Java – Bagian Spesifik

by: Yohanes Nugroho
rev: Achmad Imam Kistijantoro,
Saiful Akbar,
Yani Widyani

Kelas Akar

- Kelas akar (root class) adalah kelas yang menjadi nenek moyang (ancestor) semua objek
- Di Java, kelas akar adalah Object (`java.lang.Object`)
- Kelas yang tidak diturunkan dari apapun berarti diturunkan dari kelas yang bernama Object
  - Implikasinya: semua kelas adalah turunan dari Object
- Object memiliki beberapa method dasar seperti: `toString()`, `clone()`, dan method untuk sinkronisasi

Method String toString()

- Override method `toString()` untuk mencetak objek dengan lebih baik
- Misal, untuk kelas `Point`, isi methodnya:

```
String toString() {
   return "[" + x + ", " + y + "]";
}
```

- Dengan method di atas, Point bisa dicetak dengan mudah:

```
Point p = new Point(1,2);
System.out.println(p); /*mencetak point*/
```

- output potongan kode di atas: `[1, 2]`

final

- Sesuatu yang final tidak bisa diubah
  - Pada member variable berarti **konstanta**
  - Pada member function berarti tidak bisa diubah di turunannya
- Final member

```
final int pi=3.14;
```

- Final function

```
final void hello() {
}
```

  - Error jika di-redefinisi di turunannya

`finalize`

- Dalam kondisi tertentu ada resource yang tidak bisa direlease oleh Java
  - misal: kode yang memanggil kode native
  - resource ini harus direlease secara manual
- Kita bisa membuat Method `finalize` yang akan dieksekusi sebelum garbage collector menghancurkan Objek tersebut
  - biasanya yang dilakukan adalah membebaskan resource

Package

Catatan : bandingkan dengan namespace di CPP

Mengkompilasi kelas dalam Package

- Dari direktori di c:\source

```
javac image\ImageGIF.java
javac image\ImageJPEG.java
```

- Lalu bagaimana memakai kelas dalam package tertentu?
  - Kita perlu menyebutkan dengan lengkap nama package dan kelas atau
  - Kita perlu mengimpor package atau kelas tersebut

Memakai ImageGIF dengan Import

- Instruksi import digunakan agar nama kelas pada suatu package dikenali tanpa nama lengkapnya (cukup nama kelasnya), contoh:

```
import image.ImageGIF;
class TestImage {
   void hello() {
      ImageGIF a = new ImageGIF();
   }
}
```

Instruksi import

- import namapackage.NamaKelas;
  - harus satu per satu nama kelas disebutkan
- atau import namapackage.*;
  - semua Kelas dalam package tersebut diimport
  - lebih singkat menuliskannya
  - kompilasi lebih lama (semua nama kelas dicek)

Mengkompilasi TestImage

- Pindah ke drive C
  - ketik `c:`
- Pindah ke direktori source
  - ketik: `cd \source`
- Kompilasi seperti biasa
  - ketik: `javac TestImage.java`

Hierarki Package

- Package bisa bertingkat
  - misalnya kita ingin membuat package `SMSserver`
  - di dalam package smsserver ada package gsm dan cdma, masing-masing memiliki kelas `SMS`
  - boleh ada 2 kelas bernama sama di package berbeda
- cara membuat direktori:

```
mkdir c:\SMSserver
mkdir c:\SMSserver\gsm
mkdir c:\SMSserver\cdma
```

Menempatkan file

- File `SMS.java` untuk package cdma diletakkan di direktori `c:\SMSserver\cdma`
  - header file `SMS.java` berisi:

```
package SMSserver.cdma;
```

- File `SMS.java` untuk package gsm diletakkan di direktori `c:\SMSserver\gsm`
  - header file SMS.java berisi:

```
package SMSserver.gsm;
```

Mengkompilasi Isi Package

- Kompilasi dilakukan seperti biasa (dari c:\)

```
javac c:\smsserver\cdma\*.java
javac c:\smsserver\gsm\*.java
```

Mengimpor package dalam hierarki

- Import dengan nama package (yang hierarkinya nama packagenya dipisah dengan titik)

```
import smsserver.gsm.SMS;
```

- Hirerarki bisa bertingkat sebanyak mungkin
- Jika dalam package smsserver ada file `MainServer.java`, maka
  - import smsserver.*;
  - hanya akan mengimpor semua kelas dalam package smsserver tapi tidak mengimpor kelas dalam subpackage gsm dan cdma

Penamaan Package

- Semua karakternya memakai huruf kecil
- Sesuai penamaan domain, tapi terbalik, misalnya package XML milik lab programming ITB:

```
id.ac.itb.informatika.programming.
  xml
```

- nama id.ac.itb.if.programming tidak bisa dipakai, karena if adalah keyword di Java

Pemaketan package dalam file JAR

- Selain diletakkan dalam direktori, file kelas bisa dimasukkan dalam file JAR
- Contoh pembuatan file JAR:

```
jar –cf smsserver.jar c:\
```

- File JAR harus dimasukkan ke classpath agar dapat dipakai

Class path

- Classpath adalah lokasi (*path*) di mana Java akan mencari file class
- Default classpath java adalah . (titik) yang berarti direktori saat ini, dan file JAR milik sistem (bawaan Java)
- Classpath bisa diubah

```
export CLASSPATH=/usr/test.jar:. (Linux)
set CLASSPATH=c:\test.jar;. (Windows)
```

Contoh Pemakaian Classpath

- Masukkan seluruh direktori smsserver atau direktori image dalam contoh sebelumnya ke c:\library
- set classpath menjadi c:\library dan direktori saat ini:

```
set CLASSPATH=c:\library;.
```

- Setelah classpath diset, maka file yang memakai kelas dalam package boleh berada di mana saja

Contoh: isi kelas dalam JAR

- Masukkan file jar ke c:\library
- set classpath menjadi:

```
set
  CLASSPATH=c:\library\smsserve
  r.jar;c:\library\image.jar;.
```

- File yang memakai kelas dalam package boleh berada di mana saja

Menjalankan Program dalam Package

- Set classpath, lalu:

```
java
  namapackage.subpackage.KelasX
```

- Atau set classpath untuk saat ini saja:

```
java –cp test.jar;.
  namapackage.subpackage.KelasX
```