



REVIEW OBJECT ORIENTED PROGRAMMING (OOP)

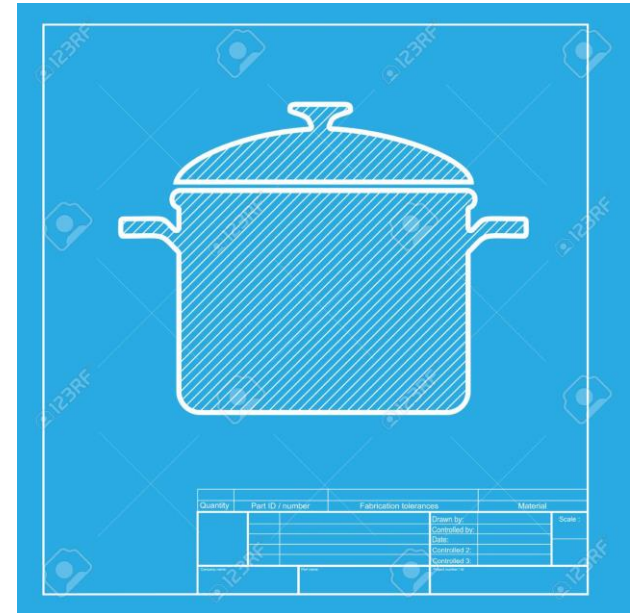


1. Class
2. Class Member
3. Constructor
4. Object

CLASS

- **Blueprint**/cetak biru/**rancangan** untuk membuat objek
- Deskripsi dari sekumpulan objek dengan **atribut** dan **perilaku** yang sama

```
public class Panci {  
    int diameter;  
    String warna;  
  
    String panciBerbunyi(){  
        return "Tuuuuuttt....";  
    }  
  
    void informasiPanci(){  
        System.out.println("Diameter panci "+diameter);  
        System.out.println("Warna panci "+warna);  
    }  
}
```



CLASS MEMBER

- Variabel (**atribut**) dan method (**perilaku**) yang didefinisikan di dalam sebuah class.
- Class member tersebut akan **dimiliki** oleh **objek** yang dibuat dari class tersebut

Perilaku/method
milik class

```
public class Panci {  
    int diameter;  
    String warna;  
  
    String panciBerbunyi(){  
        return "Tuuuuuttt....";  
    }  
  
    void informasiPanci(){  
        System.out.println("Diameter panci "+diameter);  
        System.out.println("Warna panci "+warna);  
    }  
}
```

Atribut/variabel milik class

CONSTRUCTOR

- **Cetakan** untuk membuat objek
- Digunakan untuk **memberi nilai awal** ketika instansiasi/pembuatan objek
- Memiliki **nama yang sama** dengan nama class-nya

```
public class Panci {  
    int diameter;  
    String warna;  
  
    String panciBerbunyi(){  
        return "Tuuuuuttt....";  
    }  
  
    void informasiPanci(){  
        System.out.println("Diameter panci "+diameter);  
        System.out.println("Warna panci "+warna);  
    }  
}
```

Jika constructor tidak dibuat di dalam class maka Java akan menggunakan **default constructor** ketika proses pembuatan/instansiasi objek

DEFAULT CONSTRUCTOR

```
public class Panci {  
    int diameter;  
    String warna;  
  
    String panciBerbunyi(){  
        return "Tuuuuuttt....";  
    }  
  
    void informasiPanci(){  
        System.out.println("Diameter  
panci "+diameter);  
        System.out.println("Warna  
panci "+warna);  
    }  
}
```

```
public class TesPanci {  
    public static void main(String[] args) {  
        Panci pc = new Panci();  
        pc.informasiPanci();  
    }  
}
```

```
Diameter panci 0  
Warna panci null  
PS D:\CODING JAVA\00 - Kuliah FPA\P9\P9>
```

Jika constructor tidak dibuat di dalam class maka Java akan menggunakan **default constructor** ketika proses pembuatan/instansiasi objek

OVERLOADING CONSTRUCTOR

- Membuat konstruktor lain dengan **nama sama** dan **berbeda parameter**
- Lebih fleksibel dalam pemberian nilai awal saat instansiasi/pembuatan objek
- Overloading juga berlaku untuk pembuatan method sebuah class

```
public class TesPanci {  
    public static void main(String[] args) {  
        Panci pc1 = new Panci(20, "Silver");  
        Panci pc2 = new Panci(30);  
        Panci pc3 = new Panci("Gold");  
    }  
}
```

```
public class Panci {  
    int diameter;  
    String warna;  
  
    Panci (int diameter, String warna){  
        this.diameter = diameter;  
        this.warna = warna;  
    }  
  
    Panci (int diameter){  
        this.diameter = diameter;  
    }  
  
    Panci (String warna){  
        this.warna = warna;  
    }  
}
```

FUNDAMEN

PENGEMBANGAN APLIKASI

Pertemuan 9



UNIVERSITAS
ISLAM
INDONESIA

VARIABEL DAN MODIFIER

Program Studi Informatika - Program Sarjana
Fakultas Teknologi Industri



TOPIK MATERI



1. Instance Variable
2. Class Variable
3. Local Variable
4. Access modifier: Private
5. Access modifier: Default
6. Access modifier: Public



VARIABEL





VARIABEL DI JAVA



1. Variabel adalah suatu tempat penyimpanan yang memiliki suatu nama dan dapat diisi atau diberi nilai
2. isi variabel dapat diubah atau dimanipulasi.
3. Tiga macam variable di Java:
 - a. Instance Variable
 - b. Class Variable
 - c. Local Variable

INSTANCE VARIABLE

1. **Instance Variable** adalah variabel yang dideklarasikan **di dalam** class, namun berada **di luar *methods*, *constructors*, atau *blok program***.
2. **Instance Variable** menyimpan nilai yang akan digunakan oleh lebih dari satu *methods*, *constructors*, *blok*, atau bagian penting lain dari objek.
3. Instance Variable:
 - a. **Dibuat** saat suatu objek dibuat atau diinstansiasi.
 - b. **Dihancurkan** saat objek dihancurkan.

INSTANCE VARIABLE

Instance Variable dari contoh **class Mahasiswa** di atas adalah **nim** dan **nama**.

Dideklarasikan

- Di dalam *class*
- Di luar *methods*
- Di luar *constructors*
- Di luar *blok program*

Contoh *instance variable*

src > **J** Mahasiswa.java > ...

```
1  public class Mahasiswa {
2      int NIM;
3      String nama;
4
5      Mahasiswa(int noMHS, String namaMHS){
6          NIM = noMHS;
7          nama = namaMHS;
8      }
9
10     String identitasToString(){
11         return "Nama mahasiswa: "+nama;
12     }
13 }
```

CLASS VARIABLE

1. Disebut juga **Static Variable**
2. Class Variable dideklarasikan dengan kata kunci **Static**.
3. Class Variable dideklarasikan **di dalam** *class*, namun berada **di luar** *methods, constructors*, atau suatu **blok program**.
4. **Perbedaan** dengan **Instance Variable**, jika Class Variable telah dideskripsikan nilainya, nilai tersebut akan **berlaku untuk semua objek** yang dibentuk/*diinstansiasi* dari suatu Class.
5. Class Variable:
 - **Dibuat** saat program dimulai.
 - **Dihancurkan** saat program dihentikan.

CLASS VARIABLE

```
3 public class Mahasiswa {
4     int nim;
5     String nama;
6     static String jurusan;
7
8     Mahasiswa(int nim, String nama){
9         this.nim = nim;
10        this.nama = nama;
11    }
12
13    String identitasToString(){
14        return nama+" kuliah di jurusan "+jurusan;
15    }
16 }
```

```
run:
Nussa kuliah di jurusan null
Rara kuliah di jurusan null
Nussa kuliah di jurusan Informatika
Rara kuliah di jurusan Informatika
BUILD SUCCESSFUL (total time: 0 seconds)
```

```
3 public class 09_ModifierVar {
4     public static void main(String[] args) {
5         Mahasiswa mhs1 = new Mahasiswa(11, "Nussa");
6         Mahasiswa mhs2 = new Mahasiswa(22, "Rara");
7         System.out.println(mhs1.identitasToString());
8         System.out.println(mhs2.identitasToString());
9         mhs1.jurusan = "Informatika";
10        System.out.println(mhs1.identitasToString());
11        System.out.println(mhs2.identitasToString());
12    }
13 }
```

- Perhatikan bahwa variable **jurusan** di objek **mhs1** dan **mhs2** bernilai sama, *Informatika*.
- Variable **jurusan** dideskripsikan dengan **static**.

CLASS VARIABLE

1. Class Variable merupakan variable milik Class, maka **Class Variable** dapat diakses langsung melalui **Class** (tanpa membuat objeknya terlebih dahulu)

2. Contoh:

Code di main() method:

```
3 public class 09_ModifierVar {  
4     public static void main(String[] args) {  
5         Mahasiswa.jurusan = "Informatika";  
6     }  
7 }
```

3. Perhatikan bahwa variable **jurusan** dapat diakses langsung melalui class **Mahasiswa** tanpa harus membentuk objek dari class Mahasiswa terlebih dahulu.

LOCAL VARIABLE

1. Variabel lokal adalah variable yang dideskripsikan **di dalam** *methods, constructors*, atau *blok program*.
2. Variabel lokal:
 - a. **Dibuat** saat program mengeksekusi *methods, constructors*, atau *blok*.
 - b. **Dihancurkan** saat program keluar dari *methods, constructors*, atau *blok*.
3. Variabel lokal **tidak dapat** diakses di luar *methods, constructors*, atau *blok* yang mendeskripsikannya.

LOCAL VARIABLE

```
3 public class Mahasiswa {
4     int nim;
5     String nama;
6
7     void hitungNilai(int uts, int uas){
8         int nilai;
9         nilai = (uts+uas)/2;
10    }
11 }
```

- Variabel **nilai** merupakan **variable lokal** yang dideskripsikan di dalam *methods* **hitungNilai()**.
- Variabel **nilai** dibuat saat program mengeksekusi *methods* **hitungNilai()** dan akan dihancurkan saat program keluar dari *methods* **hitungNilai()**.

```
3 public class Mahasiswa {
4     int nim;
5     String nama;
6
7     void hitungNilai(int uts, int uas){
8         int nilai;
9         nilai = (uts+uas)/2;
10    }
11
12    String nilaiToString(){
13        return "Nilai akhir: "+nilai;
14    }
15 }
```

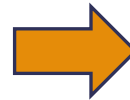
cannot find symbol
symbol: variable nilai
location: class Mahasiswa

(Alt-Enter shows hints)

- Jika variabel **nilai** coba diakses di luar *methods* **hitungNilai()** maka akan memberikan pesan **error**.

LOCAL VARIABLE

```
3 public class Mahasiswa {
4     int nim;
5     String nama;
6
7     void hitungNilai(int uts, int uas){
8         int nilai;
9         nilai = (uts+uas)/2;
10    }
11
12    String nilaiToString(){
13        return "Nilai akhir: "+nilai;
14    }
15 }
```



- Agar variabel **nilai** dapat diakses di luar methods **hitungNilai()** maka variabel **nilai** perlu dideskripsikan di luar methods **hitungNilai()**.

```
3 public class Mahasiswa {
4     int nim;
5     String nama;
6     int nilai;
7
8     void hitungNilai(int uts, int uas) {
9         nilai = (uts + uas) / 2;
10    }
11
12    String nilaiToString() {
13        return "Nilai akhir: " + nilai;
14    }
15 }
```



CLASS METHOD



CLASS METHOD

1. Seperti class variable, **Class Method** juga dideklarasikan dengan kata kunci **Static**.
2. Sehingga disebut juga **Static Method**
3. Class Method adalah **method milik class bukan objek**, artinya Class Method dapat diakses langsung melalui Class.
 - Jika sebuah method sebuah class **tidak dideklarasikan** dengan kata kunci **static** maka method tersebut **hanya dapat diakses melalui objek** yang sudah diinstansiasi dari class tersebut.

CLASS METHOD

```
3 public class Mahasiswa {
4     int nim;
5     String nama;
6     static String jurusan;
7
8     static String cekStatusLulus(int nilai){
9         String lulus = (nilai >= 80) ? "Lulus" : "Tidak lulus" ;
10        return lulus;
11    }
12 }
```

Code di **main()** method:

```
3 public class 09_ModifierVar {
4     public static void main(String[] args) {
5
6         Mahasiswa.cekStatusLulus(83);
7
8     }
9 }
```

- Perhatikan bahwa *method* **cekStatusLulus()** diberi *modifier* **static**.
- Hal ini membuat *method* **cekStatusLulus()** dapat langsung diakses melalui *class*-nya **tanpa** perlu menginstansiasi objek terlebih dahulu.

CLASS METHOD

Batasan pada **Static Method**:

1. **Static Method** tidak dapat menggunakan *non-static class member* atau memanggil *non-static methods* secara langsung.
2. **Static Method** tidak dapat mengandung *class member* dengan kata kunci **this** dan **super**.



ACCESS MODIFIER





ACCESS MODIFIER DI JAVA



1. **Access Modifier** berfungsi untuk mengatur aksesibilitas suatu *variables, methods, dan constructors*.
2. Empat macam Access Modifier di Java:
 - a. Private
 - b. Default
 - c. Public
 - d. Protected (akan dibahas di materi **Pewarisan**)

PRIVATE

1. Dideklarasikan dengan kata kunci **private**.
2. *Methods, variables, atau constructors* yang dideklarasikan dengan kata kunci **private**:
 - **Hanya** dapat **diakses di dalam class** yang memuat deklarasi *methods, variables, atau constructors* tersebut.
 - Objek hasil instansiasi class tersebut **TIDAK DAPAT** mengaksesnya

PRIVATE

```
3 public class Mahasiswa {
4     int nim;
5     private String nama;
6
7     String identitasToString(){
8         return "Nama mahasiswa: "+nama;
9     }
10 }
```

```
3 public class O9_ModifierVar {
4     public static void main(String[] args) {
5         Mahasiswa mhs1 = new Mahasiswa();
6
7         mhs1.nim = 123;
8         mhs1.nama = "Nussa";
9     }
10 }
```

nama has private access in Mahasiswa

(Alt-Enter shows hints)

- Perhatikan bahwa variable **nama** diberi kata kunci **private** sedangkan variable **nim** tidak.
- Saat objek **mhs1** dibentuk di *main() method* yang *berbeda class*:
 - Variable **nama tidak dapat diakses** di class **O9_ModifierVar** sedangkan variable **nim** dapat diakses.

ACCESS MODIFIER DI JAVA

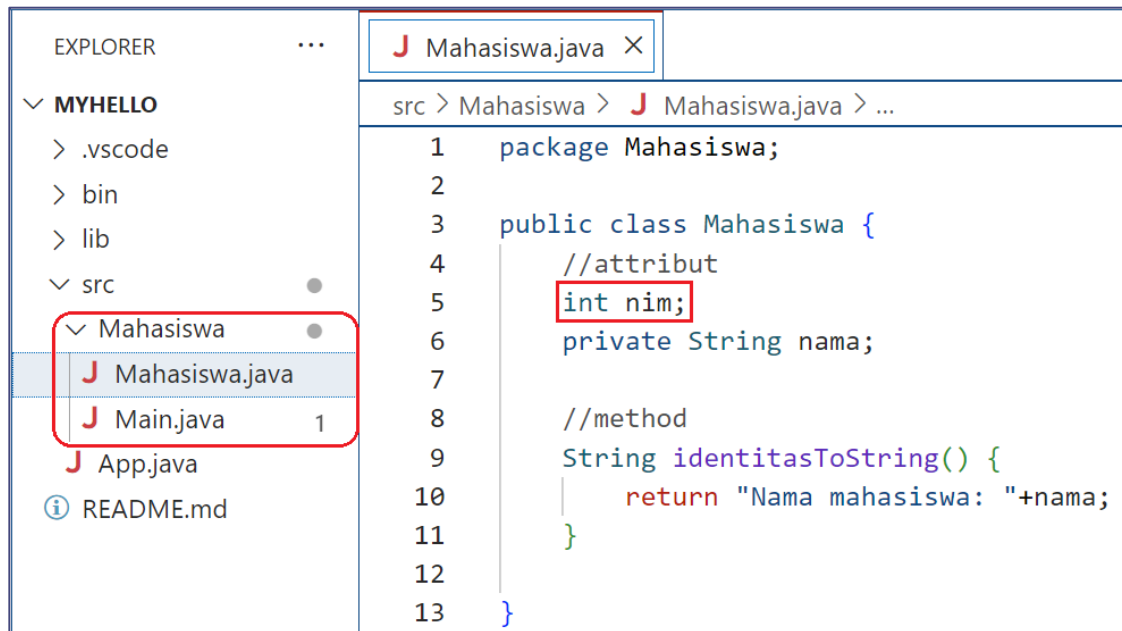
Access Modifier	Class yang sama	Class di Package yang sama	Subclass (di package lain)	Class manapun (di package lain)
Private	✓	-	-	-
Default				
Public				
Protected				

✓ : bisa diakses

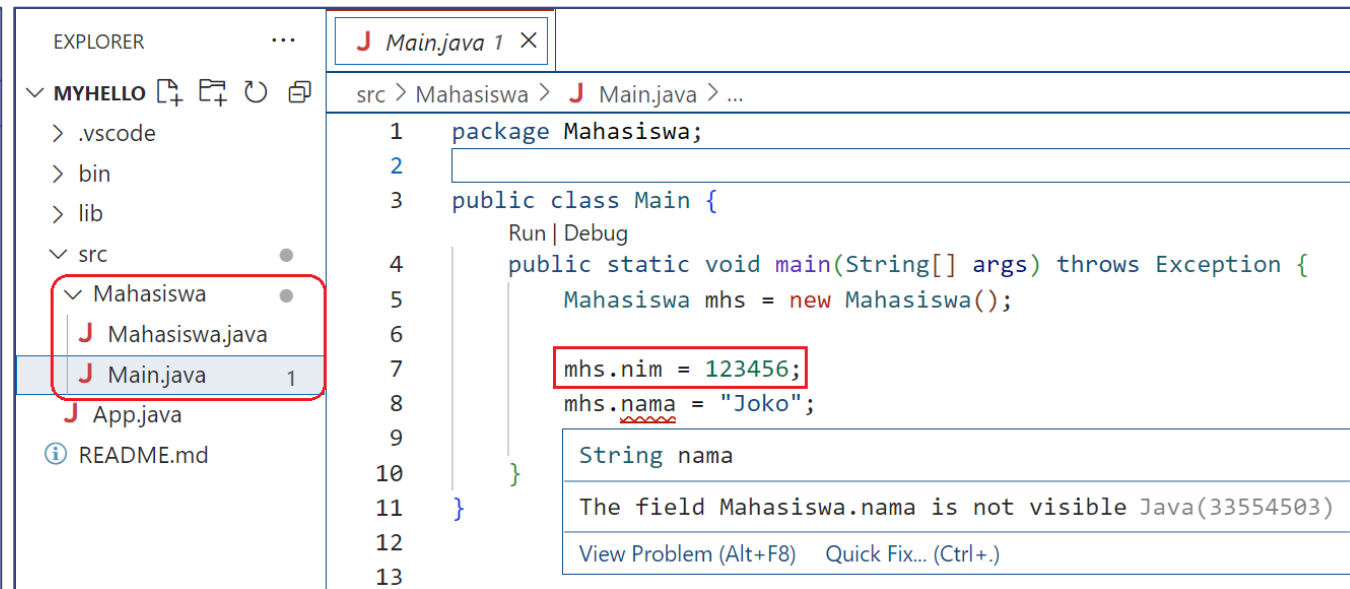
DEFAULT

1. Jika tidak ada ***modifier*** atau kata kunci yang ditulis secara eksplisit dalam deklarasi *methods*, *variables*, atau *constructors*:
 - Akses yang dimiliki oleh *methods*, *variables*, atau *constructors* tersebut adalah **default**.
2. *Methods*, *variables*, atau *constructors* yang memiliki akses ***modifier*** **default** maka:
 - **Dapat** diakses di luar class-nya
 - Tetapi **tidak dapat** diakses di luar **package** yang memuat class tersebut.

DEFAULT



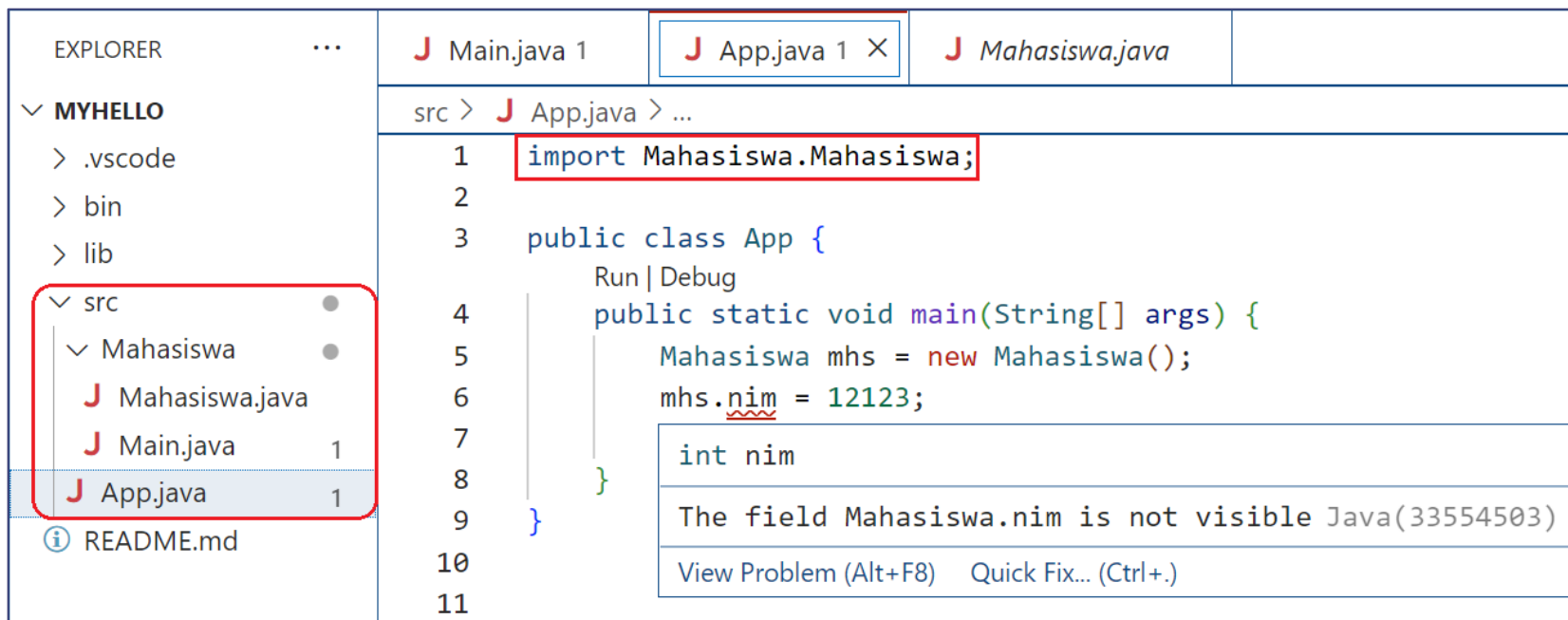
```
1 package Mahasiswa;
2
3 public class Mahasiswa {
4     //attribut
5     int nim;
6     private String nama;
7
8     //method
9     String identitasToString() {
10         return "Nama mahasiswa: "+nama;
11     }
12
13 }
```



```
1 package Mahasiswa;
2
3 public class Main {
4     Run | Debug
5     public static void main(String[] args) throws Exception {
6         Mahasiswa mhs = new Mahasiswa();
7         mhs.nim = 123456;
8         mhs.nama = "Joko";
9     }
10
11 }
12
13
```

- Perhatikan bahwa variable **nim** tidak dideklarasikan dengan access *modifier* secara eksplisit, artinya variable **nim** memiliki access *modifier* **default**.
- Variable **nim** masih bisa diakses di class lain **dalam satu package**, yaitu class **Main**.
- Bagaimana kalau diakses oleh class yang **berbeda package**?

DEFAULT



The screenshot shows the VS Code interface. On the left, the Explorer panel displays a project named 'MYHELLO' with a 'src' folder containing 'Mahasiswa.java', 'Main.java', and 'App.java'. The 'App.java' file is selected. The main editor shows the code for 'App.java' with the following content:

```
1 import Mahasiswa.Mahasiswa;
2
3 public class App {
4     Run | Debug
5     public static void main(String[] args) {
6         Mahasiswa mhs = new Mahasiswa();
7         mhs.nim = 12123;
8     }
9 }
10
11
```

A red box highlights the first line of code: `import Mahasiswa.Mahasiswa;`. Below the code editor, an error message is displayed: 'The field Mahasiswa.nim is not visible Java(33554503)'. The error message also includes a 'View Problem (Alt+F8)' and a 'Quick Fix... (Ctrl+.)' option.

- Variable **nim** tidak bisa diakses di class lain yang berbeda package
- Bagaimana caranya agar variabel **nim** dapat diakses oleh class yang berbeda package?

Perhatikan! Untuk membuat objek dari kelas yang berbeda package maka perlu dilakukan proses **import**

import <nama_package>.<nama_class>

Atau dengan cara: **import <nama_package>.***

Yang berarti akan meng-import semua class yang ada di dalam package tersebut

ACCESS MODIFIER DI JAVA

Access Modifier	Class yang sama	Class di Package yang sama	Subclass (di package lain)	Class manapun (di package lain)
Private	✓	-	-	-
Default	✓	✓	-	-
Public				
Protected				

✓ : bisa diakses



PUBLIC



1. Dideklarasikan dengan kata kunci **public**.
2. *Methods, variables, atau constructors* yang dideklarasikan dengan kata kunci **public** dapat diakses di class manapun (di dalam maupun di luar package).
3. Access modifier **public** merupakan access modifier yang memiliki aksesibilitas paling luas di antara access modifier yang lain.

DEFAULT → PUBLIC

```
J Mahasiswa.java ×
src > Mahasiswa > J Mahasiswa.java > ...
1 package Mahasiswa;
2
3 public class Mahasiswa {
4     //attribut
5     public int nim;
6     private String nama;
7
8     //method
9     String identitasToString() {
10         return "Nama mahasiswa: "+nama;
11     }
12
13 }
```

- Agar variabel **nim** dapat diakses oleh class yang **berbeda package** maka perlu nditambahkan modifier **public** pada variabel **nim**.

```
J App.java ×
src > J App.java > ...
1 import Mahasiswa.Mahasiswa;
2
3 public class App {
4     Run | Debug
5     public static void main(String[] args) {
6         Mahasiswa mhs = new Mahasiswa();
7         mhs.nim = 12123;
8     }
9 }
```

Terlihat bahwa sudah tidak muncul *error* dan variable **nim** sudah dapat diakses oleh class **App** yang berbeda package.

ACCESS MODIFIER DI JAVA

Access Modifier	Class yang sama	Class di Package yang sama	Subclass (di package lain)	Class manapun (di package lain)
Private	✓	-	-	-
Default	✓	✓	-	-
Public	✓	✓	✓	✓
Protected				

✓ : bisa diakses

CLASS DIAGRAM

CLASS DIAGRAM

Dalam membuat Class Diagram menggunakan aplikasi StarUML, berikut simbol modifier yang telah kita pelajari pada pertemuan kali ini.

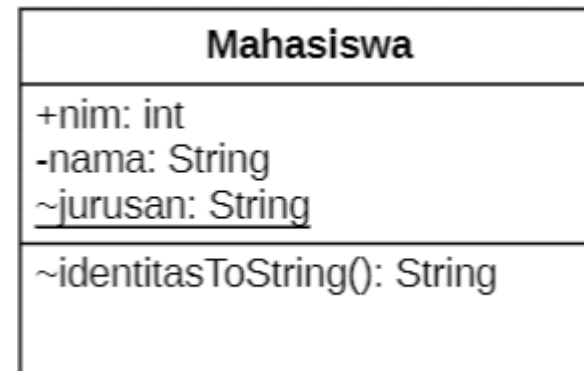
- Static: diberi garis bawah (underline)
- Private: -
- Default: ~
- Public: +

CLASS DIAGRAM

```
3 public class Mahasiswa {  
4     public int nim;  
5     private String nama;  
6     static String jurusan;  
7  
8     String identitasToString(){  
9         return nama+" kuliah di jurusan "+jurusan;  
10    }  
11 }
```

1. Atribut **nim**: **public**.
2. Atribut **nama**: **private**.
3. Atribut **jurusan**: **default** dan **static**.
4. Method **identitasToString()**: **default**.

Class diagram dari code di samping:



“TERIMA KASIH”

FUNDAMEN **PENGEMBANGAN APLIKASI**

Pertemuan 9

Program Studi Informatika - Program Sarjana
Fakultas Teknologi Industri