

Packages, Import dan Interfaces

1. Packages

Packages dalam JAVA berarti pengelompokan beberapa *class* dan *interface* dalam satu unit. Fitur ini menyediakan mekanisme untuk mengatur *class* dan *interface* dalam jumlah banyak dan menghindari konflik pada penamaan.

Mengimport Packages

Supaya dapat menggunakan *class* yang berada diluar *package* yang sedang dikerjakan, Anda harus mengimport *package* dimana *class* tersebut berada. Pada dasarnya, seluruh program JAVA mengimport *package java.lang.**, sehingga Anda dapat menggunakan *class* seperti String dan Integer dalam program meskipun belum mengimport *package* sama sekali.

Penulisan import *package* dapat dilakukan seperti dibawah ini :

```
import <namaPaket>;
```

Sebagai contoh, bila Anda ingin menggunakan *class* Color dalam *package* awt, Anda harus menuliskan import *package* sebagai berikut :

```
import java.awt.Color;  
import java.awt.*;
```

Baris pertama menyatakan untuk mengimport *class* Color secara spesifik pada *package*, sedangkan baris kedua menyatakan mengimport seluruh *class* yang terkandung dalam *package java.awt*.

Cara lain dalam mengimport *package* adalah dengan menuliskan referensi *package* secara eksplisit. Hal ini dilakukan dengan menggunakan nama *package* untuk mendeklarasikan *object* sebuah *class* :

```
java.awt.Color color;
```

2. **Import**

Pada sebuah file .java, dibutuhkan referensi file-file mana saja yang menjadi referensi dari class-class, method-method, ataupun segala sesuatu yang digunakan dalam sebuah program java yang ditulis dalam sebuah file .java tersebut. aturan penulisan pada umumnya ditulis di bawah penulisan package, contohnya adalah sebagai berikut :

```
import java.io.RandomAccessFile;
import java.net.*;
```

3. **Interface**

Mendeklarasikan sebuah interface pada dasarnya mendeklarasikan sebuah class tetapi sebagai penggantinya menggunakan kata kunci *class*, kata kunci *interface* digunakan. Berikut *syntaxnya*.

```
<interfaceDeclaration> ::=
<modifier> interface <name> {
<attributeDeclaration>*
[<modifier> <returnType> <name>(<parameter>*);]*
}
```

Anggotanya adalah *public* ketika interface dideklarasikan *public*.

Petunjuk Penulisan Program:

Secara mutlak atribut adalah static dan final dan harus diinisialisasi dengan nilai konstanta. Seperti mendeklarasikan class teratas, acces modifier yang valid hanyalah public dan package(yakni jika tidak ada acces modifier mengawali kata kunci class).

Class mengimplementasikan sebuah interface yang telah ada dengan menggunakan kata kunci *implements*. Class ini dibuat untuk mengimplementasikan semua method interface. Sebuah class boleh mengimplementasikan lebih dari satu interface.

Contoh berikut menunjukkan bagaimana mendeklarasikan dan menggunakan sebuah interface.

```
interface MyInterface {
void iMethod();
}
class MyClass1 implements MyInterface {
public void iMethod() {
System.out.println("Interface method.");
}
void myMethod() {
System.out.println("Another method.");
}
}
class MyClass2 implements MyInterface {
public void iMethod() {
System.out.println("Another implementation.");
}
}
class InterfaceDemo {
public static void main(String args[]) {
MyClass1 mc1 = new MyClass1();
MyClass2 mc2 = new MyClass2();
mc1.iMethod();
mc1.myMethod();
mc2.iMethod();
}
}
```

Output Program:

```
C:\Documents and Settings\Administrator\My Documents>javac
InterfaceDemo.java
```

```
C:\Documents and Settings\Administrator\My Documents>java
InterfaceDemo
Interface method.
Another method.
Another implementation.
```