anggota1.getIdAnggota() + ")");
22         System.out.println("Anggota: " + anggota2.getNama() + " (ID: " + anggota2.getIdAnggota() + ")");
23         System.out.println();
24
25         // Contoh peminjaman dan pengembalian
26         anggota1.pinjamBuku( judulBuku: "Filosofi Teras");
27         anggota2.pinjamBuku( judulBuku: "Perahu Kertas", durasiHari: 7);
28         System.out.println();
29
30         anggota1.kembalikanBuku( judulBuku: "Filosofi Teras");
31         anggota2.kembalikanBuku( judulBuku: "Perahu Kertas");
32     }
33 }

```

- 1) Untuk mengakses *Package* **perpustakaan**, memakai **import perpustakaan;** untuk mengambil komposisi yang ada di *Package* **perpustakaan**, kalau mengambil semuanya komposisi tambahkan **.\***;
- 2) Instansiasi objek buku, anggota, peminjaman, dan pengembalian.
- 3) Panggil method untuk menampilkan output.

### **PENJELASAN:**

#### 1. Jenis-jenis Entitas

Entitas	Penjelasan	Implementasi
<b>Prime</b>	Entitas mandiri yang tidak bergantung pada entitas lain.	<i>Class</i> <b>buku</b> (abstract) dapat eksis tanpa memerlukan entitas lain.
<b>Dependent</b>	Entitas yang keberadaannya bergantung pada entitas lain.	<i>Class</i> <b>fiksi</b> dan <b>nonfiksi</b> bergantung pada induk <b>buku</b> (harus di- <i>extend</i> ).

## 2. Jenis-jenis Atribut

Artibut	Penjelasan	Implementasi
<b>Tunggal</b>	Atribut yang tidak bisa dipecah menjadi bagian lebih kecil.	<b>judul</b> (String), <b>penulis</b> (String) pada <i>class buku</i> .
<b>Nilai Tunggal</b>	Hanya menyimpan satu nilai per e ntitas.	<b>genre</b> (String) pada <i>class fiksi</i> . <b>bidang</b> (String) pada <i>class nonfiksi</i> .
<b>Mandatory</b>	Atribut wajib diisi tidak boleh kosong	<b>judul</b> dan <b>penulis</b> pada constructor <b>buku</b> .

## 3. Jenis-Jenis Relasi

Relasi	Penjelasan	Implementasi
<b>Inheritance</b>	Hubungan pewarisan antara parent class dan subclass.	<b>fiksi extends buku</b> , <b>nonfiksi extends buku</b> .
<b>Interface (Implements)</b>	Kontrak method yang wajib diimplementasikan class.	<b>anggota implements peminjaman</b> (wajib ada <b>pinjamBuku()</b> dan <b>kembalikanBuku()</b> ).
<b>One-to-Many</b>	Satu entitas terhubung ke banyak entitas lain.	Satu <i>class buku (abstract)</i> memiliki banyak subclass ( <b>fiksi</b> , <b>nonfiksi</b> ).

## 4. Artibut spesifik

Entitas	Artibut	Tipe data	Mandatory/Optional	Penjelasan
buku	Judul	String	Mandatory	Judul buku (unik).
Buku	Penulis	String	Mandatory	Nama penulis.
fiksi	Genre	String	Mandatory	Genre buku (Fantasi).
nonfiksi	Bidang	String	Mandatory	Bidang (ilmu, Sejarah, kesehatan, dll).
anggota	idAnggota	String	Mandatory	ID anggota (format: Huruf + 3 digit NIM).

## 5. Implementasi konsep OOP

Konsep	Penjelasn	Contoh kode
<b>Abstraction</b>	Menyembunyikan detail dengan abstract class.	<pre>public abstract class Buku {     private String judul;     private String penulis;      public Buku(String judul, String penulis) {         this.judul = judul;         this.penulis = penulis;     }     public abstract void displayInfo(); }</pre>
<b>Polymorphism</b>	Satu method dengan implementasi berbeda.	<p>Overriding di <i>class fiksi</i> maupun di <i>class nonfiksi</i></p> <pre>@Override public void displayInfo() {     System.out.println("Buku Fiksi: " +         getJudul() + " oleh " + getPenulis() + " "         "(Genre: " + genre + ")"); }</pre> <p>Overriding di <i>class anggota</i></p> <pre>public void kembalikanBuku(String judulBuku) {     System.out.println(nama + " mengembalikan buku berjudul: " + judulBuku); }  public void pinjamBuku(String judulBuku, int durasiHari) {     System.out.println(nama + " meminjam buku \"" + judulBuku + "\" selama " + durasiHari + " hari."); }</pre>
<b>Interface</b>	Kontrak method yang wajib diimplementasikan.	<pre>public interface Peminjaman {     void pinjamBuku(String judulBuku);     void kembalikanBuku(String judulBuku); }</pre>

## 6. Relasi dan Kardinalitas

Entitas 1	Entitas 2	Jenis Relasi	kardinalitas	Penjelasan
<b>buku</b>	<b>fiksi</b>	Inheritance	One-to-Many	Satu <i>class buku</i> punya banyak subclass.
<b>anggota</b>	<b>peminjaman</b>	Interface	One-to-One	Satu <b>anggota</b> mengimplementasikan satu <i>interface</i> .