

# *Encapsulation & Inheritance*

Teosofi Hidayah Agung  
Hafidz Mulia

Departemen Matematika  
Institut Teknologi Sepuluh Nopember

12 Mei 2025

**JavaScript injection** adalah teknik di mana seseorang menyisipkan (*inject*) kode JavaScript ke dalam halaman web, baik dengan tujuan baik (seperti debugging atau automasi) maupun jahat (seperti eksploitasi keamanan).

Web-web pemerintah atau bahkan web ITS sendiri sangat rawan untuk terkena hal ini, terlebih beberapa mahasiswa sudah tau bagaimana melakukan *JavaScript injection* sendiri. Mau mencobanya?

## Masalah

Semua yang sudah dicontohkan sebelumnya adalah akibat dari tidak adanya tata kelola atau kontrol terhadap sebuah akses dalam aplikasi, sehingga semua orang bisa mengakses semua data dan fungsi yang ada di dalamnya. Oleh karena itu, perlu adanya pembatasan akses terhadap data dan fungsi yang ada di dalam suatu kelas atau objek.

# Daftar isi

## 1 Modifier

- Access Modifier
- Non-Access Modifier

## 2 Encapsulation

- Setter
- Getter

## 3 Diagram Kelas

- Komponen

## 4 Latihan

## Definisi

**Modifier** adalah kata kunci (keyword) yang digunakan untuk mengubah perilaku dari suatu class, method, atau atribut. Modifier dapat digunakan untuk menentukan visibilitas (akses) dari suatu class, method, atau atribut.

Modifier dibagi menjadi dua, yaitu:

- **Access Modifier**: Menentukan visibilitas dari class, method, atau atribut.
- **Non-Access Modifier**: Menentukan perilaku dari class, method, atau atribut.

# Modifier

## Access Modifier

### Access Modifier

Mengatur siapa yang bisa mengakses class, method, atau atribut.

<b>Access Modifier</b>	<b>Class sama</b>	<b>Package sama</b>	<b>Subclass sama</b>	<b>Package beda</b>
public	Ya	Ya	Ya	Ya
protected	Ya	Ya	Ya	Tidak
<i>default</i>	Ya	Ya	Tidak	Tidak
private	Ya	Tidak	Tidak	Tidak

**Tabel:** Perbandingan Access Modifier dalam Java

# Modifier

## Access Modifier

```
1 public class Modifier { // Class
2
3     // Atribut
4
5     public int a;
6     protected char b;
7     String c;
8     private double d;
9
10    // Method
11
12    public void methodPublic() {...}
13    protected int methodProtected(...) {...}
14    String[] methodDefault(...) {...}
15    private void methodPrivate(...) {...}
16 }
```

# Modifier

## Non-Access Modifier

### Non-Access Modifier

Mengatur perilaku dari class, method, atau atribut.

Disini akan dijelaskan 3 modifier yang sering digunakan, yaitu:

- **static**: Sebuah atribut atau method bisa diakses tanpa harus membuat objek dari class tersebut.
- **final**: Digunakan untuk mendeklarasikan atribut atau method yang tidak dapat diubah nilainya setelah dideklarasikan.
- **abstract**: Digunakan untuk mendeklarasikan class atau method yang tidak memiliki implementasi, sehingga harus diimplementasikan oleh subclass-nya.



# Modifier

## Non-Access Modifier

Kode: Penggunaan static untuk method

```
1  class Utilitas {
2      static int jumlah(int a, int b) { // Method static
3          return a + b;
4      }
5  }
6  class Main {
7      public static void main(String[] args) {
8          Utilitas util = new Utilitas();
9          int hasil1 = util.jumlah(3, 4); //Pemanggilan dengan objek
10
11         int hasil2 = Utilitas.jumlah(3, 4); //Pemanggilan tanpa ada objek
12     }
13 }
```

# Modifier

## Non-Access Modifier

Kode: Penggunaan static untuk atribut

```
1  class Contoh {
2      int x = 0; // non-static
3      static int y = 0; // static
4  }
5  public class Main {
6      public static void main(String[] args) {
7          Contoh a = new Contoh(); Contoh b = new Contoh();
8          a.x = 5; a.y = 10;
9
10         System.out.println(b.x); // 0 karena x milik objek b sendiri
11         System.out.println(b.y); // 10 karena y milik class dan sudah
            diubah lewat a
12     }
13 }
```

# Modifier

## Non-Access Modifier

### Kode: Penggunaan final

```
1 final class Konstanta {  
2     final double PI = 3.14;  
3  
4     final void methodFinal() {  
5         System.out.println("Ini adalah method final");  
6     }  
7 }
```

### Catatan

- Class Konstanta tidak bisa diwarisi (*Inheritance*) oleh class lain.
- Method methodFinal() tidak bisa di-override oleh subclass.
- Atribut PI tidak bisa diubah nilainya setelah dideklarasikan.

# Modifier

## Non-Access Modifier

*“Untuk **abstract class** dan **abstract method** akan dibahas lebih lanjut pada pertemuan selanjutnya yang membahas tentang **Interface**”*

# Daftar isi

## 1 Modifier

- Access Modifier
- Non-Access Modifier

## 2 Encapsulation

- Setter
- Getter

## 3 Diagram Kelas

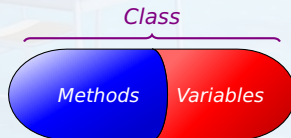
- Komponen

## 4 Latihan

# Encapsulation

## Definisi

**Encapsulation** adalah prinsip PBO yang menyatakan bahwa data (atribut) dan perilaku (method) dari suatu objek harus dibungkus (dikapsulkan) dalam satu unit, dan akses terhadap data langsung harus dibatasi agar hanya bisa diubah melalui method yang ditentukan



Gambar: Ilustrasi Encapsulation

# Encapsulation

## Manfaat

- **Data Hiding:** Data internal dari suatu objek disembunyikan dari dunia luar, sehingga mencegah akses langsung ke dalam atribut tersebut.
- **Data Integrity:** Hanya nilai-nilai yang telah divalidasi atau aman yang dapat diberikan ke atribut objek melalui method *setter*.
- **Reusability:** Kode yang dienkapsulasi menjadi lebih fleksibel dan dapat digunakan kembali untuk kebutuhan atau modifikasi di masa depan.
- **Security:** Data yang bersifat sensitif akan terlindungi karena tidak dapat diakses secara langsung dari luar class.

# Encapsulation

## Setter

### Setter

*Setter* adalah method yang digunakan untuk mengubah atau memberi nilai (menulis) pada atribut private di dalam suatu class.

- Nama method biasanya diawali dengan set diikuti nama atribut.
- Memiliki parameter untuk nilai baru.
- Tidak mengembalikan nilai `void`.



# Encapsulation

## Setter

Kode: Penggunaan setter

```
1 public class Mahasiswa {  
2     private String nama;  
3  
4     // Setter untuk nama  
5     public void setName(String namaBaru) {  
6         nama = namaBaru;  
7     }  
8 }
```

# Encapsulation

## Setter

Kode: Validasi dalam setter

```
1 public class Hero {  
2     private int HP;  
3  
4     // Setter dengan validasi  
5     public void setHP(int HPBaru) {  
6         if (HPBaru < 0) {  
7             this.HP = 0; // Jika negatif, atur ke 0  
8         } else {  
9             this.HP = HPBaru;  
10        }  
11    }  
12 }
```

# Encapsulation

## Getter

### Getter

*Getter* adalah method yang digunakan untuk mengambil nilai (membaca) dari atribut private di dalam suatu class.

- Nama method biasanya diawali dengan get diikuti nama atribut.
- Tidak memiliki parameter.
- Mengembalikan nilai atribut.

# Encapsulation

## Getter

Kode: Penggunaan *getter*

```
1 public class Mahasiswa {  
2     private String nama;  
3  
4     // Getter untuk nama  
5     public String getNama() {  
6         return nama;  
7     }  
8 }
```

# Encapsulation

## Getter

Kode: Perhitungan dalam *getter*

```
1 public class Lingkaran {  
2     private double jariJari;  
3  
4     // Getter: menghitung luas  
5     public double getLuas() {  
6         return Math.PI * jariJari * jariJari;  
7     }  
8 }
```

# Daftar isi

## 1 Modifier

- Access Modifier
- Non-Access Modifier

## 2 Encapsulation

- Setter
- Getter

## 3 Diagram Kelas

- Komponen

## 4 Latihan

## Definisi

**UML (Unified Modeling Language)** adalah bahasa standar untuk memvisualisasikan, merancang, dan mendokumentasikan sistem perangkat lunak berorientasi objek. UML memudahkan kita memahami struktur dan perilaku sistem dengan diagram.

Berikut adalah beberapa contoh diagram UML yang sering digunakan:

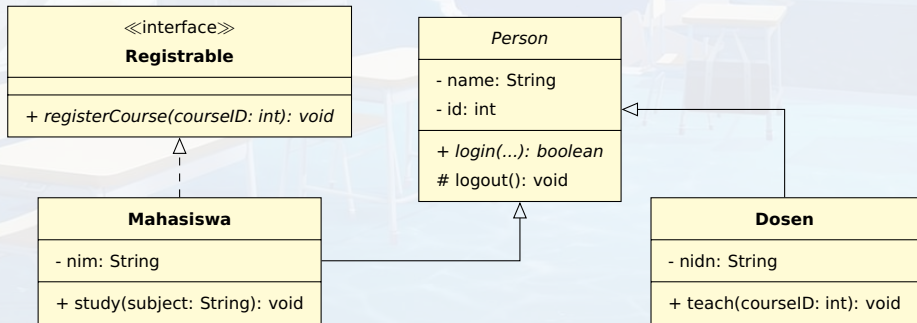
- Use Case Diagram
- Activity Diagram
- Sequence Diagram
- Class Diagram
- Statemachine Diagram
- Component Diagram

Namun, pada mata kuliah ini kita hanya akan membahas *Class Diagram*.

# Diagram Kelas

## Diagram Kelas

Representasi statis dari struktur sistem perangkat lunak. Diagram ini menggambarkan kelas-kelas dalam sistem, atribut-atributnya, metode-metode yang dimiliki, serta hubungan antar kelas.



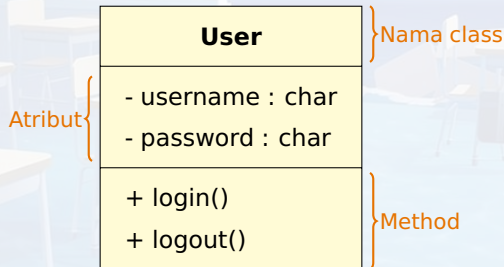
Gambar: Contoh Diagram Kelas



# Diagram Kelas

## Komponen

Diagram kelas memiliki tiga komponen penyusun. Berikut ini adalah komponen-komponennya:



Gambar: Komponen Diagram Kelas

# Diagram Kelas

## Komponen

Selain itu, beberapa simbol dan gaya penulisan juga berpengaruh dalam memaknai diagram kelas.

- - : Menandakan method atau variabel bersifat **private**
- + : Menandakan method atau variabel bersifat **public**
- # : Menandakan method atau variabel bersifat **protected**
- ~ : Menandakan method atau variabel bersifat **default**
- **method()** yang penulisannya miring menandakan bahwa method tersebut bersifat abstract (begitu juga dengan class).

# Daftar isi

## 1 Modifier

- Access Modifier
- Non-Access Modifier

## 2 Encapsulation

- Setter
- Getter

## 3 Diagram Kelas

- Komponen

## 4 Latihan

# Latihan 1

Buatlah *class* dari diagram kelas berikut ini!

Lingkaran
- radius : double = 0
# Lingkaran() # Lingkaran(double r) ~ setRadius(double r) ~ getRadius : double + luas() : double + keliling() : double

Tabung
- tinggi : double = 0 - alas : Lingkaran = 0
# Tabung() # Tabung(double t, Lingkaran circ) ~ setTinggi(double t) : double ~ setAlas(Lingkaran circ) : Lingkaran ~ getTinggi() : double ~ getAlas() : Lingkaran + luasAlas() : double + luasPermukaan() : double + volume() : double

Elips
- sbMinor : double = 0 - sbMayor : double = 0
# Elips() # Elips(double a, double b) ~ setMinor(double m) ~ setMayor(double M) ~ getMinor : double ~ getMayor : double + luas() : double + keliling() : double