

Pemrograman Berorientasi Obyek

Enkapsulasi

Oleh Politeknik Elektronika Negeri Surabaya
2017



Politeknik Elektronika Negeri Surabaya
Departemen Teknik Informatika dan Komputer

1

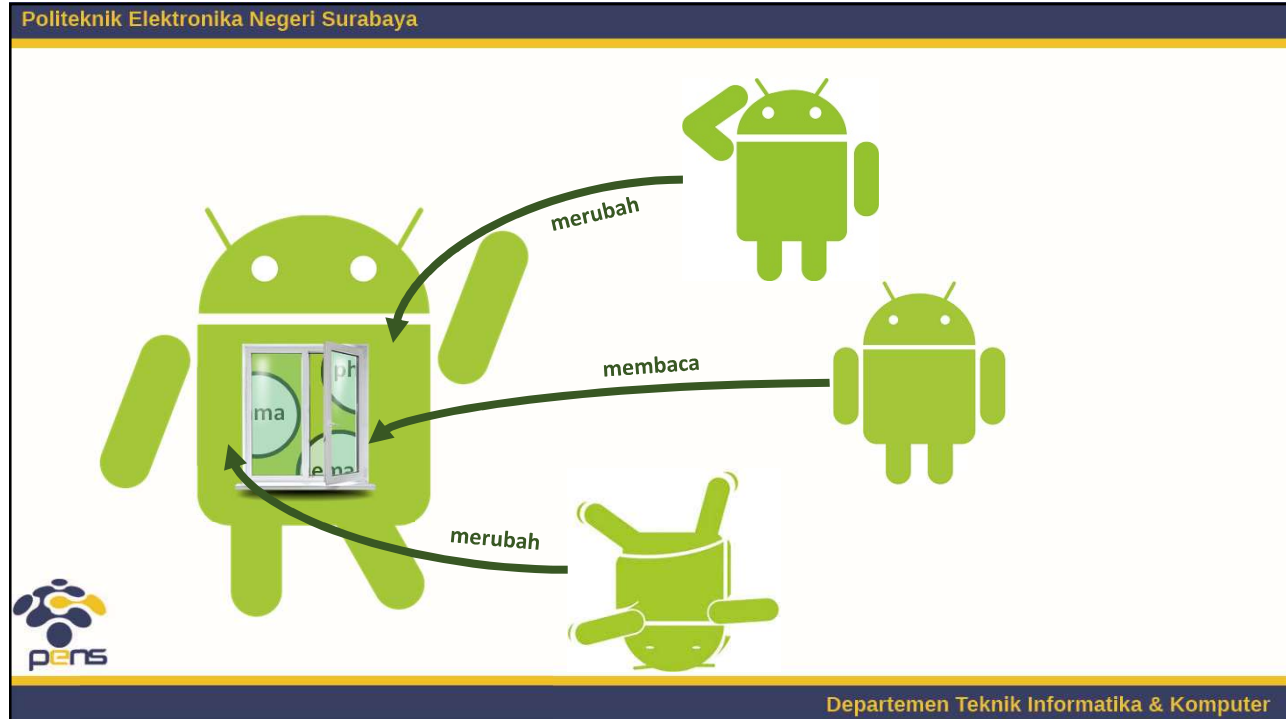
Politeknik Elektronika Negeri Surabaya

APA YANG DIMAKSUD ENKAPSULASI (*ENCAPSULATION*)?



Departemen Teknik Informatika & Komputer

2



3

Politeknik Elektronika Negeri Surabaya

Apa gunanya disembunyikan jika ternyata masih diberikan jalan untuk mengakses?

PENS

Departemen Teknik Informatika & Komputer

4



- Enkapsulasi adalah menyembunyikan informasi/data/attribute.
- Tujuannya untuk melindungi informasi agar tidak dapat diakses secara sembarangan.



Lindungi data dengan enkapsulasi

| Date |
|--------------|
| + day: int |
| + month: int |
| + year: int |
| |

```
Date myBirthDay = new Date();
myBirthDay.day = 32;
myBirthDay.month = 13;
myBirthDay.year = 0;
```

Oleh karena itu kita perlu melakukan enkapsulasi



Implementasi Enkapsulasi

1. Penyembunyian Informasi
2. Membuat method untuk akses data
3. Tambahkan pengamanan

Verify data

| Date | |
|-----------------------|------------|
| - day: int | day: int |
| - month: int | month: int |
| - year: int | year: int |
| + setDay(day:int) | |
| + getDay(): int | |
| + setMonth(month:int) | |
| + getMonth(): int | |
| + setYear(year:int) | |
| + getYear(): int | |



Apakah enkapsulasi harus 3 tahap?

- Tidak harus semua tahap dilakukan
- Jika data benar-benar tidak perlu diakses class lain maka cukup disembunyikan saja (tahap 1)
- Jika data perlu diakses oleh class lain namun dengan aturan khusus maka implementasikan tahap 1-3
- Jika data perlu diakses oleh class lain namun tidak ada aturan khusus untuk akses data itu, maka tidak perlu dilakukan enkapsulasi.



The Access Modifiers

- Untuk implementasi enkapsulasi kita memanfaatkan modifier.
- Untuk mengatur hak akses terhadap class feature.
- Suatu feature dalam class minimal harus memiliki satu modifier.
- Yang termasuk class feature:
 - The class itself
 - Its member variables
 - Its methods and constructors



The Access Modifiers

- Misal terdapat dua class yaitu: Mahasiswa dan TestMahasiswa;
- Class TestMahasiswa mempunyai akses ke class Mahasiswa.
- Artinya → class TestMahasiswa tersebut mampu:
 - Membuat instance dari class Mahasiswa.
 - Melakukan extend terhadap class Mahasiswa.
 - Mengakses methods dan variabel yg ada dalam class Mahasiswa.



Declaration

- Deklarasi class
`<modifier> class <identifier>{ ... }`
- Deklarasi Attribute
`<modifier> <type> <identifier>;`
- Deklarasi Method
`<modifier> <return type> <identifier>(...){...}`



private

- Digunakan oleh:
 - variabel
 - Method
 - Inner Class (khusus java)
- Variabel dan method yang dideklarasikan private hanya bisa diakses oleh instance dari class yg mendeklarasikan variabel dan method tersebut.
- Private variabel dan method dari class Xxx hanya bisa diakses melalui (within) class Xxx.
- Instance dari subclass tidak bisa mengakses private variabel dan method.



Hak akses modifier

| Modifier | Class yang sama | Package yang sama | Subclass | Class Manapun |
|-----------|-----------------|-------------------|----------|---------------|
| private | ✓ | | | |
| default | ✓ | ✓ | | |
| protected | ✓ | ✓ | ✓ | |
| public | ✓ | ✓ | ✓ | ✓ |



The import Statement

- Basic syntax of the import statement:

```
import <pkg_name>[.<sub_pkg_name>].<class_name>;
OR
import <pkg_name>[.<sub_pkg_name>].*;
```

- Examples:

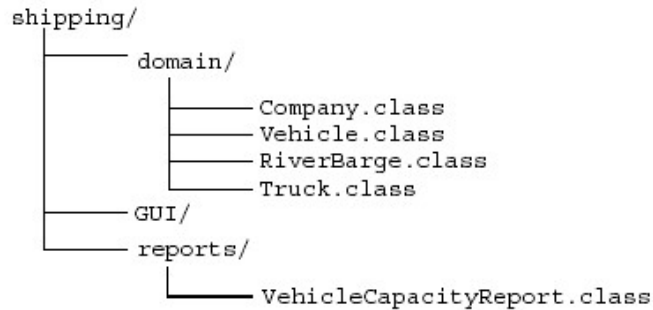
```
import shipping.domain.*;
import java.util.List;
import java.io.*;
```

- Precedes all class declarations
- Tells the compiler where to find classes to use

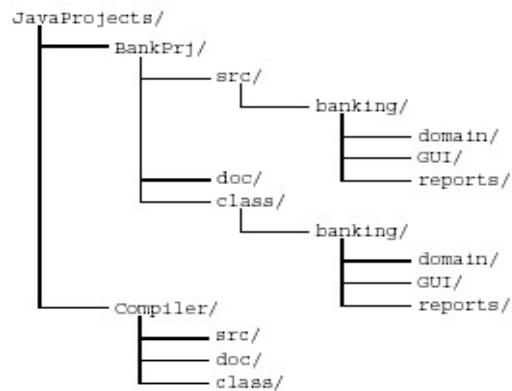


Directory Layout and Packages

- Packages are stored in the directory tree containing the package name.
- Example, the “shipping” application packages:



Development



Compiling using -d

```

cd JavaProjects/BankPrj/src / banking/domain
javac -d ../class/ banking/domain/*.java
  
```



Example1: Meangakses private variabel dari class lain

```

1. class Complex {
2.     private double real, imaginary;
3.
4.     public Complex(double r, double i) {
5.         real = r; imaginary = i;
6.     }
7.     public Complex add(Complex c) {
8.         return new Complex(real + c.real,
9.             imaginary + c.imaginary);
10.    }
11. }
12.
13. class Client {
14.     void useThem() {
15.         Complex c1 = new Complex(1, 2);
16.         Complex c2 = new Complex(3, 4);
17.         Complex c3 = c1.add(c2);
18.         double d = c3.real; // Illegal!
19.     }
20. }

```



Example2: Mengakses private variabel dari subclass.

```

1. class Complex {
2.     private double real, imaginary;
3. }
4.
5.
6. class SubComplex extends Complex {
7.     SubComplex(double r, double i) {
8.         real = r; // Trouble!
9.     }
10.}

```

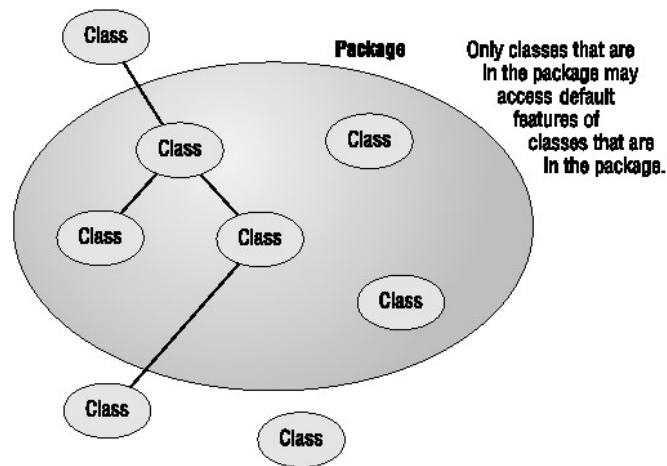


Default

- Mode akses untuk kelas, variabel, dan method jika mode akses tidak didefinisikan.
- Bukan merupakan Java keyword.
- Semua feature class-class yang ada dalam satu package bisa diakses oleh semua yang ada dalam package tersebut.
- Class diluar package boleh melakukan melakukan subclass, tetapi subclass tersebut tidak bisa mengakses feature superclass.



Default access



Example: default

```
1. package sportinggoods;
2. class Ski {
3.     void applyWax() { . . . } → default access
4. }
```

```
1. package sportinggoods;
2. class DownhillSki extends Ski {
3.     void tuneup() {
4.         applyWax();
5.         // other tuneup functionality here
6.     }
7. }
```



protected

- Protected mempunyai kemampuan akses yang lebih besar daripada private dan default.
- Protected digunakan pada:
 - Variabel
 - Method
- Protected feature dari suatu class bisa diakses oleh semua class dalam satu package.
- Class diluar package boleh melakukan melakukan subclass, dan subclass tersebut bisa mengakses feature superclass.



Example: protected

```

1. package adifferentpackage; // Class Ski now in
   // a different package
2. public class Ski {
3.     protected void applyWax() { . . . }
4. }

1.package sportinggoods;
2.import adifferentpackage.*;
2. class DownhillSki extends Ski {
3.     void tuneup() {
4.         applyWax();
5.         // other tuneup functionality here
6.     }
7. }

```



Summary of Access Modes

- **public**: A public feature may be accessed by any class.
- **protected**: A protected feature may only be accessed by a subclass of the class that owns the feature, or by a member of the same package as the class that owns the feature, or by a member of the different package as the class that owns the feature
- **default**: A default feature may only be accessed by a class from the same package as the class that owns the feature.
- **private**: A private feature may only be accessed by the class that owns the feature.



Summary of Access Modes to Class Members

| Visibility | Public | Protected | Default | Private |
|--|--------|-----------|---------|---------|
| From the same class | Yes | Yes | Yes | Yes |
| From any class in the same package | Yes | Yes | Yes | No |
| From any class outside the package | Yes | No | No | No |
| From a subclass in the same package | Yes | Yes | Yes | No |
| From a subclass outside the same package | Yes | Yes | No | No |



Tugas

1. Apakah yang dimaksud dengan enkapsulasi?
2. Apakah yang dimaksud dengan constructor?
3. Apakah yang dimaksud dengan overloading constructor?



1. Oracle Java Documentation, The Java™ Tutorials, <https://docs.oracle.com/javase/tutorial/>, Copyright © 1995, Oracle 2015.
2. Tita Karlita, Yuliana Setrowati, Rizky Yuniar Hakkun, Pemrograman Berorientasi Obyek, PENS-2012
3. Sun Java Programming, Sun Educational Services, Student Guide, Sun Microsystems, 2001.
4. John R. Hubbard, Programming With Java, McGraw-Hill, ISBN: 0-07-142040-1, 2004.
5. Patrick Niemeyer, Jonathan Knudsen, Learning Java, O'reilly, CA, ISBN: 1565927184, 2000.
6. Philip Heller, Simon Roberts, Complete Java 2 Certification Study Guide, Third Edition, Sybex, San Francisco, London, ISBN: 0-7821-4419-5, 2002.
7. Herbert Schildt, The Complete Reference, Java™ Seventh Edition, Mc Graw Hill, Osborne, ISBN: 978-0-07-163177-8, 2007