

ACCESS MODIFIER

MIKE YULIANA

OUTLINE

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

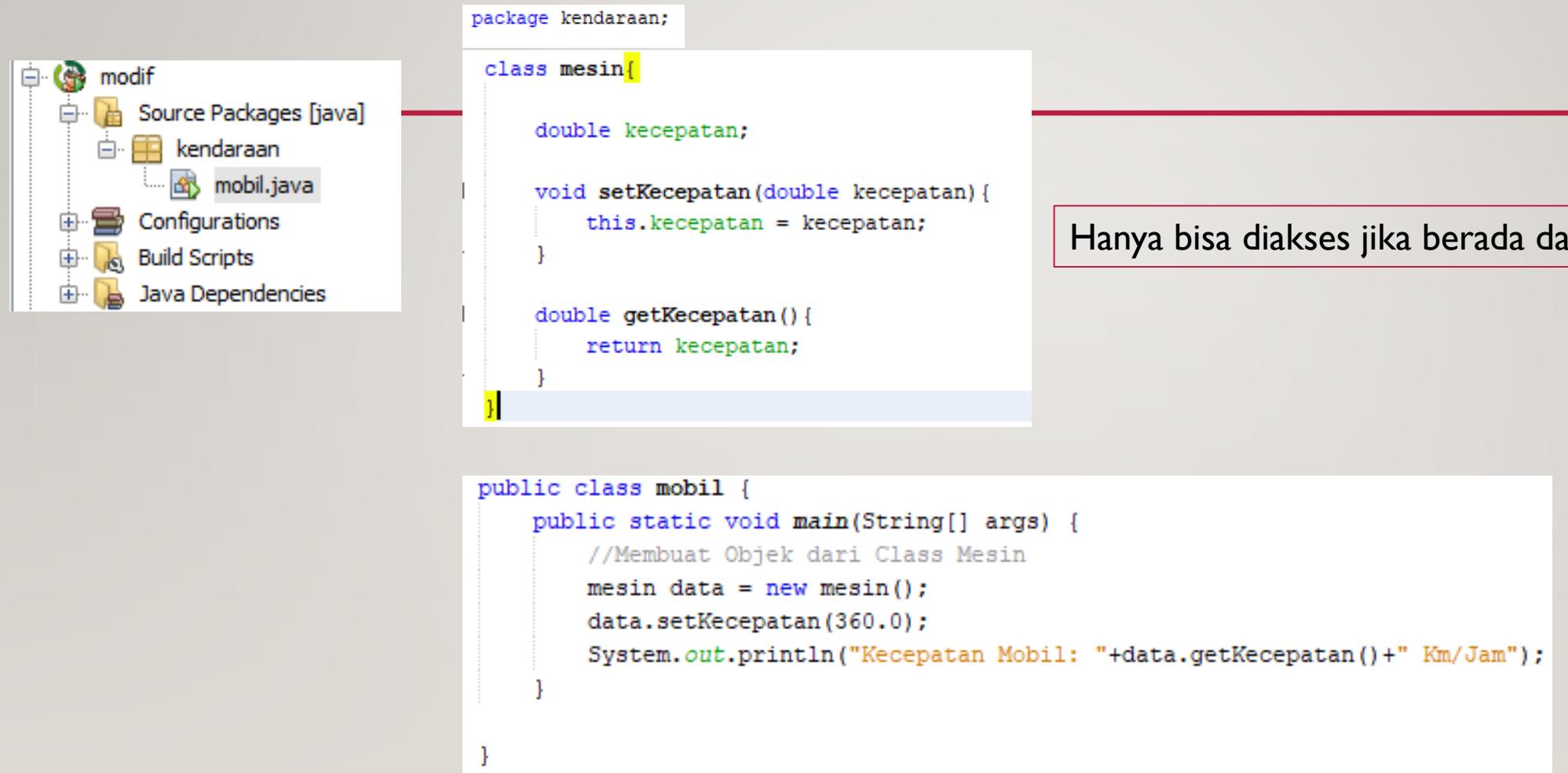
ACCESS MODIFIER

- 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 (I)



The screenshot shows a Java project structure in an IDE. The project is named 'modif' and contains a package named 'kendaraan' which includes a file 'mobil.java'. The code for 'mesin.java' is as follows:

```
package kendaraan;

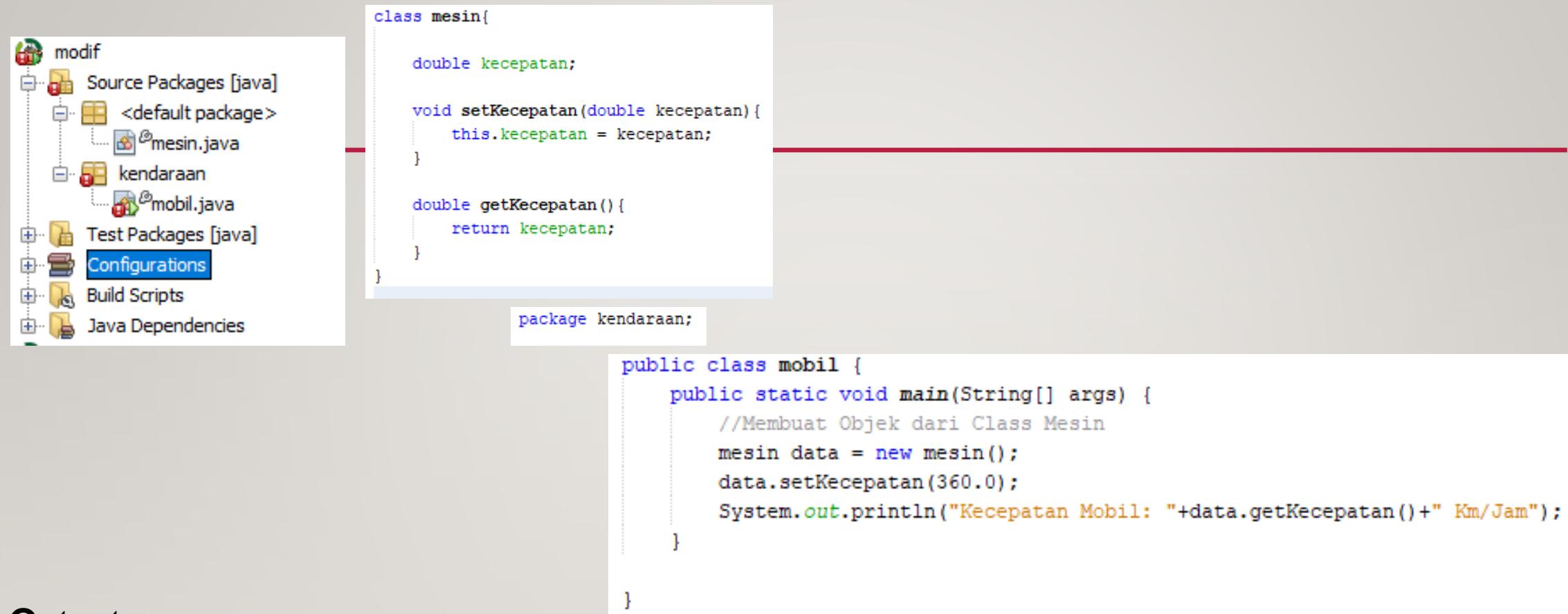
class mesin{
    double kecepatan;
    void setKecepatan(double kecepatan) {
        this.kecepatan = kecepatan;
    }
    double getKecepatan() {
        return kecepatan;
    }
}
```

The code for 'mobil.java' is as follows:

```
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");
    }
}
```

Hanya bisa diakses jika berada dalam package yang sama

NO MODIFIER (2)



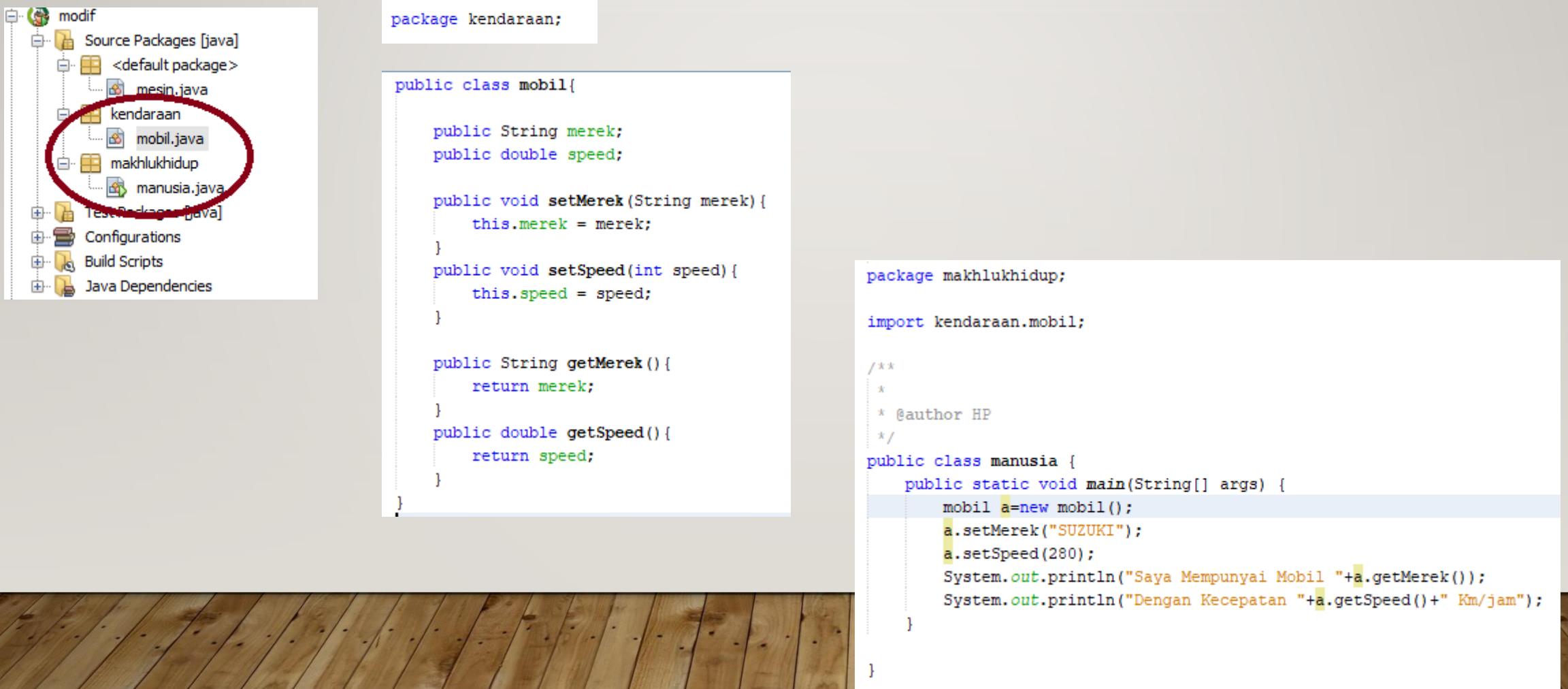
```
class mesin{  
    double kecepatan;  
  
    void setKecepatan(double kecepatan) {  
        this.kecepatan = kecepatan;  
    }  
  
    double getKecepatan() {  
        return kecepatan;  
    }  
}  
  
package kendaraan;  
  
public class mobil {  
    public static void main(String[] args)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

Output :

```
19  cannot find symbol  
20      symbol:  class mesin  
21      location: class mobil  batan() {  
22          epatan;  
23  cannot find symbol  
24      symbol:  class mesin  
25      location: class mobil  {  
26          void main(String[] args) {  
27              Objek dari Class Mesin  
(Alt-Enter shows hints)
```

PUBLIC (I)

Hak Akses *public* digunakan pada class/variable/method/konstruktor, 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



```
package kendaraan;

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;
    }
}
```

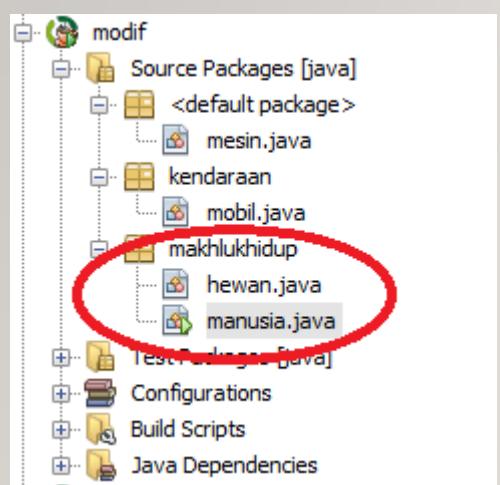
```
package makhluhidup;

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 (I)

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



```
package makhukhidup;

/*
 * @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;
    }
}
```

```
package makhukhidup;

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 (I)

- 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 mengakses variable atau method tersebut maka perlu dibuatkan method yang mempunyai hak akses *public* (package yang berbeda) atau *protected* (pada package yang sama)

The screenshot shows an IDE interface with a project structure on the left and two code editors on the right.

Project Structure:

- modif
- Source Packages [java]
 - <default package>
 - mesin.java
 - kendaraan
 - mobil.java
 - makhlukhidup
 - hewan.java
 - manusia.java
 - techno
 - latihan.java
 - programming.java
- Test Packages [java]
- Configurations
- Build Scripts
- Java Dependencies

A red circle highlights the "techno" package and its files.

Code Editor 1 (Top):

```
package technico;

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

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

Code Editor 2 (Bottom):

```
package technico;

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

SOAL (I)

```
package modifier;

class Person {
    public String name;

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

The screenshot shows a Java development environment with the following details:

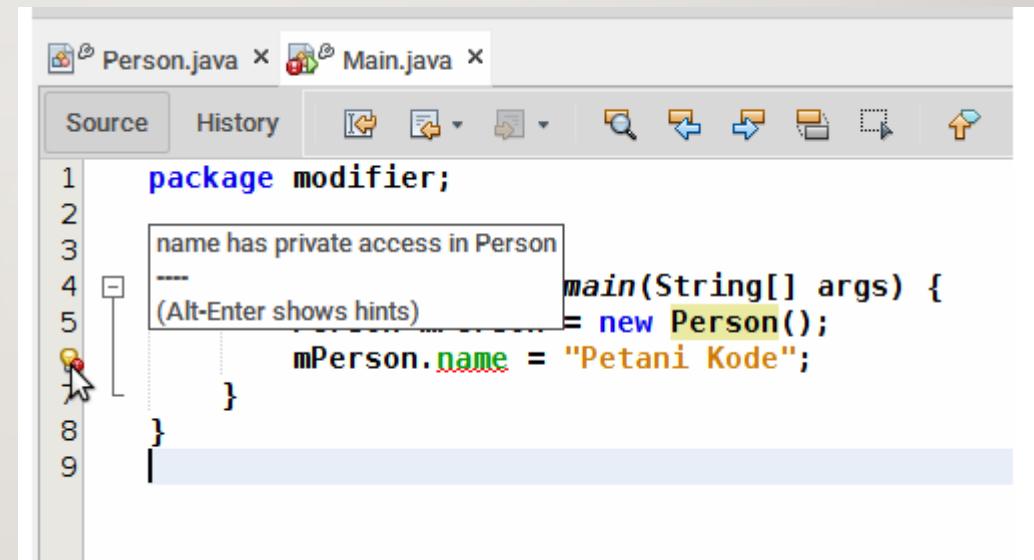
- Projects View:** Shows a project named "BelajarOOP" with Source Packages and Test Packages. Under Source Packages, there is a package named "modifier" containing files: dasar.java, inheritance.java, Main.java, Person.java, and User.java.
- Navigator View:** Shows the same file structure as the Projects view.
- Editor View:** Displays the code for "Author.java".
- Toolbars and Menus:** Standard Java IDE toolbars and menus are visible at the top.
- Message Bar:** A message bar at the bottom of the editor window displays the following text:

```
import modifier.Person;
Person is not public in modifier; cannot be accessed from outside package
Unused Import
---
```
- Status Bar:** The status bar at the bottom right indicates "(Alt-Enter shows hints)".

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;  
    }  
}
```

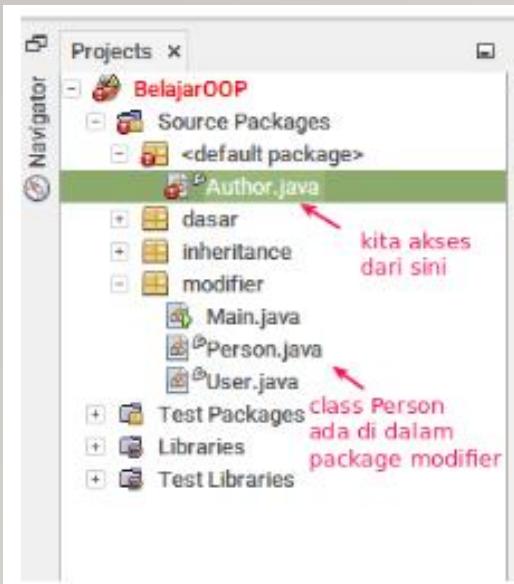


The screenshot shows an IDE interface with two tabs: Person.java and Main.java. The Main.java tab is active, displaying the following code:

```
package modifier;  
---  
name has private access in Person  
---  
(Alt-Enter shows hints)  
main(String[] args) {  
    Person p = new Person();  
    p.name = "Petani Kode";  
}  
}
```

A tooltip is visible above the line 'name has private access in Person'. The line 'name has private access in Person' is highlighted in yellow, and the word 'name' in the line 'p.name = "Petani Kode"' is also highlighted in yellow. A cursor is visible near the start of the main method.

SOAL (3)



```
package modifier;

public class Person {
    protected String name;

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

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

```
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

Contoh:

variabelreferensiobjek.nama metode

```
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();  
    }  
}
```

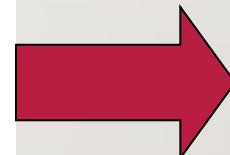
Hasil:

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