ACCESS MODIFIER

MIKE YULIANA

OUTLINE

- Access Modifier public, default, protected, private
- Key word Static

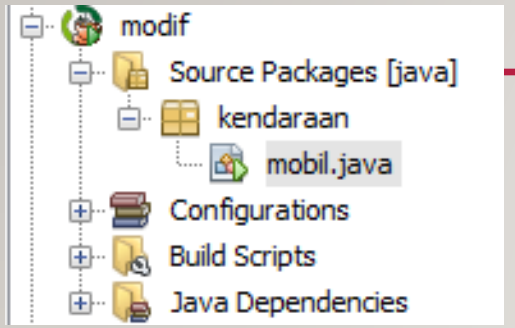
ACCESS MODIFIER

- Sebuah *"hak akses"* yang diberikan kepada variabel, method atau class yang bertujuan untuk menjaga integritas dari data ketika ingin diakses oleh object lain.
- Dengan adanya Access Modifier, kita dapat membatasi resource-resource mana saja yang dapat diakses oleh object tertentu, turunannya, ataupun oleh method tertentu.

TINGKAT HAK AKSES

Modifier	Class	Package	Sub Class	World
No Modifier	Y	Y	N	N
Public	Y	Y	Y	Y
Protected	Y	Y	Y	N
Private	Y	N	N	N

NO MODIFIER (1)

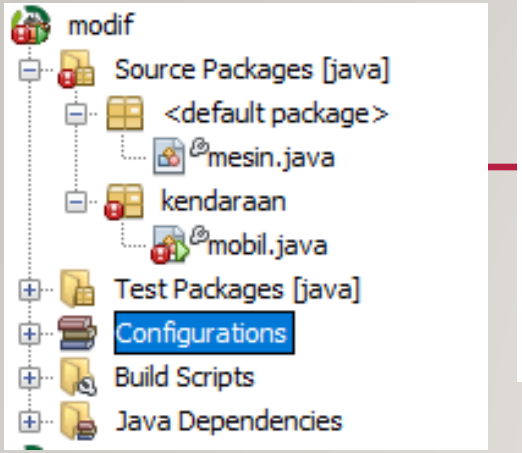



```
package kendaraan;

class mesin{

    double kecepatan;

    void setKecepatan(double kecepatan){
        this.kecepatan = kecepatan;
    }

    double getKecepatan(){
        return kecepatan;
    }
}
```

Hanya bisa diakses jika berada dalam package yang sama

```
public class mobil {
    public static void main(String[] args) {
        //Membuat Objek dari Class Mesin
        mesin data = new mesin();
        data.setKecepatan(360.0);
        System.out.println("Kecepatan Mobil: "+data.getKecepatan()+" Km/Jam");
    }

}
```

NO MODIFIER (2)




```java
class mesin{

    double kecepatan;

    void setKecepatan(double kecepatan){
        this.kecepatan = kecepatan;
    }

    double getKecepatan(){
        return kecepatan;
    }
}
```

```java
package kendaraan;

public class mobil {
    public static void main(String[] args) {
        //Membuat Objek dari Class Mesin
        mesin data = new mesin();
        data.setKecepatan(360.0);
        System.out.println("Kecepatan Mobil: "+data.getKecepatan()+" Km/Jam");
    }

}
```

Output :

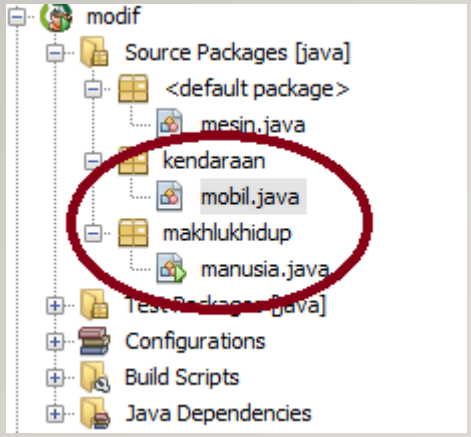
```
19
20
21
22
23
24
25
26
27
```

```
cannot find symbol
  symbol:   class mesin
  location: class mobil

cannot find symbol
  symbol:   class mesin
  location: class mobil
----
(Alt-Enter shows hints)
```

```
patan(){
epatan;

 {
oid main(String[] args) {
Objek dari Class Mesin
```

PUBLIC (1)

Hak Akses *public* digunakan pada class/variable/method/konstuktor, agar dapat diakses oleh seluruh class didalam package yang sama atau diluar package yang berbeda, modifier jenis ini memiliki tingkat akses yang sangat luas, hingga seluru sumber daya dapat diakses oleh class manapun tanpa batasan



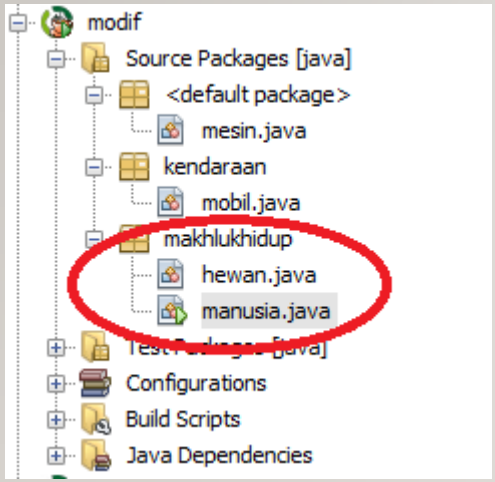

```java
package kendaraan;
```

```java
public class mobil{

    public String merek;
    public double speed;

    public void setMerek(String merek){
        this.merek = merek;
    }
    public void setSpeed(int speed){
        this.speed = speed;
    }

    public String getMerek(){
        return merek;
    }
    public double getSpeed(){
        return speed;
    }
}
```

```java
package makhlukhidup;

import kendaraan.mobil;

/**
 *
 * @author HP
 */
public class manusia {
    public static void main(String[] args) {
        mobil a=new mobil();
        a.setMerek("SUZUKI");
        a.setSpeed(280);
        System.out.println("Saya Mempunyai Mobil "+a.getMerek());
        System.out.println("Dengan Kecepatan "+a.getSpeed()+" Km/jam");
    }

}
```

PROTECTED (1)

Variable, Method atau konstruktor yang dideklarasikan protected dapat diakses oleh subclass atau class lain Asalkan didalam satu package yang sama



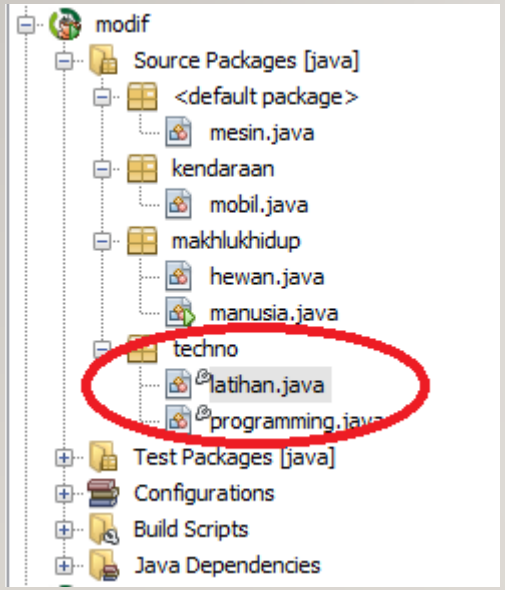

```java
package makhlukhidup;

/**
 *
 * @author HP
 */
public class hewan {
    protected String makanan = "Daging dan Ikan";
    protected boolean hidup;

    protected void setHidup(boolean hidup){
        this.hidup = hidup;
    }
    protected boolean getHidup(){
        return hidup;
    }
}
```

```java
package makhlukhidup;
```

```java
public class manusia extends hewan{
    public static void main(String[] args){
        hewan data = new hewan();
        data.setHidup(true);
        System.out.println("Makanan Saya: "+data.makanan);
        System.out.println("Saya Hidup: "+data.getHidup());
    }
}
```

PRIVAT (1)

- Variable dan method yang diberikan hak akses *private* hanya bisa diakses oleh class itu sendiri, data-data tersebut tidak bisa diwariskan pada subclass atau class lainnya
- Agar class lain bisa mengkases variable atau method tersebut maka perlu dibuatkan method yang mempunyai hak akses public (package yang berbeda) atau protected (pada package yang sama)

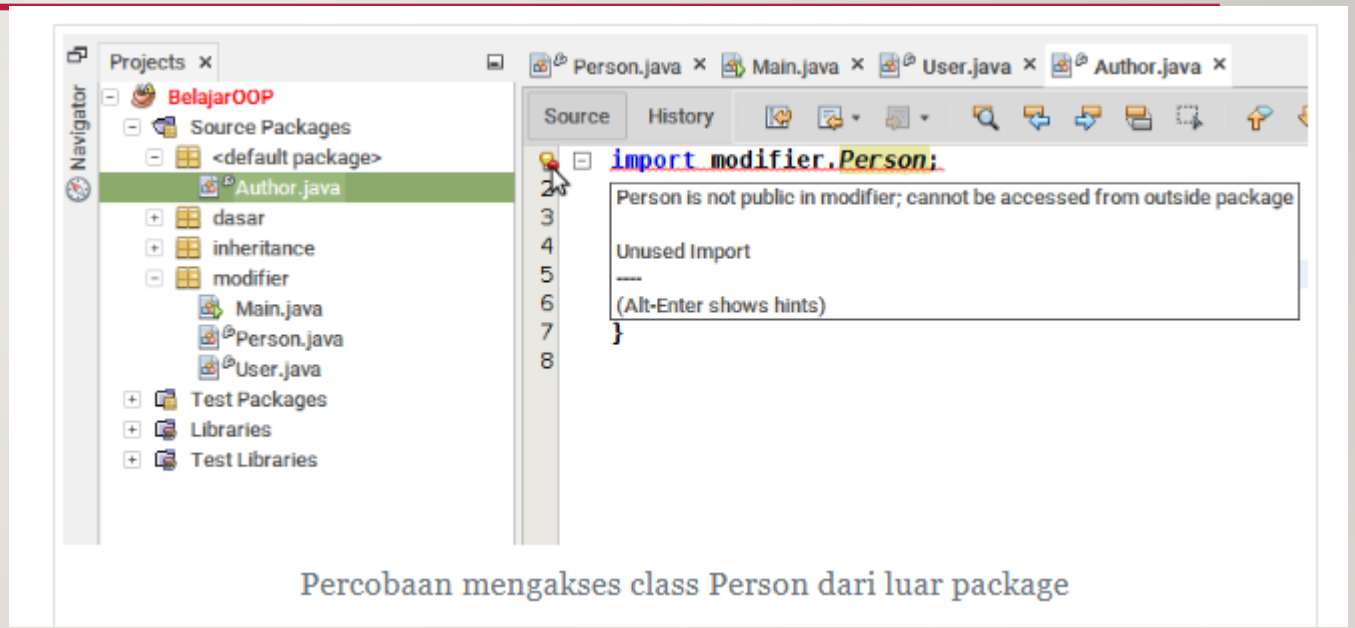



```java
package techno;

/**
 *
 * @author HP
 */
class programming {
    private String language = "Java";

    public String getLanguage(){
        return language;
    }
}
```

```java
package techno;

/**
 *
 * @author HP
 */
public class latihan extends programming{
    public static void main(String[] args) {
        latihan data = new latihan();
        System.out.println("Bahasa Pemrograman: "+data.getLanguage());
        ystem.out.println("Bahasa Pemrograman: "+data.language);
    }
}
```

SOAL (1)

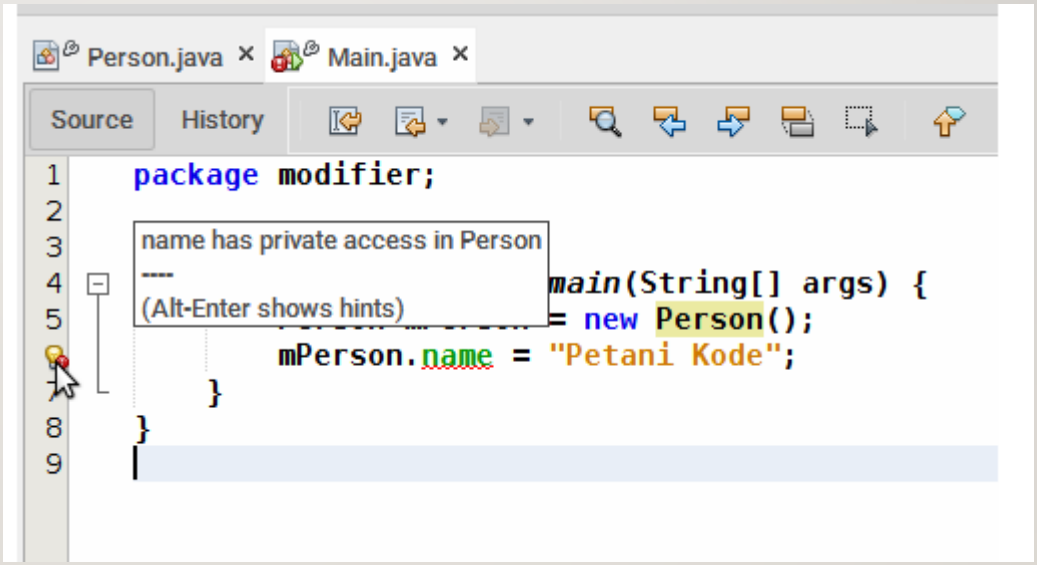
```java
package modifier;

class Person {
    public String name;

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



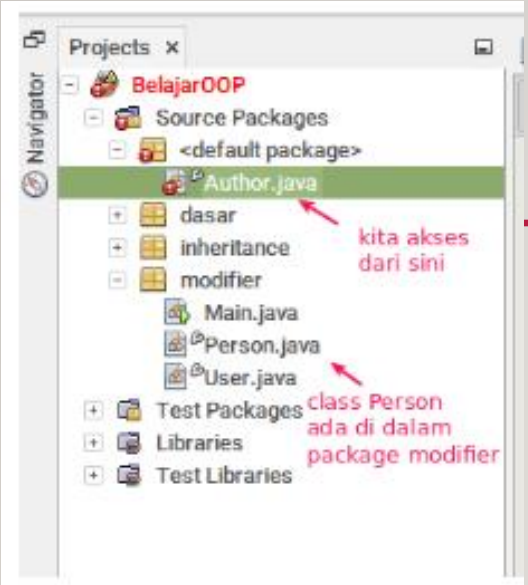
Percobaan mengakses class Person dari luar package

SOAL (2)

```
class Person {
    private String name;

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

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



SOAL (3)




```java
package modifier;

public class Person {
    protected String name;

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

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

```java
import modifier.Person;

public class Author {

    Person p = new Person();

    public Author() {
        // akan terjadi error di sini karena atribut name
        // telah diberikan modifier protected
        p.name = "Petani Kode";
    }

}
```

KATA KUNCI(KEYWORD) STATIC

Bentuk umum: nama kelas.nama metode

bukan

variabelreferensiobjek.nama metode

Contoh:

```
class matematika
{
    static public double kuadrat(double nilai)
    {
        return nilai*nilai;
    }
}

public class coba
{
     public static void  main(String args[])
     {
        double bilangan=matematika.kuadrat(25.0);
        System.out.println(bilangan);
     }

}
```

nama kelas.nama metode

contoh:

```
public class coba4{
static int pencacah=0;
int nilai;

coba4(int nilai){
this.nilai=nilai;
this.pencacah++;
}

public void info(){
System.out.println(this.nilai);
System.out.println(this.pencacah);
}
}
```

```
class coba{
public static void main(String args[]){
coba4 a=new coba4(4);
a.info();

coba4 b=new coba4(5);
b.info();

coba4 c=new coba4(6);
c.info();
}
}
```

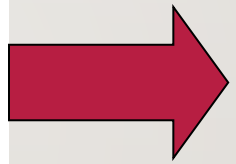
Hasil:

```
4
1
5
2
6
3
```

CONTOH LAIN:

```
public class coba5{
static int x=10,y=20;
}

class coba{
public static void main(String args[]){
coba5 a=new coba5( );
coba5 b=new coba5( );

System.out.println(a.x);
System.out.println(a.y);

b.x=25;
b.y=55;

System.out.println(a.x);
System.out.println(a.y);



}
}
```

Hasil:



```
10
20
25
55
```