



PEMROGRAMAN BERORIENTASI OBJEK

MODUL 2 – OOP, CLASS, OBJECT, METHOD, ATRIBUT

DISUSUN OLEH:

TAUFIQ RAMADHAN

SUTRISNO ADIT PRATAMA

DIAUDIT OLEH:

Ir. Galih Wasis Wicaksono, S.Kom, M.Cs.

PRESENTED BY: TIM LAB. IT

UNIVERSITAS MUHAMMADIYAH MALANG

PEMROGRAMAN BERORIENTASI OBJEK

TUJUAN

1. Mahasiswa mampu memahami konsep oop
 2. Mahasiswa mampu memahami konsep java class dan objects, methods (static dan non-static)
 3. Mahasiswa mampu memahami konsep dasar class diagram
 4. Mahasiswa mampu membangun aplikasi sederhana menggunakan paradigma object oriented.
-

TARGET MODUL

1. Mahasiswa mampu memahami & menerapkan Object Oriented Programming
 2. Mahasiswa mampu menggunakan syntax java class dan objects, methods (static dan non-static)
 3. Mahasiswa mampu mengimplementasikan dasar class diagram
-

PERSIAPAN

1. Java Development Kit.
 2. Text Editor / IDE (Visual Studio Code, Netbeans, IntelliJ IDEA, atau yang lainnya).
-

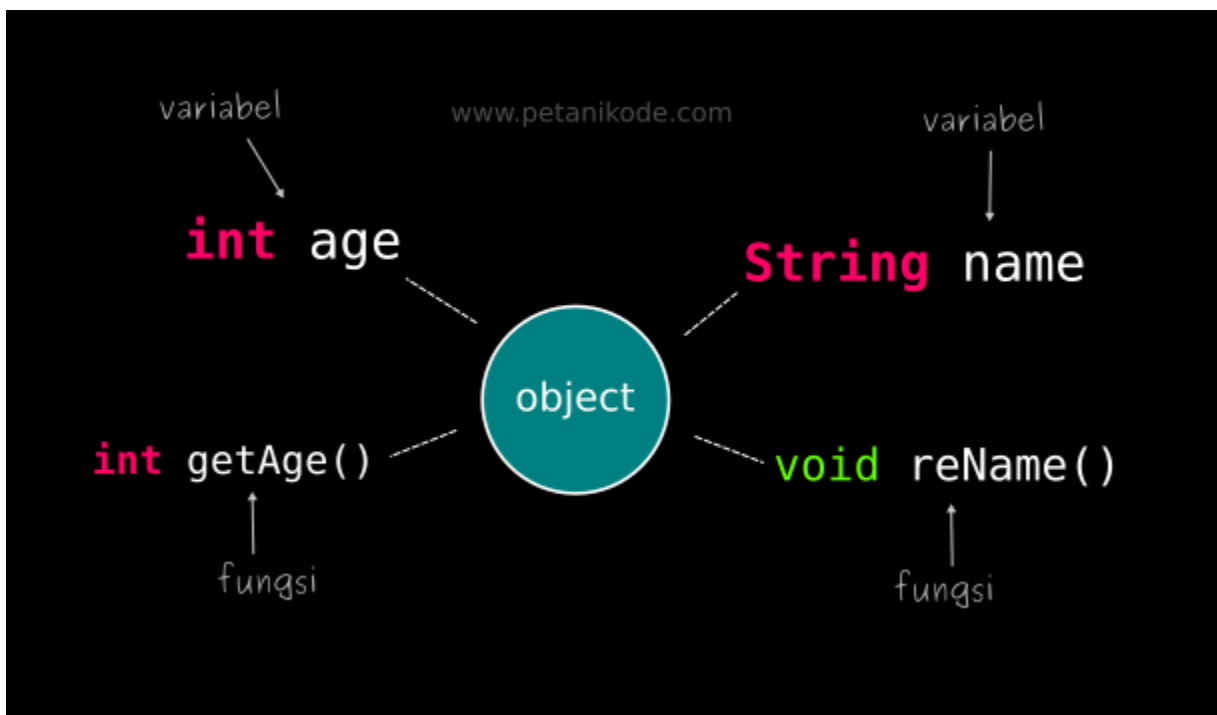
KEYWORDS

- OOP java
- Class
- Methods
- Object
- Static method
- Non-static method

TEORI

- **Konsep PBO**


Pemrograman Berorientasi Objek (OOP) adalah paradigma pemrograman yang berfokus pada objek. Objek adalah entitas yang menggabungkan data dan fungsi yang terkait. OOP membantu kita membuat program yang lebih terstruktur, mudah dipahami, dan mudah diubah. Program-program yang telah ada merupakan gabungan dari beberapa komponen-komponen kecil yang sudah ada sebelumnya. Hal itu dapat mempermudah pekerjaan seorang programmer dalam melakukan pengembangan program. Objek-objek yang saling berkaitan dan disusun kedalam satu kelompok ini disebut dengan class. Nantinya, objek-objek tersebut akan saling berinteraksi untuk menyelesaikan masalah program yang rumit.



- **Class**

Class adalah “blueprint” atau “cetakan biru” untuk menciptakan suatu object yang mendefinisikan struktur dan perilaku suatu objek dalam *object oriented programming* (OOP). Class sebenarnya bertugas untuk mengumpulkan fungsi/method dan variabel

dalam suatu tempat. Dalam contoh di sini kita ibaratkan class dengan sebuah mobil, yang artinya Mobil bisa memiliki sebuah variabel warna, nama, kecepatan dan juga bisa memiliki sebuah method untuk melakukan sesuatu seperti bergerak maju, mundur, dan berbelok. Contoh jika kita membuat sebuah class Mobil:



```
//class mobil
public class Mobil {

}
```

Class sebenarnya berisi **variabel** yang disebut sebagai **atribut** atau **properti** dan **fungsi** yang disebut sebagai **method**. Sebuah class berbeda dengan sebuah fungsi, class tidak memerlukan sebuah **()** tetapi langsung pada body class yang dimulai dengan **"{"** dan diakhiri dengan **"}"**.

- **Atribut**

Atribut merupakan identitas data atau informasi dari object class yang sudah kita buat, atau di bahasa C sebelumnya disebut sebagai variabel. Ada beberapa hal yang terkait dengan atribut yaitu:

1. Atribut adalah data atau properti yang dimiliki oleh sebuah class
2. Atribut dapat berupa variabel seperti nama, usia, warna, dan lain sebagainya
3. Atribut mendefinisikan keadaan suatu objek

Misalnya pada suatu mobil memiliki spesifikasi seperti merk, tahun, kecepatan, warna. Maka kita bisa menulis pada class Mobil seperti ini:

```
//class mobil
public class Mobil {
    //atribute mobil
    String merk;
    String model;
    int tahun;
    String spekMesin;
    String transmisi;
}
```

Atau bisa juga dengan cara seperti gambar di bawah, hal ini juga bisa dilakukan untuk mengurangi pengulangan kode:

```
//class mobil
public class Mobil {
    //atribute mobil
    String merk,
        model,
        spekMesin,
        transmisi;
    int tahun;
}
```

- **Method**

Method atau bisa juga disebut sebagai behavior adalah tindakan atau perilaku yang dapat dilakukan oleh sebuah objek yang menjadi instance dari sebuah class. Method merepresentasikan cara kerja atau bagaimana suatu objek dapat berjalan atau beroperasi. Contoh:

```

public class Mobil {
    String merk,
           model,
           spekMesin,
           transmisi;
    int tahun;
    void menyalakanMesin() {
        // untuk menghidupkan mesin mobil
        System.out.println("Mesin mobil Menyala");
    }

    String mengemudi(String arah) {
        // untuk menggerakkan mobil ke arah yang ditentukan
        return "Mobil bergerak ke arah " +arah;
    }

    String mengerem() {
        // untuk menghentikan mobil
        return "Berhenti";
    }

    int topSpeed(int topSpeed) {
        // untuk menentukan Top Speed dari sebuah mobil
        return topSpeed;
    }
}

```

Sebenarnya method sama dengan fungsi (di bahasa C) yang dapat melakukan sesuatu. Untuk contoh kode di atas method adalah sesuatu yang dapat dilakukan oleh instance objek mobil nantinya, berikut penjelasannya:

- `menyalakanMesin()` : adalah sebuah method yang dimana membuat sebuah instance nantinya bisa menyalakan mesin
- `mengemudi()` : menggerakkan mobil ke depan, belakang, dan juga berbelok
- `mengerem()` : menghentikan mobil

Karakteristik method:

- **Nama** : nama method sama dengan nama fungsi yang bisa digunakan untuk melakukan sesuatu. Dalam contoh di sini adalah sebuah mobil bisa melakukan `mengemudi()`, `mengerem()`, dan `menyalakanMesin()`.
- **Parameter** : parameter adalah sebuah data yang dilakukan input ketika sebuah method itu dipanggil. Dalam contoh di sini terdapat pada method **`mengemudi(String arah)`**, yang dimana *String arah* adalah sebuah parameter.

- Return type: return type menentukan data yang dihasilkan oleh sebuah method. Dalam contoh di sini adalah method **topSpeed(int topSpeed)**, yang dimana method ini *return* sebuah nilai integer.
- Body: body berisi kode yang akan dijalankan oleh sebuah method. Body dari sebuah method diawal dengan "{" dan diakhiri dengan "}".

Manfaat menggunakan method:

- Meningkatkan reusability kode. Contoh fungsi mengemudi() yang dapat digunakan berulang kali untuk menggerakkan mobil.
- Meningkatkan readability kode. Kode akan lebih terstruktur dengan baik sehingga membuat kode lebih mudah dibaca dan dipahami.
- Membantu membagi program menjadi bagian-bagian yang lebih kecil. Hal ini membuat program lebih mudah dikelola dan diubah.
- Meningkatkan maintainability. Program yang menggunakan method akan lebih mudah diperbaiki dan dipelihara pada waktu yang mendatang.

• Object

Object atau objek adalah hasil dari deklarasi atau instance dari sebuah class. Dalam objek, kita dapat mengakses dan memanipulasi isi dari sebuah class yang sudah kita buat sebelumnya. Sebagai contoh jika kita memiliki class Mobil sebelumnya, maka kita bisa membuat sebuah objek dari class tersebut di dalam method main seperti ini:

```
public class Mobil{
    String merk;
    String model;
    String spekMesin;
    int tahun;

    void menyalakanMesin(){
        System.out.println("Mesin mobil menyala");
    }

    String mengemudi(String arah){
        return "Mobil bergerak ke arah" + arah;
    }
}
```

```

String mengerem(){
    return "Berhenti";
}

int topSpeed(int topSpeed){
    return topSpeed;
}

public static void main(String[] args) {
    Mobil mobil = new Mobil();
}
}

```

Ketika kita jalankan programnya maka tidak akan ada sesuatu yang terjadi, karena kita hanya membuat sebuah instance objek saja dari class Mobil dengan nama objek **mobil**. Setelah kita membuat sebuah objek, kita bisa memanggil method yang sudah dibuat dengan cara seperti ini pada main method:

```

public static void main(String[] args) {
    Mobil mobil = new Mobil();
    mobil.menyalakanMesin();
}

```

Maka output program akan seperti ini:

```

PS C:\Users\radan> cd "d:\
Mesin mobil menyala
PS D:\kuliah\Aslab\asisten

```

Selain itu, jika kita mengubah isi dari atribut pada class Mobil seperti ini:

```

String merk = "Honda";
String model = "X100";

```

Maka kita juga bisa memanggil isi dari masing-masing atribut dengan cara seperti ini:

```

public static void main(String[] args) {
    Mobil mobil = new Mobil();
    mobil.menyalakanMesin();
    System.out.println(mobil.merk);
    System.out.println(mobil.model);
}

```

Output dari program ketika dijalankan:


```

PS C:\Users\radan> cd "d:\t
Mesin mobil menyala
Honda
X100

```

Apakah ketika kita membuat sebuah class, kita hanya bisa membuat 1 instance objek? Jawabannya tidak. Kita bisa membuat sebuah instance objek sebanyak yang kita butuhkan, contohnya disini kita akan coba untuk membuat 3 instance objek dari class Mobil, maka bisa kita tulis kodenya seperti ini pada main method:

```

public static void main(String[] args) {
    Mobil mobil1 = new Mobil();
    Mobil mobil2 = new Mobil();
    Mobil mobil3 = new Mobil();
}

```

Masing-masing instance objek terpisah satu sama lain dan berbeda, maksudnya adalah objek mobil1 berdiri sendiri begitupun dengan objek mobil2 dan mobil3. Untuk cara memanggil method dan atribut pada masing-masing objek ialah sama dengan cara sebelumnya.

Selain kita bisa mengakses pada nilai atribut yang sudah kita inisialisasikan, kita bisa memanipulasi nilai pada atribut yang sudah kita deklarasikan sebelumnya. Contoh jika kita memiliki sebuah atribut seperti ini di class Mobil:

```

String merk;
String model;
String spekMesin;
String transmisi;
int tahun;

```

Dan kita memiliki 1 objek mobil pada main method, maka kita bisa memanipulasi nilai pada masing-masing atribut seperti ini:

```

public static void main(String[] args) {
    Mobil mobilPertama = new Mobil();

    mobilPertama.merk = "F1 Red Bull";
    mobilPertama.model = "RB-8";
    mobilPertama.spekMesin = "2.400 cc V8, limited to 18.000 RPM";
    mobilPertama.transmisi = "Red Bull 7-speed, hydraulic power shift";
}

```

```

    mobilPertama.tahun = 2012;
}

```

Untuk cara akses pada masing-masing atribut bisa satu-satu seperti ini:

```

public static void main(String[] args) {
    Mobil mobilPertama = new Mobil();

    mobilPertama.merk = "F1 Red Bull";
    mobilPertama.model = "RB-8";
    mobilPertama.spekMesin = "2.400 cc V8, limited to 18.000 RPM";
    mobilPertama.transmisi = "Red Bull 7-speed, hydraulic power shift";
    mobilPertama.tahun = 2012;

    System.out.println(mobilPertama.merk);
    System.out.println(mobilPertama.model);
    System.out.println(mobilPertama.spekMesin);
    System.out.println(mobilPertama.transmisi);
    System.out.println(mobilPertama.tahun);
}

```

Atau bisa juga dengan cara seperti ini:

```

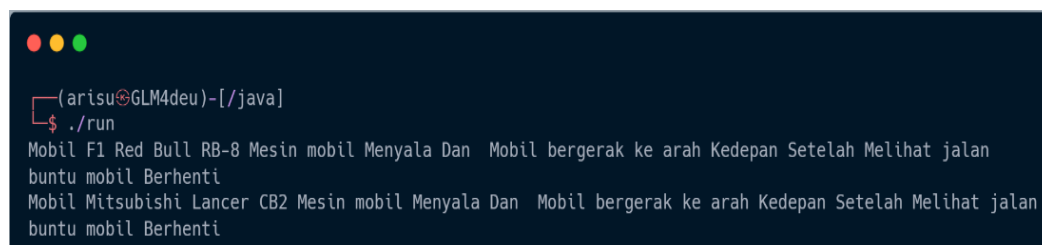
public static void main(String[] args) {
    Mobil mobilPertama = new Mobil();

    mobilPertama.merk = "F1 Red Bull";
    mobilPertama.model = "RB-8";
    mobilPertama.spekMesin = "2.400 cc V8, limited to 18.000 RPM";
    mobilPertama.transmisi = "Red Bull 7-speed, hydraulic power shift";
    mobilPertama.tahun = 2012;

    System.out.println("Mobil " + mobilPertama.merk + " " + mobilPertama.model);
    mobilPertama.menyalakanMesin();
    System.out.println("Dan " + mobilPertama.mengemudi("Kedepan") + " Setelah melihat
jalan buntu " + mobilPertama.mengerem());
}

```

Contoh lain untuk yang lebih kompleks:



```

(arisu@GLM4deu)-[/java]
$ ./run
Mobil F1 Red Bull RB-8 Mesin mobil Menyala Dan Mobil bergerak ke arah Kedepan Setelah Melihat jalan
buntu mobil Berhenti
Mobil Mitsubishi Lancer CB2 Mesin mobil Menyala Dan Mobil bergerak ke arah Kedepan Setelah Melihat jalan
buntu mobil Berhenti

```

- **Static dan non-static method**

Static method adalah method yang dapat dieksekusi atau dipanggil tanpa harus membuat sebuah instance objek. Sedangkan method non-static adalah method yang harus membuat instance objek untuk menggunakan atau memanggilnya, di mana method ini yang akan berkaitan erat dengan konsep OOP.

Cara mendeklarasikan sebuah method static di Java adalah dengan menambahkan keyword static. Contohnya seperti ini:

```
static returnType namaMethod(){
    // body dari method
}
```

Contoh implementasi dari static method kita akan coba ubah method mengerem dengan static menjadi seperti ini:



```
static String mengerem() {
    return "Berhenti";
}
```

Untuk cara pemanggilannya seperti ini pada main method:

```
public static void main(String[] args) {
    // tidak perlu membuat instance objek mobil
    System.out.println(mengerem());
}
```

Pada kode di atas kita bisa langsung memanggil atau menggunakan method mengerem() tanpa membuat sebuah instance objek dari class Mobil.

Untuk proses mendeklarasikan sebuah method static sebenarnya pada materi di atas sudah kita buat sebuah method non-static. Untuk pembuatan method static dideklarasikan tanpa menggunakan kata kunci *static* seperti ini:

```
returnType namaMethod(){
    // body method
}
```

Untuk implementasinya bisa cek lagi pada class Mobil yang sudah kita buat sebelumnya seperti ini:

```

public class Mobil{
    String merk;
    String model;

    void menyalakanMesin(){
        System.out.println("Mesin mobil menyala");
    }

    String mengemudi(String arah){
        return "Mobil bergerak ke arah" + arah;
    }

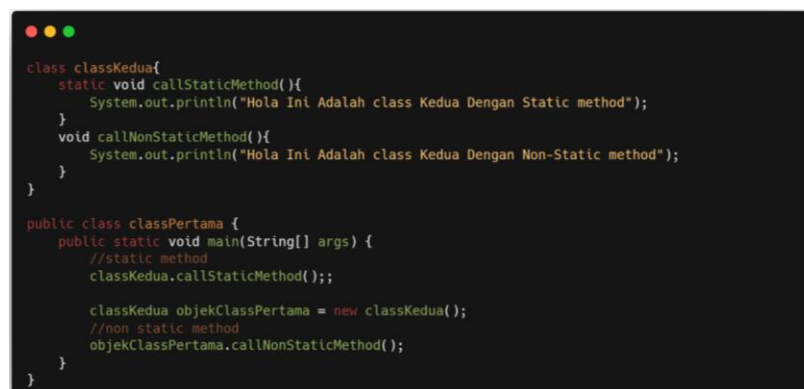
    String mengerem(){
        return "Berhenti";
    }

    int topSpeed(int topSpeed){
        return topSpeed;
    }
}

```

Pada kode class Mobil di atas, semua method yang ada bentuknya adalah non-static. Hal yang utama dari method non-static adalah pemanggilan menggunakan ‘.’ setelah nama objek.

Catatan untuk **static method**, bisa dipanggil langsung dengan nama method selama masih di lingkup class tersebut (satu file) dan jika dari class lain kita bisa memanggil dengan nama kelas lalu nama method. Contohnya:



```

class classKedua{
    static void callStaticMethod(){
        System.out.println("Hola Ini Adalah class Kedua Dengan Static method");
    }
    void callNonStaticMethod(){
        System.out.println("Hola Ini Adalah class Kedua Dengan Non-Static method");
    }
}

public class classPertama {
    public static void main(String[] args) {
        //static method
        classKedua.callStaticMethod();

        classKedua objekClassPertama = new classKedua();
        //non static method
        objekClassPertama.callNonStaticMethod();
    }
}

```

Output program:

```

(arisu@GLM4deu)-[/java]
$ ./run
Hola Ini Adalah class Kedua Dengan Static method
Hola Ini Adalah class Kedua Dengan Non-Static method

```

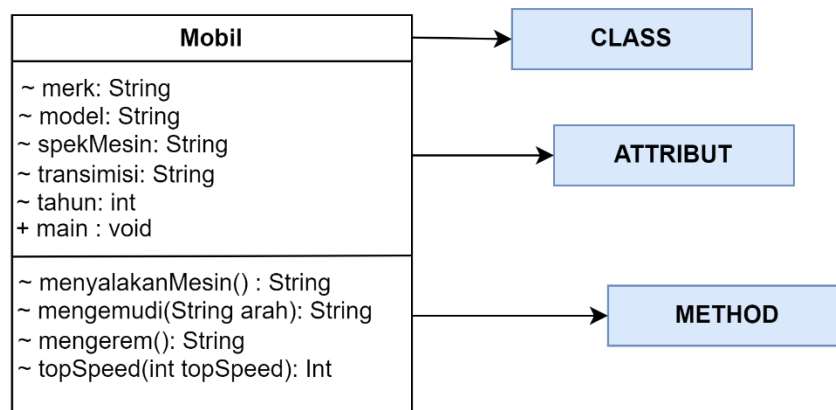
• Diagram class

Class dapat diibaratkan sebagai cetakan untuk membuat objek. Cetakan ini mendefinisikan karakteristik dan kemampuan yang dimiliki oleh semua objek yang dibuatnya. Class diagram adalah sebuah gambar yang menunjukkan struktur dan hubungan antar class. Ibarat denah rumah, diagram ini membantu kita memahami bagaimana berbagai bagian sistem saling terhubung.

Bagian-bagian penting class diagram:

1. Nama (dan stereotype) : Identitas dan jenis class
2. Atribut : karakteristik yang dimiliki oleh class
3. Method : kemampuan yang dimiliki oleh class

Contoh:

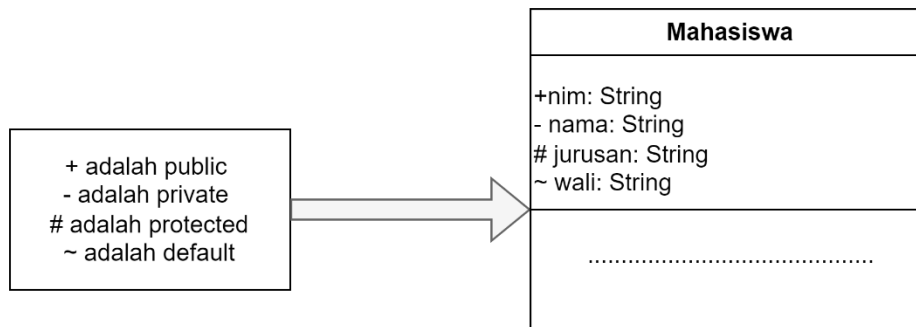


Penjelasan:

- Tipe data setelah nama method adalah tipe pengembalian (return type) dari function/method
- Access modifier

Fungsi dari akses modifier pada java adalah untuk membatasi scope dari sebuah class, constructor , atribut, method, atau anggota data lain yang terdapat dalam sebuah class Java. Selengkapnya tentang modifier akan dijelaskan di modul selanjutnya.

Contoh:



▪ Hubungan antar class

Asosiasi, yaitu hubungan statis antar class yang biasanya menggambarkan class yang memiliki atribut berupa class lain. Terdapat beberapa jenis asosiasi, seperti:

- 1) Asosiasi Sederhana: Bentuk asosiasi sederhana (———)
- 2) Agregasi yaitu hubungan yang menyatakan bagian, biasanya hubungan data master dan detailnya. Misal satu pembelian terdiri dari sejumlah item barang (◆——).
- 3) Navigability : menunjukan arah query/komunikasi antar objek, bisa satu atau dua arah, terlihat pada tanda panahnya (—————>)
- 4) Campuran / Composit : campuran asosiasi (◆——>)

Manfaat dari class diagram:

- 1) **Memahami struktur sistem:** Class diagram membantu kita melihat gambaran besar sistem dan bagaimana berbagai bagiannya saling terhubung.
- 2) **Meningkatkan komunikasi:** Class diagram membantu tim developer untuk berkomunikasi dengan lebih efektif tentang desain sistem.

- 3) **Mempermudah pengembangan:** Class diagram membantu developer untuk membuat kode yang lebih terstruktur dan mudah dipahami

CODELAB

Buatlah program input data mahasiswa yang dimana terdapat 2 class yaitu class Main dan class Mahasiswa. Untuk data mahasiswa disimpan di class Mahasiswa, dengan spesifikasi berikut:

- Method tampilUniversitas() merupakan static method
- Panjang nim harus tidak kurang dan tidak lebih dari 15 angka

Class diagram:

Mahasiswa
~ nama : String ~ nim: Int ~ jurusan: String
~ tampilUniversitas(): String ~tampilDataMahasiswa() : String

Main

~ menu(): void

Contoh output:

```
(arisu@GLM4deu)-[/java]
$ ./run
Menu:
1. Tambah Data Mahasiswa
2. Tampilkan Data Mahasiswa
3. Keluar
Pilihan Anda: 1
Masukkan nama mahasiswa: Sutrisno Adit Pratama
Masukkan NIM mahasiswa: 2022
Nim Harus 15 Digit!!!
Masukkan NIM mahasiswa: 202210370311203
Masukkan jurusan mahasiswa: Informatika
Data mahasiswa berhasil ditambahkan.

Menu:
1. Tambah Data Mahasiswa
2. Tampilkan Data Mahasiswa
3. Keluar
Pilihan Anda: 2
Data Mahasiswa:
Universitas Muhammadiyah Malang
Nama: Sutrisno Adit Pratama, NIM: 202210370311203, Jurusan: Informatika

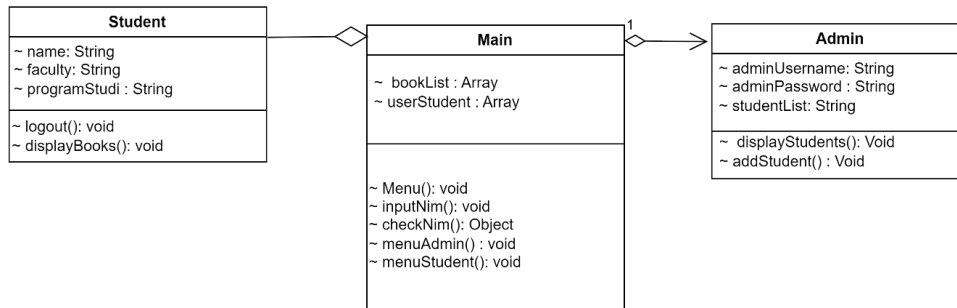
Menu:
1. Tambah Data Mahasiswa
2. Tampilkan Data Mahasiswa
3. Keluar
Pilihan Anda: 1
Masukkan nama mahasiswa: Jane Doe
Masukkan NIM mahasiswa: 200510370311521
Masukkan jurusan mahasiswa: Informatika
Data mahasiswa berhasil ditambahkan.

Menu:
1. Tambah Data Mahasiswa
2. Tampilkan Data Mahasiswa
3. Keluar
Pilihan Anda: 2
Data Mahasiswa:
Universitas Muhammadiyah Malang
Nama: Sutrisno Adit Pratama, NIM: 202210370311203, Jurusan: Informatika
Nama: Jane Doe, NIM: 200510370311521, Jurusan: Informatika

Menu:
1. Tambah Data Mahasiswa
2. Tampilkan Data Mahasiswa
3. Keluar
Pilihan Anda: 3
Adios
```


TUGAS

Melanjutkan tugas pada modul 1 sebelumnya yang hanya memanfaatkan class Main saja, pada tugas ini pecah method-method yang ada di class main menjadi class Student dan Admin seperti pada diagram berikut:



Penjelasan

Class Main:

- Atribut array **bookList** berisi data data buku seperti stok buku ,author,id_buku,dan juga judul
- Atribut array **userStudent** berisi data data user yang akan menjadi role user
- **Menu()** : sebagai tampilan awal pilihan login sebagai admin / student
- **inputNim()** : sebagai Inputan user jika memilih user student
- **checkNim()** : merupakan method untuk menjadi validator apakah user yang di inputkan oleh user ada atau tidaknya
- **menuAdmin()** : method untuk menampilkan dashboard admin
- **menuStudent()** : untuk menampilkan dashboard student

Class Student

- **displayBooks()** : menampilkan daftar buku yang ada
- **logout()** : untuk keluar dari program

Class Admin

- **addStudent()** : digunakan untuk menambahkan student, ketika proses penambahannya terdapat input nama, nim, fakultas, dan program studi. Terdapat pengecekan jika input nim memiliki panjang tidak sama dengan 15 maka nim tidak valid
- **displayStudent()** : digunakan untuk menampilkan daftar mahasiswa yang terdaftar

Contoh output program:

```
=====
|| No. || Id buku      || Nama Buku      || Author  || Category || Stock ||
=====
|| 1   || 388c-e681-9152    || title         || author  || Sejarah  || 4     ||
|| 2   || ed90-be30-5cdb    || title         || author  || Sejarah  || 0     ||
|| 3   || d95e-0c4a-9523    || title         || author  || Sejarah  || 2     ||
=====
```

Input Id buku yang ingin dipinjam (input 99 untuk back)

Input : 99

Kembali ke menu awal...

==== Student Menu ====

1. Buku terpinjam
2. Pinjam buku
3. Pinjam Buku atau Logout

Choose option (1-3): 3

System logout...

==== Library System ====

1. Login as Student
2. Login as Admin
3. Exit

Choose option (1-3): 3

Thank you. Exiting program.

==== Admin Menu ====

1. Add Student
2. Display Registered Students
3. Logout

Choose option (1-3): 2

List of Registered Students:

Name: Taufiq Ramadhan

Faculty: Teknik

NIM: 202210370311208

Program: Informatika

Name: Who

Faculty: Teknik

NIM: 200510370310521

Program: Informatika

Name: Sutrisno Adit Pratama

Faculty: Teknik

NIM: 202210370311203

Program: Informatika

==== Admin Menu ====

1. Add Student
2. Display Registered Students
3. Logout

Choose option (1-3): 3

Logging out from admin account.

==== Library System ====

1. Login as Student
2. Login as Admin
3. Exit

Choose option (1-3): 1

Enter your NIM (input 99 untuk back): 202210370311203

==== Student Menu ====

1. Buku terpinjam
2. Pinjam buku
3. Pinjam Buku atau Logout

Choose option (1-3): 3

```

(arisu@GLM4deu)-[/java]
$ ./run

===== Library System =====
1. Login as Student
2. Login as Admin
3. Exit
Choose option (1-3): 2
Enter your username (admin): test
Enter your password (admin): test
Invalid credentials for admin.

===== Library System =====
1. Login as Student
2. Login as Admin
3. Exit
Choose option (1-3): 2
Enter your username (admin): admin
Enter your password (admin): admin

===== Admin Menu =====
1. Add Student
2. Display Registered Students
3. Logout
Choose option (1-3): 1
Enter student name: Sutrisno Adit Pratama
Enter student NIM: 21313131
Nim Harus 15 Digit
Enter student NIM: 202210370311203
Enter student faculty: Teknik
Enter student program: Informatika
Student successfully registered.

```

RUBRIK PENILAIAN

Aspek Penilaian	Poin
Codelab	20
Tugas	30
Pemahaman	50
Total	100%