VERSION 3.0

13 Februari 2024


# PEMROGRAMAN BERORIENTASI OBJEK

## MODUL 3 – CONSTRUCTOR, ENCAPSULATION, INHERITANCE, OVERRIDING, SUPER KEYWORDS

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 membangun aplikasi sederhana dengan menerapkan Constructor
2. Mahasiswa mampu membangun aplikasi sederhana dengan menerapkan Encapsulation
3. Mahasiswa mampu membangun aplikasi sederhana dengan menerapkan Overriding
4. Mahasiswa mampu membangun aplikasi sederhana dengan menerapkan Super keywords

## TARGET MODUL

1. Mahasiswa mampu memahami konsep Constructor
2. Mahasiswa mampu memahami konsep Encapsulation
3. Mahasiswa mampu memahami konsep Overriding
4. Mahasiswa mampu memahami konsep Super keywords
5. Mahasiswa mampu memahami dan menerapkan Object Oriented Programming.

## PERSIAPAN

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

## KEYWORDS

- Pemrograman Berorientasi Objek
- Encapsulation
- OOP
- Access Modifier
- Overriding
- super keywords

## TEORI

- **Constructor**

Constructor merupakan suatu method yang akan memberikan nilai awal pada saat suatu objek dibuat. Pada saat program dijalankan, constructor akan langsung memberikan nilai awal pada saat membuat suatu *instance* objek. Pada saat kita bekerja dengan contructor, ada beberapa hal yang perlu diperhatikan, yaitu:

1. Nama constructor harus sama dengan nama class
2. Tidak ada return type yang diberikan ke dalam constructor signature.
3. Tidak ada return statement, di dalam tubuh constructor.

Contoh ketika kita membuat sebuah class Mobil dan menggunakan constructor:

```
public class Mobil {
    String nama;
    int speed;

    Mobil(){
        System.out.println("Method constructor dieksekusi");
    }
}
```

Pada contoh di atas constructor ditulis seperti ini:

```
    Mobil(){
        System.out.println("Method constructor dieksekusi");
    }
```

Mari kita coba untuk membuat objek baru dari class Mobil di Main class:

```
Mobil mobil = new Mobil();
```

Sehingga sekarang kode pada Main class kita menjadi seperti ini:

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

Ketika dijalankan akan menampilkan output berikut:

```
PS C:\Users\radan> cd "d:\kulia
method constructor dieksekusi
```

Method constructor akan dieksekusi ketika sebuah instance objek baru dibuat, bahkan tanpa memanggil method itu. Cara penulisan ketika membuat sebuah contructor class ada 2 cara, yaitu:

1. Menulis method dengan nama yang sama pada class

```
class Mobil {
    String nama;
    int speed;

    Mobil(){
        System.out.println("Method constructor dieksekusi");
    }
}
```

2. Menggunakan modifier public

```
class Mobil {
    String nama;
    int speed;

    public Mobil(){
        System.out.println("Method constructor dieksekusi");
    }
}
```

- **Constructor dengan Parameter**

Setelah mengetahui apa itu constructor, kita akan bertanya-tanya untuk apa fungsi dari constructor itu sendiri? Jawabannya ialah dengan sebuah contoh kasus, misalnya kita memiliki sebuah data mobil 5 buah dengan data setiap mobil memiliki nama dan speed yang berbeda-beda dan ingin membuat sebuah objek mobil satu-satu. Mungkin kode yang kita buat akan seperti ini:

- pada class Mobil

```
class Mobil {
    String nama;
    int speed;

    public Mobil(){
```

```
        System.out.println("Method constructor dieksekusi");
    }
}
```

- pada class Main

```
public class Main{
    public static void main(String[] args){
        Mobil mobil1 = new Mobil();
        Mobil mobil2 = new Mobil();
        Mobil mobil3 = new Mobil();
        Mobil mobil4 = new Mobil();
        Mobil mobil5 = new Mobil();

        mobil1.nama = "Toyota Avanza";
        mobil1.speed = 160;
        mobil2.nama = "Suzuki SX4 2007";
        mobil2.speed = 170;
        mobil3.nama = "Mobil sedan";
        mobil3.speed = 210;
        mobil4.nama = "BMW i8";
        mobil4.speed = 190;
        mobil5.nama = "Bugatti Chiron";
        mobil5.speed = 240;
    }
}
```

pada kode class Main terlihat bahwasannya kita harus melakukan input satu-satu dan memakan banyak tempat hanya untuk menginputkan sebuah data saja. Maka dari itu jika kita ingin menginisialisasi sebuah data secara langsung pada saat sebuah instance class dibuat, maka bisa kita manfaatkan sebuah parameter dengan cara menambahkan sebuah parameter pada constructor class Mobil dan menggunakan argumen ketika membuat instance objek class Mobil. Kodenya akan tampak seperti ini.

- pada class Mobil

```
class Mobil {
    String nama;
    int speed;

    public Mobil(String nama, int speed){
        this.nama = nama;
        this.speed = speed;
```

```
        System.out.println("Method constructor dieksekusi");
    }
}
```

- pada class Main

```
public class Main{
    public static void main(String[] args){
        Mobil mobil1 = new Mobil("Toyota Avanza", 160);
        Mobil mobil2 = new Mobil("Suzuki SX4 2007", 170);
        Mobil mobil3 = new Mobil("Mobil sedan", 210);
        Mobil mobil4 = new Mobil("BMW i8", 190);
        Mobil mobil5 = new Mobil("Bugatti Chiron", 240);
    }
}
```

Pada kode class Mobil di atas, kita menambahkan parameter nama dan speed ke dalam constructor. Jadi ketika membuat sebuah instance objek dari class Mobil, kita harus menambahkan argumen seperti ini:

```
Mobil mobil5 = new Mobil("Porsche 911", 250);
```

Sangat terlihat perbedaannya ketika kita tidak menggunakan constructor dan ketika menggunakan constructor, kode akan lebih simpel dan mudah dibaca ketika menggunakan constructor.

Jika kita tidak membuat constructor pada sebuah class, maka secara otomatis dan tidak terlihat sebenarnya class itu sudah membuat constructor sendirinya tetapi body constructor yang kosong.

- **Encapsulation**

  Encapsulation adalah pembungkus, encapsulation pada object oriented maksudnya adalah membungkus class dan menjaga apa apa saja yang ada di dalam class tersebut, baik method ataupun atribut, agar tidak dapat diakses oleh class lainnya. Untuk menjaga hal tersebut dalam Encapsulation dikenal nama Hak Akses Modifier yang terdiri dari Private. Public dan Protected. Tapi perlu diingat, modifier tidak hanya bisa diberikan kepada atribut dan method saja. Tapi juga bisa diberikan kepada interface, enum, dan class itu sendiri.

Fungsi dari access modifier pada Java adalah untuk membatasi scope dari sebuah class, constructor, variabel, method, atau anggota data lainnya yang terdapat dalam suatu program Java.

| Modifier | Class | Package | Subclass | World |
|---|---|---|---|---|
| public | True | True | True | True |
| protected | True | True | True | False |
| no modifier | True | True | False | False |
| private | True | False | False | False |

Perhatikan kode berikut:

```
package com.praktikum;

public class Mobil {
    private String nama;
    public int speed;
    protected boolean idDrive;

    public Mobil(){
        System.out.println("ini constructor");
    }
}
```

Kode yang berwarna merah dan bergaris di atas adalah modifier. Modifier akan digunakan untuk menentukan akses atribut, method, dan class.

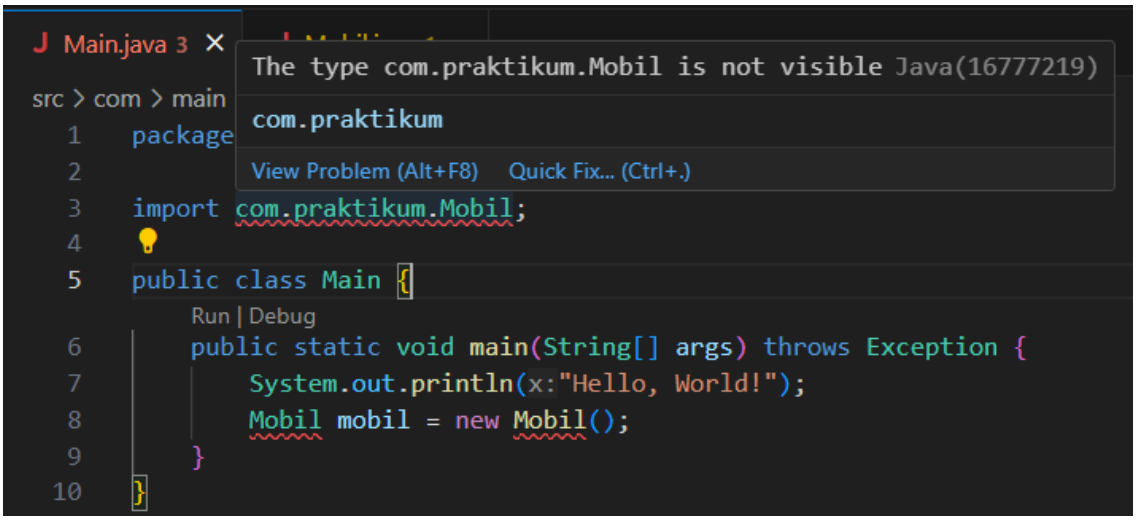
a. Public

Memberikan hak akses kepada atribut atau method agar bisa diakses oleh siapapun (property atau class lain diluar class yang bersangkutan), artinya method atau atribut yang ada di class A dapat diakses oleh siapapun baik itu class A, class B dan seterusnya. Contoh:

```
package com.praktikum;

class Mobil {
    private String name;

    public void changeName(String newName){
        this.name = newName;
```

```
    }
}
```

Pada class Mobil di atas terdapat atribut name dan method changeName(). Kedua hal tersebut kita berikan sebuah modifier public, artinya keduanya akan bisa diakses dari mana saja. Namun, class Mobil tidak kita berikan sebuah modifier. Maka yang akan terjadi adalah class tersebut tidak akan bisa diimport atau diakses dari luar package.




Class Mobil berada di package com.praktikum, lalu kita coba untuk akses dari package com.main, maka akan terjadi error seperti gambar di atas. Cara untuk memperbaikinya ialah dengan menambahkan modifier public pada class Mobil. Maka kode pada class Mobil akan menjadi seperti ini:

```
package com.praktikum;

public class Mobil {
    private String name;

    public void changeName(String newName){
        this.name = newName;
    }
}
```

Maka error akan hilang dan kode sudah bisa dijalankan secara normal.

b. Protected

Memberikan hak akses pada class itu sendiri dan class hasil turunannya (inheritance), artinya apa saja yang di class A hanya bisa diakses oleh class A sendiri

dan class yang meng extends class A. Namun harus dipahami class lain yang berada dalam satu package dengan class A mampu mengakses tipe data protected, sedangkan yang tidak mampu mengakses adalah class - class yang berada diluar package class A. Untuk dapat mengaksesnya, class yang berada diluar package class A harus meng extends class A. Terkhususkan modifier protected hanya boleh digunakan pada atribut dan method saja, tidak bisa diberikan kepada class, enum, dan interface.

Contoh class Mobil memiliki turunan yaitu class Honda:

```
package com.praktikum;

public class Mobil {
    protected String name;

    public void changeName(String newName){
        this.name = newName;
    }
}
```

Pada kode class Mobil di atas kita memberikan modifier protected pada atribut name.

```
package com.praktikum;

public class Honda extends Mobil{
    public void Driving(){
        name = "Civic";

        System.out.println("Driving menggunakan mobil Honda " + name);
    }
}
```

Maka ketika kita coba akses pada class Honda yang dimana merupakan class turunannya hal ini tidak akan terjadi error. Atau jika kita ingin mengakses dari class yang satu package dengan class Mobil, itu juga bisa meskipun bukan class turunan dari class Mobil.

Tetapi, apabila kita mencoba untuk mengakses atribut name dari package yang berbeda maka akan terjadi error karena atribut name sudah kita berikan modifier protected.

```
package com.main;

import com.praktikum.Mobil;
```

```
public class Main {
    public static void main(String[] args) throws Exception {
        Mobil mobil = new Mobil();

        // akan terjadi error di sini
        mobil.name = "Mobil civic";
    }
}
```

c. Private

Memberikan hak akses hanya pada class itu sendiri, artinya apa-apa saja yang ada di dalam class A baik itu method apapun atribut hanya bisa diakses oleh class A saja, class lain tidak bisa mengaksesnya. Modifier private juga tidak bisa diberikan kepada class, enum, dan interface. Modifier private hanya diberikan kepada attribut dan method class saja. Contoh:

```
package com.praktikum;

public class Mobil {
    private String name;

    public void setName(String newName){
        this.name = newName;
    }

    public String getName(){
        return name;
    }
}
```

Pada contoh di atas, kita memberikan modifier private pada atribut name dan modifier public pada method setName() dan getName(). Apabila kita mencoba untuk langsung mengakses pada atribut name seperti ini:

```
        Mobil mobil = new Mobil();
        mobil.name = "Civic"; // <= ini akan terjadi error
```

Lantas bagaimana cara untuk mengakses sebuah atribut atau method yang diberikan modifier private dari luar class? Jawabannya ialah bisa menggunakan method getter dan setter. Karena, method ini akan selalu diberikan medifier public. Method setter dan getter akan dipelajari setelah ini.

- **Method Setter dan Getter**

Setter adalah sebuah method saat kita memasukkan sebuah nilai/values ke dalam suatu variabel/atribut/object, sedangkan Getter adalah sebuah method saat kita mengambil sebuah nilai/values dari suatu variabel/atribut/object.

Cara untuk implementasi setter dan getter pada class Mobil, buat dahulu beberapa atribut dengan modifier private. Setelah itu buat sebuah method biasa, untuk setter harap diawali dengan kata set dan untuk getter harap diawali get agar supaya mudah diketahui kode mana yang termasuk setter dan getter. Untuk modifier method setter dan getter selalu berikan modifier public, karena method ini yang akan diakses oleh class lain. Contohnya:

```
package com.praktikum;

public class Mobil {
    private String name;
    private String warna;
    private int maxSpeed;
    private boolean isModified;

    public void setName(String name){
        this.name = name;
    }

    public String getName(){
        return name;
    }

    public void setWarna(String warna){
        this.warna = warna;
    }

    public String getWarna(){
        return warna;
    }

    public void setMaxSpeed(int maxSpeed){
        this.maxSpeed = maxSpeed;
    }

    public int getMaxSpeed(){
```

```
        return maxSpeed;
    }

    public void setIsModified(boolean isModified){
        this.isModified = isModified;
    }

    public boolean getIsModified(){
        return isModified;
    }
}
```

Perbedaan utama antara method setter dengan getter terletak pada nilai kembalian, parameter, dan isi method. Method setter tidak memiliki nilai kembalian void (kosong), karena tugasnya hanya untuk mengisi data ke dalam atribut. Sedangkan method getter memiliki nilai kembalian sesuai dengan tipe data yang akan diambil.

```
    // ini method setter
    public void setName(String name){
        this.name = name;
    }

    // ini method getter
    public String getName(){
        return name;
    }
```

Terkadang kita memiliki pertanyaan, bolehkan untuk awalan menggunakan bahasa indonesia? misalnya isi untuk setter dan ambil untuk getter. Hal itu boleh-boleh saja, tetapi sangat tidak dianjurkan. Karena suatu saat nanti jika kita bekerja dengan tim, tim yang lain akan kesulitan untuk membaca. Apalagi tim tersebut lintas negara yang menggunakan bahasa inggris.

Setelah kita membuat method setter dan getter, kita bisa mengakses atau menggunakannya seperti method biasa. Contoh:

```
package com.main;
import com.praktikum.Mobil;
```

```java
public class Main {
    public static void main(String[] args) throws Exception {

        Mobil mobil = new Mobil();
        // menggunakan method setter
        mobil.setName("Civic");
        mobil.setMaxSpeed(180);

        // menggunakan method getter
        System.out.println("Nama : " + mobil.getName());
        System.out.println("Max Speed : " + mobil.getMaxSpeed());
    }
}
```

Hasil output:

```
Nama : Civic
Max Speed : 180
PS C:\Users\radar
```

Beberapa alasan kenapa harus menggunakan Setter dan Getter:

1. Untuk meningkatakan keamanan data
2. Agar lebih mudah dalam mengontrol atribut dan method
3. Class bisa kita buat menjadi read-only dan write-only
4. Programmer dapat mengganti sebagian dari kode tanpa harus takut berdampak pada kode lain

- **Inheritance**

Pewarisan atau Inheritance adalah prinsip dalam pemrograman di mana sebuah class memiliki kemampuan untuk mewariskan properti dan metode yang dimilikinya kepada class lain. Konsep pewarisan digunakan untuk memanfaatkan fitur 'code reuse' guna menghindari duplikasi kode program.

Dengan adanya pewarisan, terbentuklah struktur atau hierarki class dalam kode program. Class yang memberikan warisan disebut sebagai class induk (parent class), super class, atau base class. Sementara class yang menerima warisan disebut sebagai class anak (child class), sub class, derived class, atau heir class.

Tidak semua properti dan metode dari class induk akan diwariskan. Properti dan metode dengan hak akses private tidak akan diwariskan kepada class anak. Hanya properti dan metode dengan hak akses protected dan public saja yang dapat diakses dari class anak.

Suatu class yang memiliki turunan disebut sebagai parent class atau base class. Sedangkan class turunan itu sendiri sering disebut sebagai subclass atau child class. Sebuah subclass dapat mewarisi semua yang dimiliki oleh parent class.

Karena sebuah subclass dapat mewarisi semua yang dimiliki oleh parent class-nya, maka member dari subclass terdiri dari apa yang dimilikinya sendiri dan juga apa yang diwarisi dari class parent-nya.

Secara keseluruhan, dapat dikatakan bahwa suatu subclass hanya memperluas (extend) parent class-nya. Alasan kenapa harus menggunakan inheritance, misalkan pada sebuah mobil, kita akan membuat class-class mobil yang lebih spesifik. Lalu kita membuat kode untuk masing-masing class seperti ini:

File: Honda.java

```
package com.praktikum;
public class Honda {
    public String name;

    public void bergerakMaju(){
        System.out.println("Mobil " + name + " bergerak maju");
    }

    public void berbelok(){
        System.out.println("Mobil berbelok");
    }
}
```

File: Toyota.java

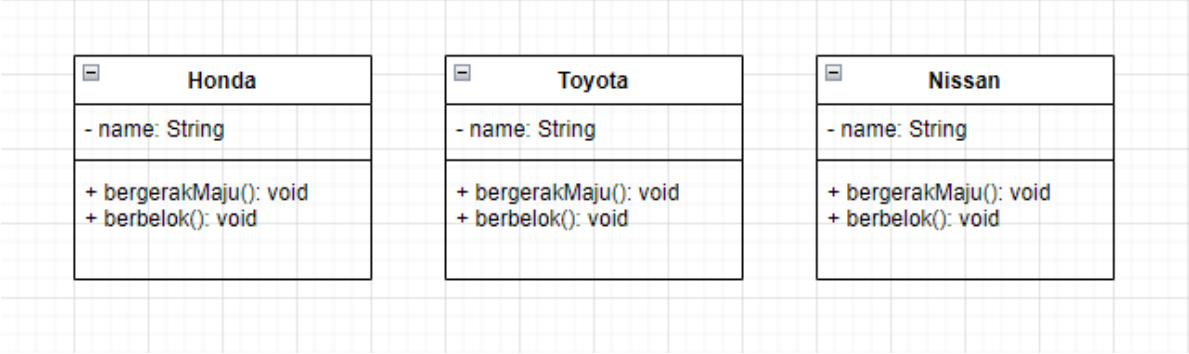
```
package com.praktikum;
public class Toyota {
    public String name;

    public void bergerakMaju(){
        System.out.println("Mobil " + name + " bergerak maju");
```

```
    }

    public void berbelok(){
        System.out.println("Mobil berbelok");
    }
}
```

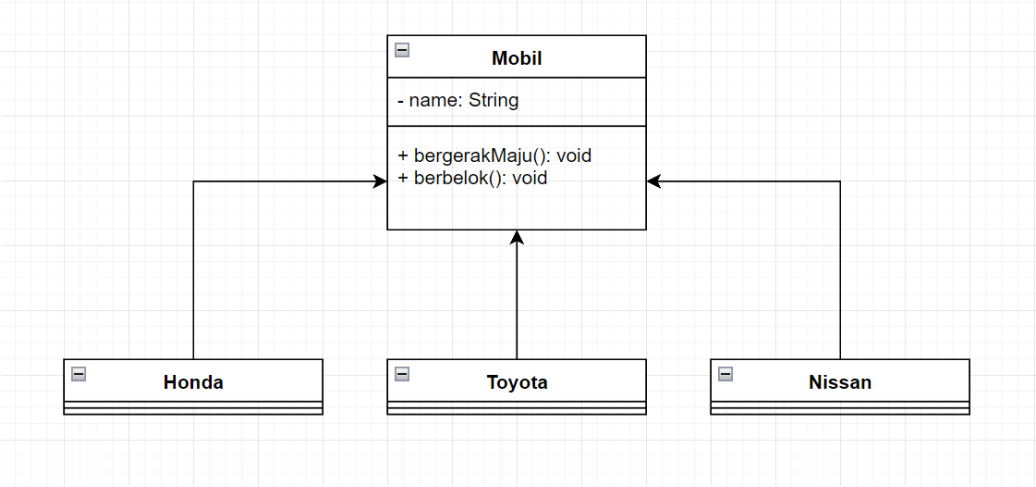
File: Nissan.java

```
package com.praktikum;
public class Nissan {
    public String name;

    public void bergerakMaju(){
        System.out.println("Mobil " + name + " bergerak maju");
    }

    public void berbelok(){
        System.out.println("Mobil berbelok");
    }
}
```

Dari ketiga class kode di atas, apakah boleh menulis kode seperti itu? Iya boleh-boleh saja. Tetapi hal itu tidak efektif, karena kita menulis kode yang sama berulang-ulang. Alih-alih kode berulang-ulang, solusi yang bisa digunakan ialah harus menggunakan inheritance. Mari kita lihat pada atribut dan method yang sama:




Pada diagram di atas terlihat bahwa atribut name, method bergerakMaju() dan berbelok() memiliki kesamaan pada class Honda, Toyota, dan Nissan. Setelah kita menggunakan inheritance dengan menyatukan atribut dan method yang memiliki cara kerja yang sama ke dalam satu class yaitu class Mobil, maka bentuk diagramnya akan menjadi seperti ini:




Class Mobil adalah class induk atau parent class yang memiliki class anak atau child class yaitu Honda, Toyota, dan Nissan. Apapun atribut yang ada di parent class akan dimiliki juga oleh class anak. Bentuk kode class Mobil seperti ini:

```
package com.praktikum;
public class Mobil {
    private String name;

    public void bergerakMaju(){
        System.out.println("Mobil " + name + " bergerak maju");
    }

    public void berbelok(){
        System.out.println("Mobil berbelok");
    }
}
```

Pada child class, kita harus menggunakan kata kunci extends untuk menyatakan kalau class itu adalah child class dari class Mobil.

File: Honda.java

```
package com.praktikum;
public class Honda extends Mobil{

}
```

File: Toyota.java

```
package com.praktikum;
public class Toyota extends Mobil{

```

```
}
```

File: Nissan.java

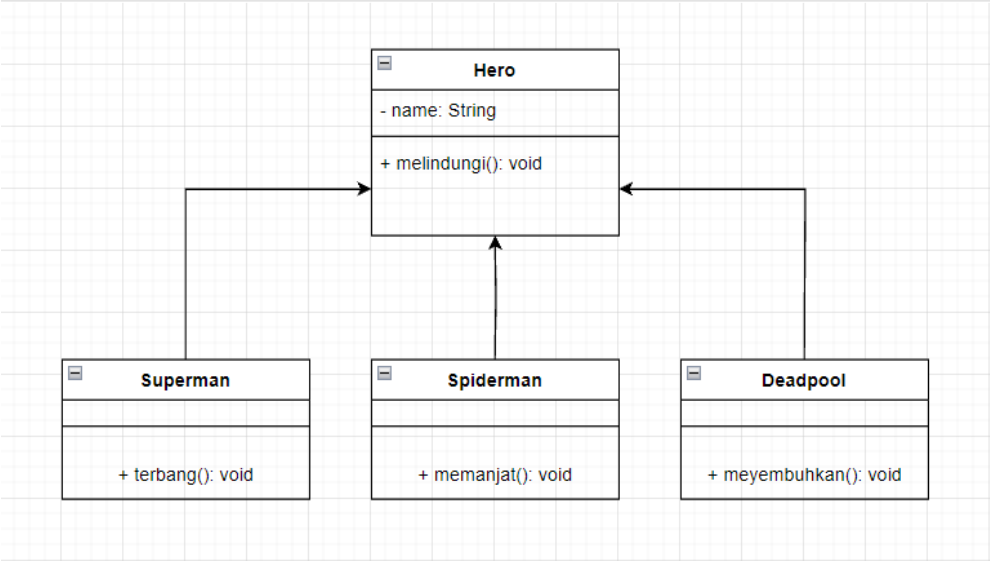
```
package com.praktikum;
public class Nissan extends Mobil{

}
```

Terlihat pada class Honda, Toyota, Nissan body class-nya kosong, pada child class kita bisa membiarkan isi body kosong atau diberi isi sesuai kebutuhan. Untuk cara membuat objek dari masing-masing class, kita bisa membuatnya seperti ini:

File: Main.java

```
        Mobil mobil = new Mobil();
        Honda honda = new Honda();
        Toyota toyota = new Toyota();
        Nissan nissan = new Nissan();
```

Contoh kasus lain ialah class Hero, misalnya class Hero yang memiliki child class berupa Superman, Spiderman, Deadpool. Ketiga child class memiliki kesamaan nama Hero, melindungi masyarakat. Tetapi masing-masing child class memiliki perbedaan yaitu untuk Superman bisa terbang, Spiderman bisa memanjat di tebing, Deadpool bisa penyembuhan diri. Diagram dari parent class dan child class bisa berbentuk seperti ini:




Untuk kode pada setiap class akan seperti ini:

File: Hero.java

```
public class Hero {
    private String name;
```

```
    public void melindungi(){
        System.out.println(name + " melindungi masyarakat");
    }

    public void setName(String name){
        this.name = name;
    }

    public String getName(){
        return name;
    }
}
```

File: Superman.java

```
public class Superman extends Hero{
    public void terbang(){
        System.out.println(getName() + " terbang...");
    }
}
```

File: Spiderman.java

```
public class Spiderman extends Hero{
    public void memanjat(){
        System.out.println(getName() + " memanjat tebing");
    }
}
```

File: Deadpool.java

```
public class Deadpool extends Hero{
    public void menyembuhkan(){
        System.out.println(getName() + " menyembuhkan diri");
    }
}
```

Untuk cara menggunakan pada file class Main.java seperti ini:

```
public class Main {
    public static void main(String[] args) {
        // pembuatan object
        Spiderman spiderman = new Spiderman();
        Superman superman = new Superman();
        Deadpool deadpool = new Deadpool();

        // memanggil method parent class melalui child class
        spiderman.setName("Tobey maguaire");
```

```
        superman.setName("Cristopher");
        deadpool.setName("Ryan Reynolds");

        // memanggil method yang ada di child class
        spiderman.memanjat();
        superman.terbang();
        deadpool.menyembuhkan();

        // Jika butuh object lain bisa membuat lagi
        Spiderman spiderman2 = new Spiderman();
    }
}
```

Pemanggilan method setName adalah milih parentclass yang dimiliki oleh class Hero, hal itu bisa dipanggil melalui child class karena masing-masing class Spiderman, Superman, Deadpool adalah turunan atau anak dari class Hero. Sedangkan method memanjat() hanya dimiliki oleh class Spiderman, maka class Superman dan Deadpool tidak bisa memanggil class memanjat() karena tidak memiliki class tersebut, berlaku juga untuk method terbang() di class Superman dan method menyembuhkan() di class Deadpool.

- **Overriding**

  Method overriding adalah sebuah kemampuan yang membolehkan untuk mereplikasi body method utama yang ada di parent class, hal yang membuat berbeda parent method dengan override method terletak pada isi method-nya.

  Secara sederhana method overriding dilakukan saat kita ingin membuat ulang sebuah method pada child class. Method overriding dapat dibuat dengan menambahkan anotasi @Override di atas nama method atau sebelum pembuatan method. Contoh:

  Parent class

```
public class parentClass {
    public void namaMethod(){
        System.out.println("Dari parent class");
    }
}
```

Child class

```
class childClass extends parentClass{
    @Override
    public void namaMethod(){
        System.out.println("Ini method override");
    }
}
```

Contoh kasus penggunaan inheritance ialah, pada class Hero di atas terdapat method melindungi() dan kita ingin mengubah isi method-nya pada class child Superman menjadi “melindungi masyarakat bumi dari serangan monster”. Method melindungi() di class Hero tidak perlu diubah apa-apa dan tetap seperti ini:

```
public class Hero {
    private String name;

    public void melindungi(){
        System.out.println(name + " melindungi masyarakat");
    }

    public void setName(String name){
        this.name = name;
    }

    public String getName(){
        return name;
    }
}
```

Tetapi pada class Superman kita ubah menjadi seperti ini:

```
public class Superman extends Hero{
    public void terbang(){
        System.out.println(getName() + " terbang...");
    }

    @Override
    public void melindungi(){
        System.out.println(getName() + " melindungi masyarakat bumi dari
serangan monster");
    }
}
```

Untuk mengetahui output dari perubahannya bisa kita panggil di Main class dengan cara yang sama ketika memanggil method biasa.

```
public class Main {
    public static void main(String[] args) {
        // pembuatan object
        Spiderman spiderman = new Spiderman();
        Superman superman = new Superman();
        Deadpool deadpool = new Deadpool();

        // memanggil method parent class melalui child class
        spiderman.setName("Tobey maguaire");
        superman.setName("Cristopher");
        deadpool.setName("Ryan Reynolds");

        // memanggil method yang ada di child class
        spiderman.melindungi();
        superman.melindungi();
        deadpool.melindungi();
    }
}
```

Terlihat pada akhir kode kita memanggil method yang sama pada 3 object berbeda, tetapi outputnya akan muncul seperti ini:

```
Tobey maguaire melindungi masyarakat
Cristopher melindungi masyarakat bumi dari serangan monster
Ryan Reynolds melindungi masyarakat
PS C:\Users\radan\Music\pbo modul>
```

Hal ini bisa terjadi karena class Superman sudah melakukan override pada method melindungi(), jadi isi dari method sudah tidak lagi sama dengan isi method pada parent class. Berbeda dengan class Spiderman dan Deadpool yang tidak melakukan override maka isinya akan sama dengan method melindungi() pada parent class.

- **Super Keywords**

Keyword super merupakan keyword yang digunakan untuk mengakses atribut ataupun method parentclass dari child class. Di java, keyword ini dapat digunakan untuk melakukan beberapa hal, seperti berikut:

- Mengakses method ataupun constructor parent class oleh child class
- Menunjuk anggota parent class oleh child class
- Mengakses method parent class oleh child class

Untuk contoh penggunaan keywords super, kita coba membuat class Hero memiliki constructor yang digunakan untuk menginisialisasi nama hero dan umur. Berikut adalah kode pada setiap class:

File: Hero.java

```
public class Hero {
    private String name;
    public int umur;

    public Hero(String name, int umur){
        this.name = name;
        this.umur = umur;
    }

    public void melindungi(){
        System.out.println(name + " melindungi masyarakat");
    }

    public void setName(String name){
        this.name = name;
    }

    public String getName(){
        return name;
    }
}
```

Parent class sudah kita ubah dengan bentuk constructor Hero(String name, int umur) maka pada child class akan menjadi seperti ini:

File: Spiderman.java

```
public class Spiderman extends Hero{
    public Spiderman(String name, int umur) {
        super(name, umur);
    }

    public void memanjat(){
```

```
        System.out.println(getName() + " memanjat tebing");
    }
}
```

File: Deadpool.java

```
public class Deadpool extends Hero{
    public Deadpool(String name, int umur) {
        super(name, umur);
    }

    public void menyembuhkan(){
        System.out.println(getName() + " menyembuhkan diri");
    }
}
```

File: Superman.java

```
public class Superman extends Hero{

    public Superman(String name, int umur) {
        super(name, umur);
    }

    public void terbang(){
        System.out.println(getName() + " terbang...");
    }

    @Override
    public void melindungi(){
        System.out.println(getName() + " melindungi masyarakat bumi dari
serangan monster");
    }
}
```

Terlihat pada child class Superman, Spiderman, dan Deadpool juga diwajibkan untuk memiliki constructor hal ini disebabkan parent class memiliki constructor. Tetapi di dalam constructor child class terpada super(name, umur); yang dipanggil, maksud dari super() adalah memanggil method pada parent class yang di mana method itu adalah constructor dari class Hero. Apakah super() hanya bisa digunakan pada method saja? Jawabannya tidak, bisa juga digunakan untuk pemanggilan variabel yang ada di parent class melalui child class. Contohnya pada class Deadpool.java kita ubah menjadi seperti ini:

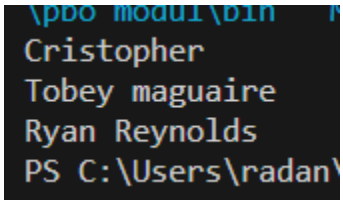
```java
public class Deadpool extends Hero{
    public Deadpool(String name, int umur) {
        super(name, umur);
    }

    public void menyembuhkan(){
        System.out.println(super.getName() + " menyembuhkan diri");
    }

    public void infoUmur(){
        System.out.println(getName() + " berumur: " + super.umur);
    }
}
```

Pada class Deadpool di atas kita menggunakan keywords super untuk memanggil atribut umur dan juga method getName(), yang di mana atribut umur dan method getName() tidak ada di class Deadpool tapi ada di class Hero yang merupakan parent class. Perlu diperhatikan untuk penggunaan keywords super pada constructor, pemanggilan keywords super() harus berada di awal setelah pembuatan constructor child class agar tidak terjadi error.

Untuk cara pemanggilan pada Main class, adalah seperti berikut:

```java
public class Main {
    public static void main(String[] args) {
        // pembuatan object disertai dengan argumen yang sesuai
        Spiderman spiderman = new Spiderman("Tobey maguaire", 34);
        Superman superman = new Superman("Cristopher", 43);
        Deadpool deadpool = new Deadpool("Ryan Reynolds", 36);

        // coba panggil
        System.out.println(superman.getName());
        System.out.println(spiderman.getName());
        System.out.println(deadpool.getName());
    }
}
```

Adapun outputnya akan seperti ini:




- **this Keywords**

Sejauh materi ini kita sudah banyak belajar materi tentang constructor, encapsulation, inheritance, overriding, super. Mungkin di pertengahan atau di awal materi muncul sebuah pertanyaan apa yang dimaksud dengan keywords this? Jawabannya keywords this biasanya digunakan untuk mengisi variabel class atau mengakses variabel yang ada di class tersebut. Contoh:

```
public class Hero {
    private String name;
    public int umur;

    public Hero(String name, int umur){
        this.name = name;
        this.umur = umur;
    }

    public void melindungi(){
        System.out.println(name + " melindungi masyarakat");
    }

    public void setName(String name){
        this.name = name;
    }

    public String getName(){
        return name;
    }
}
```

Pada constructor Hero dan method setName() terdapat kode yang menggunakan keywords this. Dilihat baik-baik pada class Hero memiliki atribut String name dan int umur, pada constructor Hero(String name, int umur) terdapat atribut juga tapi sebagai parameter.

Kedua hal ini berbeda untuk private String name dimiliki oleh class sedangkan String name yang ada di parameter constructor itu adalah parameter yang dimiliki oleh constructor. Pada constructor this.name = name memiliki sebuah arti bahwa atribut name yang awalnya kosong diisi oleh parameter name yang diisi oleh user.

Hal ini juga berlaku pada method setName(), karena method ini adalah setter atau method yang digunakan untuk inisialisasi sebuah nilai private, maka this.name = name juga memiliki arti atribut name pada class sekarang diisi oleh parameter name yang dimana pengisiannya dilakukan ketika pemanggilan method setter.

## CODELAB

### Menghitung Volume

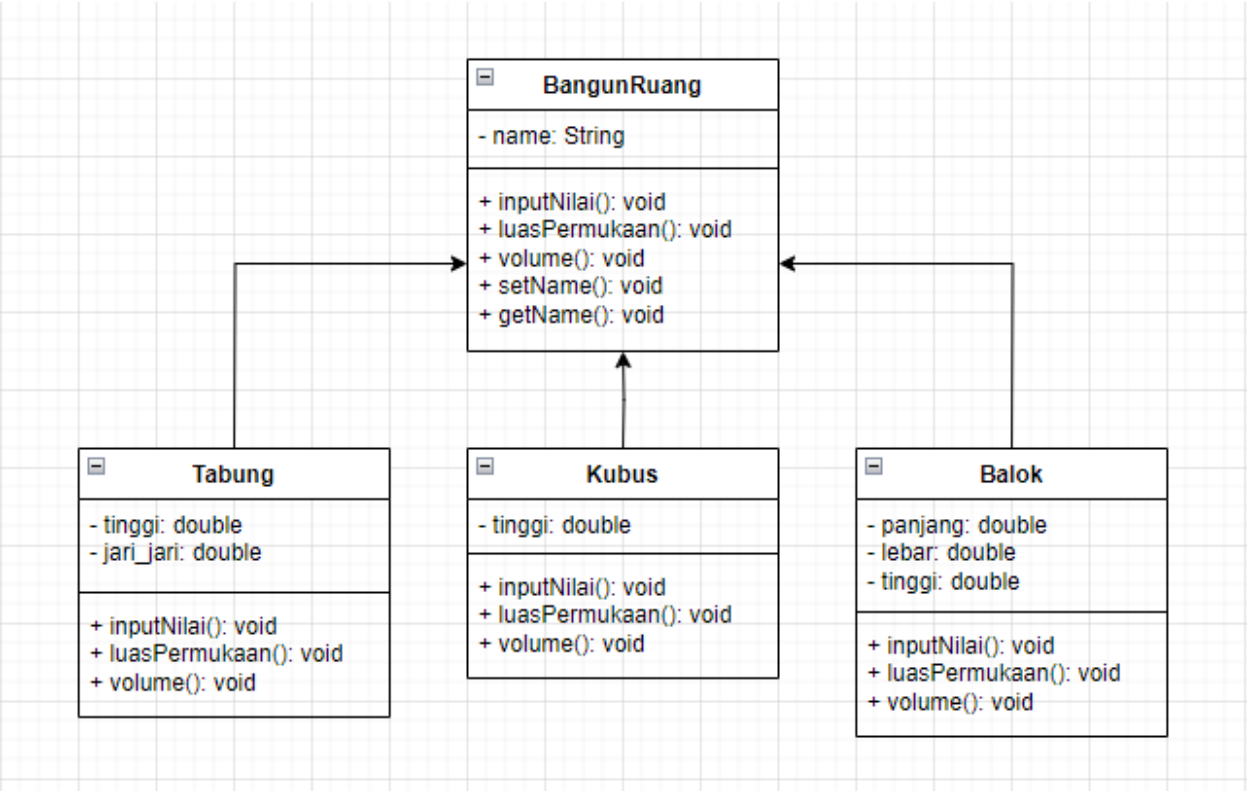
Silahkan pull codelab berikut: https://github.com/radan775/codelab_pbo_24.git, perbaiki kode program ini agar bisa digunakan untuk menghitung volume dan luas permukaan bangun ruang tabung, kubus, dan balok. Masing-masing bangun ruang memiliki parent class yang sama yaitu class BangunRaung dan class ini memiliki sebuah constructor untuk mengisi nama, buatlah juga setter dan getter terhadap atribut nama.

Class BangunRuang memiliki 3 method selain setter dan getter yaitu method inputNilai(), luasPermukaan(), volume() yang masing-masing method nantinya akan dilakukan overriding terhadap setiap child class bangun ruang dengan rumus perhitungan masing-masing. Untuk masing-masing child class akan memiliki sebuah constructor yang di dalamnya melakukan pemanggilan super() terhadap constructor parent class.

Berikut penjelasan mengenai 3 method di atas:

- **inputNilai()** digunakan untuk menginputkan atribut pada masing-masing class
- **luasPermukaan()** digunakan untuk menghitung luas permukaan
- **volume()** digunakan untuk menghitung volume

Pada main class buat semua object untuk masing-masing class, lalu panggil method inputNilai(), luasPermukaan(), volume(). Berikut adalah diagram untuk program ini:




## TUGAS

Melanjutkan tugas pada modul 2 sebelumnya, pada tugas ini tambahkan class User yang akan menjadi parent class dari class Admin dan Student. Lalu tambahkan juga class Book yang akan menjadi parent class dari class HistoryBook, StoryBook, dan TextBook.

Pada class User terdapat method displayBooks() yang digunakan untuk menampilkan daftar buku yang ada di perpustakaan. Pada class Student terdapat atribut name, nim, fakultas, prodi, array borrowedBooks. Dan untuk methodnya sebagai berikut:

- **Student(name, nim, faculty, programStudi)** sebuah method constructor yang digunakan untuk melakukan awal inisialisasi terhadap name, faculty, nim dan programStudi dengan memanfaatkan this keywords
- **displayInfo()** digunakan untuk menampilkan data diri mahasiswa yang login

- **showBorrowedBooks()** digunakan untuk menampilkan daftar buku yang sudah dipilih untuk dipinjam
- **logout()** digunakan untuk logout dari opsi student, jika tidak ada buku yang dipilih maka sistem langsung kembali ke menu utama, sedangkan jika ada maka user bisa memilih opsi batal meminjam atau proses meminjam setelah itu kembali ke menu utama.
- **displayBooks()** method ini akan dilakukan override dengan memanfaatkan super keywords dari parent class, karena student tidak hanya ditampilkan buku tapi juga bisa meminjamnya. Ketika student sudah memilih buku maka stok buku yang terdapat pada daftar buku akan berkurang 1. Untuk meminjam student akan diminta untuk meminjam berapa hari?
- **returnBooks()** digunakan untuk mengembalikan buku, jika buku dikembalikan maka stok buku akan bertambah.

Pada class Admin terdapat atribut studentlist dan beberapa method seperti ini:

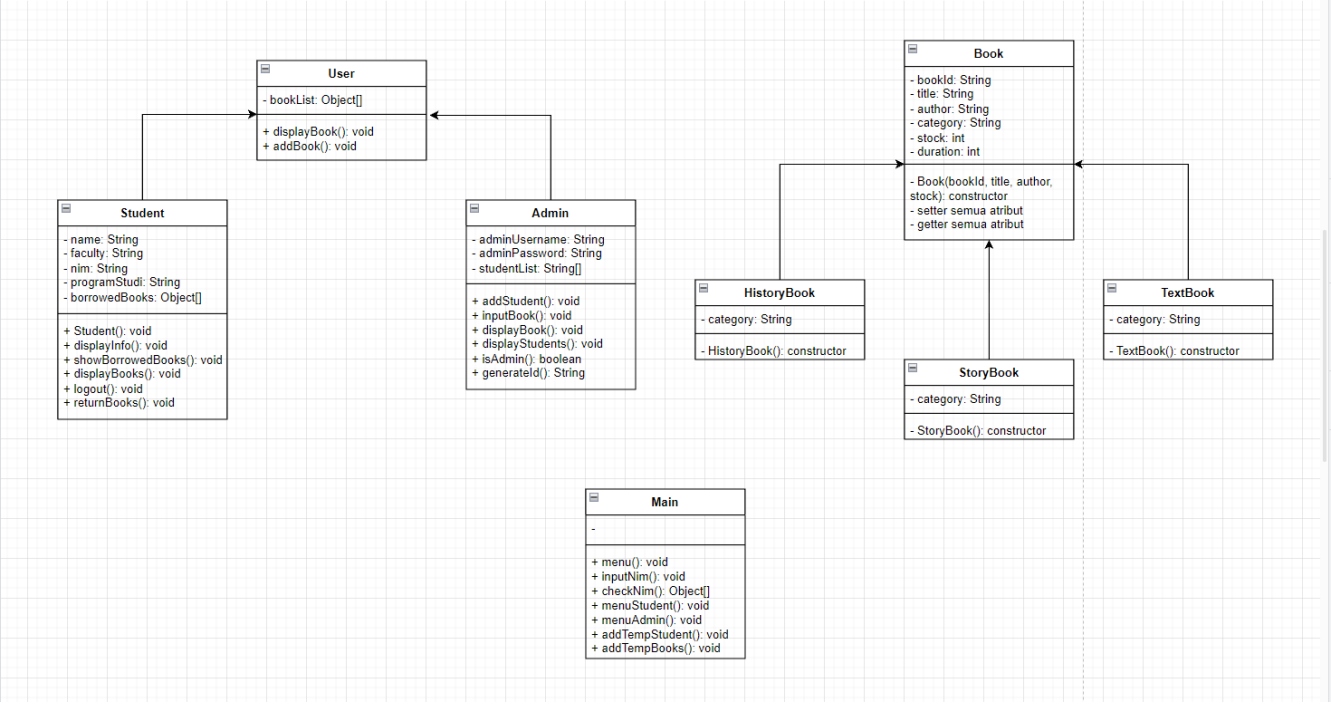
- **addStudent()** digunakan untuk menambahkan student, ketika proses penambahannya terdapat input nama, nim, fakultas, dan program studi. Terdapat pengecekan jika input nim memiliki panjang tidak sama dengan 15 maka nim tidak valid.
- **inputBook()** digunakan oleh admin untuk menambahkan buku. Ketika input akan terdapat opsi buku yang akan diinput yaitu Story Book, History Book, dan Text Book. Setelah memilih opsi untuk admin akan diminta untuk input judul, author, dan stok. Ketika proses penyimpanan akan tergenerate unik id secara otomatis yang nantinya akan tersimpan bersama dengan buku.
- **displayBooks()** method ini akan dilakukan override dengan memanfaatkan super keywords dari parent class, karena admin tidak bisa meminjam buku dan hanya bisa menampilkan daftar buku yang ada di perpustakaan.
- **displayStudent()** digunakan untuk menampilkan daftar mahasiswa yang terdaftar

- **isAdmin()** adalah method yang digunakan untuk memverifikasi apakah username dan password yang digunakan login admin termasuk kepada kredensial admin.
- **generateId()** digunakan untuk men-generate sebuah unikId, lihat di contoh output program.

Pada class Book terdapat atribut bookId, title, author, category, stock, dan duration. Buat sebuah constructor pada class Book disertai dengan parameter bookId, title, author, category. Setelah itu buat setter dan getter untuk setiap atribut.

Pada class HistoryBook buat extends dari Book, lalu buat sebuah constructor dengan memanfaatkan keywords super ke class Book. Untuk class StoryBook dan TextBook isinya sama dengan class HistoryBook.

Berikut untuk diagram pada program:




**Contoh Output:**

```
===== Library System =====
1. Login as Student
2. Login as Admin
3. Exit
Choose option (1-3): 1
```

```
Enter your NIM (input 99 untuk back): 202010370321210
==== Student Menu ====
1. Buku terpinjam
2. Pinjam buku
3. Kembalikan buku
4. Pinjam Buku atau Logout
Choose option (1-3): 1
Belum ada buku yang dipilih
Silahkan pilih buku terlebih dahulu

==== Student Menu ====
1. Buku terpinjam
2. Pinjam buku
3. Kembalikan buku
4. Pinjam Buku atau Logout
Choose option (1-3): 2
=================================================================================
|| No.|| Id buku       || Nama Buku  || Author          || Category   || Stock ||
=================================================================================
|| 1  || 388c-e681-9152 || title      || author          || Sejarah    || 4     ||
|| 2  || ed90-be30-5cdb || title      || author          || Cerita     || 0     ||
|| 3  || d95e-0c4a-9523 || title      || author          || Novel      || 2     ||
=================================================================================
Input Id buku yang ingin dipinjam (input 99 untuk back)
Input : ed90-be30-5cdb
Stock buku kosong!
Silahkan pilih yang lain.
=================================================================================
|| No.|| Id buku       || Nama Buku  || Author          || Category   || Stock ||
=================================================================================
|| 1  || 388c-e681-9152 || title      || author          || Sejarah    || 4     ||
|| 2  || ed90-be30-5cdb || title      || author          || Cerita     || 0     ||
|| 3  || d95e-0c4a-9523 || title      || author          || Novel      || 2     ||
=================================================================================
Input Id buku yang ingin dipinjam (input 99 untuk back)
Input : d95e-0c4a-9523
Berapa lama buku akan dipinjam? (maksimal 14 hari)
Input lama (hari): 5
=================================================================================
|| No.|| Id buku       || Nama Buku  || Author          || Category   || Stock ||
=================================================================================
|| 1  || 388c-e681-9152 || title      || author          || Sejarah    || 4     ||
|| 2  || ed90-be30-5cdb || title      || author          || Cerita     || 0     ||
```

```
|| 3  || d95e-0c4a-9523 || title      || author         || Novel      || 1     ||
================================================================================
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. Kembalikan buku
4. Pinjam Buku atau Logout
Choose option (1-3): 1
================================================================================
|| No.|| Id buku        || Nama Buku    || Author       || Category   || Durasi||
================================================================================
|| 1  || d95e-0c4a-9523 || title        || author       || Novel      || 5     ||
================================================================================
==== Student Menu ====
1. Buku terpinjam
2. Pinjam buku
3. Kembalikan buku
4. Pinjam Buku atau Logout
Choose option (1-3): 4
================================================================================
|| No.|| Id buku        || Nama Buku    || Author      || Category   || Durasi ||
================================================================================
|| 1  || d95e-0c4a-9523 || title        || author      || Novel      || 5      ||
================================================================================
Apakah kamu yakin untuk meminjam semua buku tersebut?
Input Y (iya) atau T (tidak): Y
Peminjaman buku berhasil dilakukan
Terima kasih...
===== 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. Add Book
3. Display Registered Students
4. Display Available Books
```

```
5. Logout
Choose option (1-5): 1
Enter student name: susanto
Enter student NIM: 201910370311000
Enter student faculty: Fikes
Enter student program: kesehatan
Student successfully registered.
===== Admin Menu =====
1. Add Student
2. Add Book
3. Display Registered Students
4. Display Available Books
5. Logout
Choose option (1-5): 2
Select book category:
1. Story Book
2. History Book
3. Text Book
Choose category (1-3): 2
Enter book title: sebuah cerita
Enter author: manusia
Enter the stock: 6
Book successfully added to the library.
===== Admin Menu =====
1. Add Student
2. Add Book
3. Display Registered Students
4. Display Available Books
5. Logout
Choose option (1-5): 4
================================================================================
|| No.|| Id buku        || Nama Buku        || Author     || Category  || Stock||
================================================================================
|| 1  || 388c-e681-9152 || title            || author     || Sejarah   || 4    ||
|| 2  || ed90-be30-5cdb || title            || author     || Cerita    || 0    ||
|| 3  || d95e-0c4a-9523 || title            || author     || Novel     || 1    ||
|| 4  || 3f10-2fb4-b478 || sebuah cerita    || manusia    || Sejarah   || 6    ||
================================================================================
===== Admin Menu =====
1. Add Student
2. Add Book
3. Display Registered Students
4. Display Available Books
```

```
5. Logout
Choose option (1-5): 3
List of Registered Students:
Name: radan
Faculty: Teknik
NIM: 202210370311208
Program: Informatika

Name: katak salto
Faculty: FEB
NIM: 201910330211809
Program: Bisnis

Name: ini nama
Faculty: FIKES
NIM: 202010370321210
Program: kedokteran mesin

Name: susanto
Faculty: Fikes
NIM: 201910370311000
Program: kesehatan

===== Admin Menu =====
1. Add Student
2. Add Book
3. Display Registered Students
4. Display Available Books
5. Logout
Choose option (1-5): 5
Logging out from admin account.
===== Library System =====
1. Login as Student
2. Login as Admin
3. Exit
Choose option (1-3): 3
Thank you. Exiting program.
```

Pada main class silahkan buat object pada masing-masing child class dan buat program dengan method yang terdapat pada diagram, jika butuh untuk membuat method lain diperbolehkan. Diagram yang diberikan hanyalah minimal spesifikasi bagaimana bentuk

program yang akan dibuat. Improvisasi terhadap program tidak masalah dengan sebuah syarat tidak jauh dari bentuk program.

## RUBRIK PENILAIAN

| Aspek Penilaian | Poin |
|---|---|
| Codelab | 20 |
| Tugas Modul | 35 |
| Pemahaman | 45 |
| **Total** | **100** |