Materi Pelatihan

Java Fundamental

Endy Muhardin
endy@artivisi.com
http://endy.artivisi.com

http://www.artivisi.com

Jakarta, 2008

Daftar Isi

1.Pendahuluan

Tentang Java

Teknologi Java telah berusia 10 tahun lebih. Dalam usia tersebut, suatu teknologi dapat dikatakan telah matang. Populasi penggunanya relatif besar, sehingga apabila kita menemui masalah, banyak referensi dan tutorial tentang cara mengatasinya.

Sepanjang perjalanannya, Java telah mengalami dua kali perubahan signifikan. Pertama pada waktu transisi dari versi 1.3 menjadi 1.4, dan kemudian 1.4 menjadi 1.5 atau dikenal juga dengan 5.0. Dalam transisi tersebut, ada perubahan signifikan dalam teknik pemrograman maupun library yang disediakan. Transisi ini berjalan relatif mulus, menunjukkan bahwa teknologi Java cukup solid dan dibangun di atas dasar yang kokoh. Semua transisi mempertahankan *binary backward compatibility*, artinya semua aplikasi yang sudah kita kompilasi dan berjalan mulus di Java versi lama akan terus berjalan dengan baik walaupun kita melakukan upgrade.

Bagaimana cara belajar Java

Seiring dengan pertambahan umurnya, fitur dalam Java juga sudah menjadi sangat besar. Mulai dari pemrograman perangkat kecil seperti handphone dan PDA, sampai ke aplikasi enterprise yang berinteraksi dengan mainframe dan mesin-mesin server besar.

Ukuran besar ini di satu sisi menguntungkan karena apapun yang kita butuhkan, Java sudah menyediakannya. Tapi di lain pihak, ini menyulitkan orang yang baru mulai belajar Java. Banyak sekali fitur yang saling berkaitan dan berhubungan, sehingga para pemula

mengalami kesulitan untuk menemukan titik awal untuk belajar.

Berdasarkan pengalaman saya belajar Java secara otodidak, jalur yang optimal adalah sebagai berikut:

1. Belajar sintaks Java
2. Belajar konsep OOP
3. Belajar dasar pemrograman database (JDBC)
4. Belajar dasar pemrograman web (Servlet)
5. Belajar dasar pemrograman desktop (Swing)

Setelah menguasai lima materi dasar tersebut, kita akan memiliki modal yang mencukupi untuk menyelam lebih jauh lagi menuju lingkup Java yang kita minati, misalnya Java Mobile Edition (JME) atau Java Enterprise Edition (JEE).

Tentang pelatihan ini

Pelatihan ini bertujuan untuk menyediakan pemahaman tentang sintaks Java dan konsep pemrograman berorientasi object (Object Oriented Programming – OOP).

Setelah mengikuti pelatihan ini, peserta mungkin belum bisa membuat aplikasi yang utuh. Tapi dasar-dasar untuk mendesain aplikasi dengan paradigma object sudah dimiliki.

Pemahaman OOP relatif sulit diserap dan dijiwai oleh programmer yang sebelumnya berasal dari latar belakang prosedural. Tetapi jangan khawatir. Pada bagian desain aplikasi, kita akan melihat contoh kasus, kemudian mendesainnya menggunakan kedua pendekatan (OOP dan Prosedural). Dengan demikian diharapkan pembaca dapat memahami perbedaan antara keduanya.

Pendekatan OOP bukanlah dimaksudkan untuk menggantikan

100% paradigma prosedural. Masing-masing pendekatan ada tempatnya. Asal digunakan dengan proporsi yang sesuai, mencampur kedua pendekatan bukanlah hal yang tabu.

Susunan materi pelatihan

Konsep OOP adalah konsep yang umum. Konsep ini bisa ditemui di berbagai bahasa pemrograman. Bahkan kita juga bisa mengimplementasikannya di bahasa yang tidak mendukung OOP secara eksplisit.

Walaupun demikian, materi pelatihan ini ditujukan untuk mempelajari OOP dengan Java. Jadi ada banyak perpaduan antara konsep yang spesifik hanya ada di Java, dan konsep yang umum dan ada di bahasa yang lainnya. Bila perbedaannya cukup jelas, saya akan menuliskannya di catatan tepi.

Berikut susunan materi pelatihan ini:

1. Pendahuluan
2. Persiapan. Berisi gambaran umum tentang cara membuat aplikasi Java.
3. Pemrograman Java. Berisi penjelasan singkat tentang bahasa Java. Di sini pembaca diasumsikan sudah pernah melakukan pemrograman dengan teknologi lainnya. Sehingga yang dibahas di sini adalah bagaimana melakukan perulangan (looping), bukan konsep perulangan itu sendiri.
4. Konsep Object Oriented Programming dengan Java. Setelah menguasai bahasa pemrograman Java, pada bagian ini pembaca akan melihat bagaimana konsep Object Oriented Programming diterapkan dalam Java.

Latar belakang pembaca

Pembaca diasumsikan sudah mengenal pemrograman sebelumnya. Istilah dan konsep umum pemrograman seperti misalnya:

- percabangan (if/else, switch)
- perulangan (for, while)
- function atau method

seharusnya sudah dipahami.

Aturan penulisan

Kode program ditulis dengan font Courier tebal seperti ini:

```
public class Shipment {
}
```

penting

hal yang penting dan perlu diperhatikan ditulis di tepi seperti ini.

Kadang-kadang kotak ini juga digunakan untuk menjelaskan fitur yang hanya ada di Java versi tertentu

Perintah command prompt ditulis dengan latar belakang kelabu seperti ini:

```
C:\> javac -version
```

Sedangkan output dari hasil eksekusi perintah tersebut ditulis tanpa latar belakang dengan font Courier biasa (tidak bold) seperti ini:

```
javac 1.6.0
```

2.Persiapan

Java dan berbagai variasinya

Kalau kita membuka website resmi Java http://java.sun.com, kita akan mendapati banyak sekali link. Pemula Java biasanya akan pusing melihat banyak sekali versi Java yang tidak mereka mengerti.

Teknologi Java dikelompokkan menjadi tiga kategori utama:

- Java Standard Edition
- Java Mobile Edition
- Java Enterprise Edition

Java Standard Edition

Pada sebagian besar kasus, kita hanya membutuhkan Java Standard Edition. Ini merupakan perlengkapan minimal untuk mulai membuat aplikasi Java. Java Standard Edition Software Development Kit (SDK) berisi beberapa peralatan penting. Berikut adalah peralatan yang paling sering kita gunakan:

- Compiler (javac) : tool untuk mengubah source code menjadi byte code yang siap dieksekusi.
- Runtime (java) : tool untuk menjalankan byte code
- JavaDoc Compiler (javadoc) : Generator untuk dokumentasi source code (JavaDoc)
- Archiver (jar) : tool untuk membuat paket java (*.jar)

Java Enterprise Edition

Ini adalah kumpulan paket untuk membuat aplikasi enterprise. Tetapi kita tidak perlu mendownload paket ini. Biasanya paket yang ada di sini telah disediakan oleh development tool kita ataupun application server yang digunakan.

Paket ini ada di website Java sebagai pusat referensi.

Java Mobile Edition

Jelajahi bagian ini jika Anda ingin membuat aplikasi yang berjalan di perangkat kecil seperti handphone atau PDA.

Instalasi Java

Download

Ada beberapa file yang perlu di download:

- SDK Installer. Gunakan saja versi terbaru (saat ini 6.0). Java SDK ini sudah berisi Runtime Environment (RE).
- Dokumentasi Library, biasa dikenal dengan JavaDoc API
- Tutorial resmi.

SDK dan Dokumentasi dapat didownload pada alamat ini:

http://java.sun.com/javase/downloads/index.jsp

Ada banyak link download di sana, tapi jangan bingung. Ambil **JDK 6** dan **Java SE 6 Documentation**. Jangan serakah dan mendownload JDK 6 with Java EE atau JDK 6 with Netbeans 5.5. Seperti sudah dijelaskan di atas, Java EE dapat didapatkan berikut application server yang kita gunakan. Netbeans juga lebih mudah diinstal secara terpisah.

Java Runtime Environment (JRE) 6 juga tidak perlu didownload karena sudah ada di dalam SDK. JRE hanya dibutuhkan bila kita

hanya ingin menjalankan aplikasi Java dan tidak akan membuat atau memodifikasi aplikasi Java.

Tutorial resmi dapat dilihat secara online di sini:

http://java.sun.com/docs/books/tutorial/index.html

Pada halaman muka tutorial, ada link untuk mendownload tutorial tersebut berikut contoh kode yang dibahas di dalamnya.

Instalasi

Instalasi Java sangat mudah. Cukup jalankan file installernya. Untuk Windows, klik dua kali installernya, kemudian klik Next beberapa kali.

Untuk Linux, ganti permission filenya supaya menjadi executable

```
$ chmod +x  jdk-6-linux-i586.bin
```

Kemudian jalankan installernya. Saya biasanya menginstal di folder /opt. Untuk itu, butuh akses root.

```
$ sudo ./ jdk-6-linux-i586.bin
```

Setelah instalasi selesai dilakukan,tes dengan cara menjalankan Java runtime :

```
C:\> java -version
java version "1.5.0_06"
Java(TM) 2 Runtime Environment, Standard Edition (build
1.5.0_06-b05)
Java HotSpot(TM) Client VM (build 1.5.0_06-b05, mixed
mode, sharing)
```

Konfigurasi variabel

Ada satu hal lagi yang perlu kita lakukan sebelum Java bisa digunakan untuk memproses kode program, yaitu setting variabel PATH.

Sebelum disetting, kita tidak bisa memanggil perintah `javac`. Perintah ini berguna untuk melakukan kompilasi.

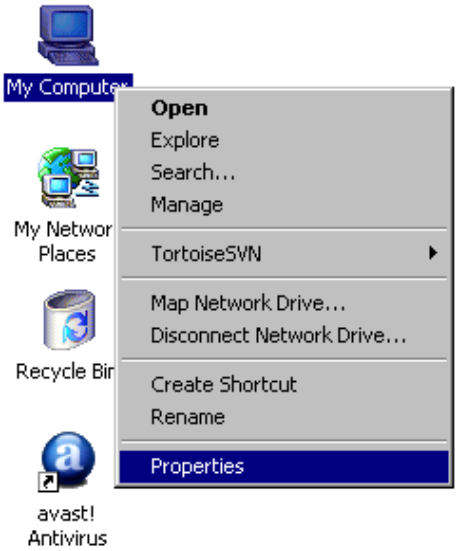
```
C:\> javac -version
'javac' is not recognized as an internal or external
command, operable program, or batch file
```

Perintah `javac` (`javac.exe` untuk Windows) berada di folder `[LOKASI_INSTALASI_JAVA_SDK]\bin`. Kita harus mendaftarkan folder ini ke dalam PATH.

Sambil mengedit variabel PATH, kita juga membuat variabel baru JAVA_HOME. Variabel ini sering dibutuhkan oleh aplikasi Java. Sudah menjadi kesepakatan umum bahwa variabel ini harus menunjuk ke lokasi instalasi Java SDK atau Java RE.

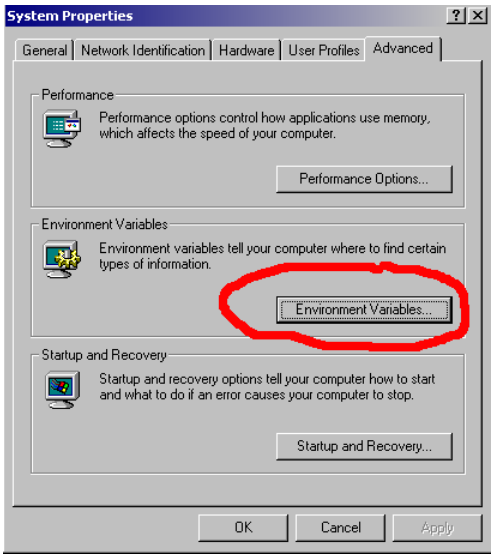
Cara setting variabel di Windows adalah dengan membuka System Properties.

Klik kanan icon My Computer

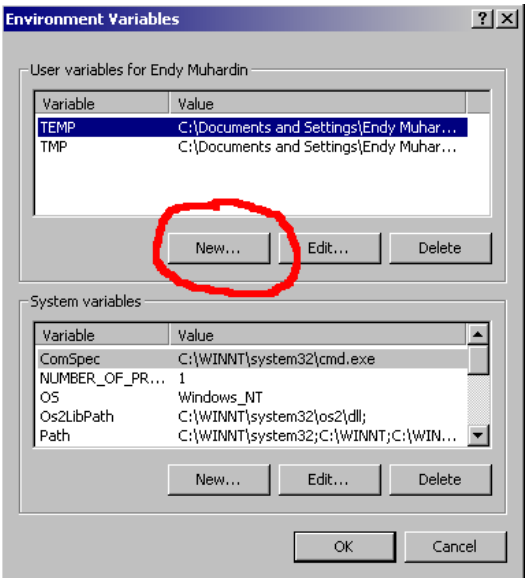


System Properties akan terbuka. Klik Advanced, dan klik

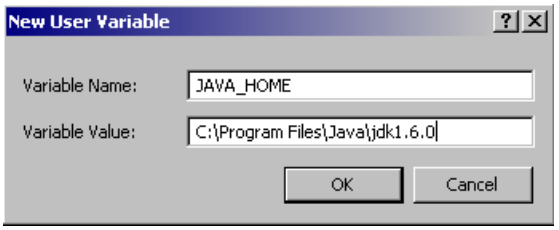
Environment Variables.



Pada layar Environment Variables, klik New untuk membuat variabel baru.



Isikan `JAVA_HOME` pada field Name, dan lokasi instalasi Java (misalnya `C:\Program Files\Java\jdk1.6.0`) pada field Value.



Setelah itu, klik OK.

Kemudian buat User Variable sekali lagi, kali ini Variable Name diisi dengan PATH, dan Variable Value diisi dengan

```
%PATH%;%JAVA_HOME%\bin
```

Sekarang coba buka Command Prompt, dan coba lagi jalankan `javac`. Seharusnya sekarang `javac` sudah dikenali.

```
C:\> javac -version
javac 1.6.0
```

Untuk Linux, variabel PATH untuk masing-masing user diatur melalui file konfigurasi yang terletak di `/home/namauser/.bash_profile`. Kita harus mengedit file ini agar variabel `JAVA_HOME` terdaftar di sistem.

Agar perubahan di file ini tidak terlalu banyak, buat satu file lagi yang berisi konfigurasi path, misalnya namanya `javapath.sh`.

```
JAVA_HOME=/opt/jdk1.6.0
export JAVA_HOME
export PATH=$PATH:$JAVA_HOME/bin
```

Kemudian panggil file javapath.sh ini dari dalam bash_profile. Tambahkan satu baris berikut di ujung akhir bash_profile.

```
. ~/javapath.sh
```

Sekarang Java SDK siap digunakan.

Hello World

Untuk mengetes hasil instalasi kita, mari kita buat kode program sederhana. Gunakan text editor yang tersedia, misalnya Notepad.

Source Code

Ketik kode program berikut.

```
public class Halo {
  public static void main(String[] args){
    System.out.println(“Halo”);
  }
}
```

kemudian simpan dengan nama `Halo.java`. Perhatikan besar kecilnya huruf. Kode program Java *case sensitive*, artinya membedakan antara huruf besar dan huruf kecil.

Kompilasi

File `Halo.java` tersebut harus dikompilasi dulu sebelum bisa dijalankan. Buka command prompt, dan masuk ke folder di mana `Halo.java` berada.

Jalankan perintah `javac`.

```
C:\Latihan> javac Halo.java
```

Perintah di atas akan menghasilkan file hasil kompilasi, yaitu `Halo.class`.

Eksekusi

Kalau tidak ada error pada saat kompilasi, kode program kita sudah bisa dijalankan. Gunakan perintah `java` untuk menjalankannya.

```
C:\Latihan> java Halo
Halo
```

Perhatikan bahwa perintah `java` menerima argumen berupa nama kelas, **bukan nama file**.

Kalau perintah di atas berhasil dijalankan, berarti instalasi Java kia sudah berhasil dengan baik.

Konsep penting

Dalam dunia Java, ada beberapa istilah penting yang harus dipahami sebelum kita melangkah lebih jauh.

Bytecode

Kode program kita, file berekstensi *.java, harus dikompilasi terlebih dulu sebelum bisa dijalankan. Ini berbeda dengan bahasa pemrograman yang berbasis interpreter seperti PHP, Python, atau Ruby yang dapat langsung dijalankan dari source code.

Source code Java dikompilasi menjadi bytecode, yaitu file yang berekstensi *.class. JRE mengerti dan bisa menjalankan bytecode ini.

Bytecode bersifat universal dan dapat berjalan di berbagai sistem operasi. Ini berbeda dengan file kompilasi C atau C++ yang hanya bisa berjalan di satu sistem operasi tertentu. Untuk bisa menjalankan kode program C di sistem operasi yang lain, kita harus melakukan kompilasi sesuai dengan sistem operasi yang kita tuju. Di Java, kita hanya perlu mengkompilasi satu kali, bytecode dapat dijalankan di semua sistem operasi.

Virtual Machine

Portabilitas bytecode dimungkinkan dengan adanya Virtual Machine (VM). Java VM biasa disebut juga Java Runtime

Environment (JRE). Java VM ini berbeda antar sistem operasi. Pastikan kita mendownload JRE yang sesuai dengan sistem operasi yang kita gunakan.

Dengan arsitektur VM, bytecode tidak langsung dieksekusi oleh sistem operasi, melainkan oleh VM. Ini tentu saja menghasilkan *overhead* dibandingkan kode program yang langsung dieksekusi sistem operasi seperti C. Kita tentu mengira Java lebih lambat daripada aplikasi yang berjalan tanpa VM (*native*). Ini benar, tapi perbedaannya tidak sebesar yang kita asumsikan. Java VM telah berusia sangat tua, sehingga sudah banyak optimasi yang dilakukan. Overhead yang ditimbulkan telah berhasil diminimasi sehingga tidak terlalu terasa. Keuntungan portabilitas kode menjadi lebih besar daripada overhead penggunaan VM.

Pada waktu pertama kali dijalankan, Java VM langsung membooking memori sebesar nilai defaultnya, biasanya 32 MB. Jumlah ini dipertahankan sampai aplikasi kita butuh memori lebih. Bila itu terjadi, VM akan membesarkan memorinya. Kegiatan memperbesar memori ini butuh waktu agak lama, sehingga aplikasi yang kita jalankan terkesan berjalan tersendat-sendat. Untuk mencegah hal ini, berikan memori awal yang cukup agar VM tidak perlu memperbesar memori di tengah jalan.

Classpath

Misalnya kita membuat kode program sendiri. Pertanyaan yang penting adalah, bagaimana Java VM bisa menemukan `*.class` yang kita buat? Sebab, kita bisa saja meletakkannya di sembarang tempat.

Jawabannya adalah, Java VM akan mencari di dalam *classpath*. Konsep *classpath* sebenarnya tidak spesifik Java. Di PHP dikenal istilah *include_path*, Python menyebutnya *sys.path*, dan dalam Ruby dikenal istilah *RUBYLIB*. Intinya, ini adalah pemberitahuan kita

kepada VM di mana harus mencari kode program yang dibutuhkan.

Kita dapat mengatur variabel classpath melalui Environment Variables di System Properties. Tapi cara ini tidak direkomendasikan, karena setting classpath untuk masing-masing aplikasi biasanya berbeda. Kita tidak ingin setting classpath bersifat permanen dan mengganggu aplikasi lainnya.

Cara yang lebih baik adalah melakukan setting classpath pada saat dibutuhkan, melalui command prompt. Kekurangan cara ini adalah, setting tersebut akan hilang begitu command prompt ditutup. Jika kita membuka dua command prompt, kita harus melakukan setting di kedua command prompt tersebut.

Cara melakukan setting melalui command prompt adalah sebagai berikut:

```
C:\> set CLASSPATH=C:\Latihan;D:\belajar-java
```

Kita bisa memasukkan banyak folder ke dalam classpath, masing-masing folder dipisahkan dengan titik koma (;) untuk Windows, dan titik dua (:) untuk Linux.

Setelah classpath kita set ke folder berisi `Halo.class`, kita bisa memanggil class tersebut dari mana saja, tidak harus di folder tempat file tersebut berada.

Referensi

Sebenarnya untuk mempelajari Java secara mandiri (otodidak), referensi yang disediakan oleh Sun Microsystem sudah memadai. Ada dua referensi utama yang wajib dimiliki: Java API Documentation dan Java Tutorial.

Java Tutorial

Java Tutorial dapat diakses di

http://java.sun.com/docs/books/tutorial/index.html

Isinya sangat lengkap dan komprehensif. Sayangnya ditulis dalam bahasa Inggris, yang biasanya menyebabkan orang enggan membacanya.

Java API

Java API Documentation berisi dokumentasi semua class yang disertakan dalam distribusi Java SDK. Kalau kita menggunakan text editor yang canggih (seperti Eclipse, Netbeans, atau Intellij IDEA), dokumentasi ini sudah diikutkan secara built-in. Tetapi apabila kita menggunakan editor biasa seperti Notepad++, dokumentasi ini sangat bermanfaat.

Siklus pemrograman Java

Sebelum kita mulai menulis kode, kita lihat dulu secara garis besar bagaimana cara membuat aplikasi dalam Java.

Desain

Aplikasi Java didesain dengan pola pikir berorientasi object (Object Oriented). Biasanya desain aplikasi ini dituliskan dengan notasi Unified Modelling Language (UML). Banyak UML editor yang tersedia secara open source, misalnya:

- Poseidon CE
- Visual Paradigm
- Jude Bamboo
- Omondo (Eclipse Plugin)

Beberapa UML editor memiliki kemampuan untuk menghasilkan kerangka kode Java. Beberapa diantaranya bahkan dapat bekerja sebaliknya, menghasilkan diagram UML dari kode program Java yang sudah ada.

Tulis source code

Langkah berikutnya tentunya adalah menulis source code. Ada banyak peralatan pembantu yang dapat mempercepat dan mempermudah penulisan kode.

Editor yang cukup terkenal dan open source diantaranya adalah Eclipse dan Netbeans. Eclipse memiliki kemampuan pengeditan source code yang sangat baik. Sedangkan Netbeans memiliki UI Designer untuk menggambar tampilan aplikasi desktop yang sangat canggih.

Kita tidak harus memilih salah satu saja. Untuk hasil yang maksimal, kita dapat menggunakan Eclipse untuk menulis kode *business logic*, dan menggunakan Netbeans untuk membuat tampilan User Interface.

Selain kedua editor yang open source itu, ada juga editor komersial yang sangat terkenal, IntelliJ IDEA. Dengan semboyannya, “Enjoy Development”, IDEA dikenal sangat nyaman digunakan dan mampu meningkatkan produktivitas programmer.

Kompilasi

Source code yang sudah kita tulis harus dikompilasi sebelum bisa dijalankan. Untuk source code yang berjumlah satu-dua file, kita dapat menggunakan `javac` untuk mengkompilasinya.

Aplikasi Java umumnya memiliki source code yang terdiri dari antara puluhan sampai ribuan file. Kemudian masih ada ketergantungan dengan library tambahan. Dengan skala ini, tidak praktis kalau kita melakukan kompilasi dengan javac. Untuk itu, kita gunakan perangkat `build` yang disebut Ant.

Ant bisa didapatkan di http://ant.apache.org. Selain melakukan kompilasi, Ant memiliki banyak fitur lainnya, seperti menjalankan test, mempaket aplikasi, mengendalikan application server, dan

masih banyak yang lainnya.

Ant adalah perangkat yang wajib dikuasai oleh programmer Java.

Test

Platform Java menyediakan perangkat test yang lengkap. Berikut adalah berbagai jenis test yang dapat dilakukan dalam pembangunan aplikasi Java.

Jenis	***Lingkup***	***Keterangan***
Unit Test	Method	Unit test mengetes satu method dalam class. Method yang kita tulis diuji terhadap berbagai input dan hasilnya diperiksa.
Integration Test	Titik sambungan	Pada test ini, hubungan aplikasi dengan pihak eksternal diujicoba. Misalnya koneksi ke database, koneksi ke email server, atau sambungan soket ke aplikasi lain.
In Container Test	Dalam Server	Kadangkala ada fungsi yang hanya bisa dites ketika aplikasi sedang berjalan di dalam server. Tes ini bertujuan untuk mengotomasi pelaksanaan tes tersebut.
Code Review	Source Code	Dalam ketergesaan, kadang kita meninggalkan praktek pemrograman yang baik dan benar. Perangkat code review mencari dan melaporkan kode program yang ditulis secara ceroboh dan berpotensi menimbulkan bug.
Coverage Test	Aplikasi	Kita ingin mengetahui apakah rangkaian test yang kita miliki sudah mengetes seluruh bagian dari kode program kita. Dengan

Jenis	*Lingkup*	*Keterangan*
		perangkat coverage test, kita dapat mengetahui bagian mana dari aplikasi kita yang belum tersentuh tes sama sekali.

Cara kerja perlengkapan testing adalah sebagai berikut:

1. Tulis kode tes untuk mengetes kode program
2. Jalankan kode test
3. Periksa laporan hasil tes
4. Perbaiki kode program bila ditemukan kegagalan tes
5. Tambah lagi tes sehingga lebih lengkap

Penggunaan peralatan testing akan meningkatkan kualitas kode kita. Perlengkapan testing yang baik akan terus menjalankan kode tes secara otomatis.

Jadi, aplikasi kita akan selalu dites, bahkan kode yang sudah lama tidak kita sentuh. Dengan demikian, apabila penambahan kode baru menyebabkan error di kode lama, kita akan segera mengetahuinya. Ini akan meningkatkan stabilitas kualitas kode program yang kita hasilkan.

Paket

Aplikasi yang sudah selesai akan dipaket sehingga mudah dideploy di application server. Atau kalau aplikasinya adalah aplikasi desktop, kita perlu membuatkan installer supaya pengguna hanya tinggal mengklik Next untuk menginstal aplikasi.

Biasanya langkah ini dilakukan di akhir siklus pembangunan aplikasi.

Deploy

Pada saat pembangunan aplikasi, kita akan sering melakukan

deploy-undeploy aplikasi kita di server. Untuk mempercepat siklus ini, kita dapat mengotomasi prosesnya dengan menggunakan Ant.

Distribusi

Aplikasi yang sudah selesai dibuat siap didistribusikan agar bisa digunakan khalayak ramai. Selain metode download atau bagi-bagi CD yang sudah umum dilakukan, Java juga menyediakan fasilitas Java Web Start. Dengan fasilitas ini, pengguna cukup mengklik satu link di website yang kita sediakan. Setelah itu, Java secara otomatis akan mendownload, menginstal, dan menjalankan aplikasi. Setelah itu, aplikasi akan disimpan di mesin lokal, sehingga ketika dijalankan lagi, pengguna tidak perlu mendownload lagi.

Aplikasi yang diiinstal melalui Java Web Start juga akan memeriksa website asalnya setiap kali dijalankan untuk mencari versi terbaru. Apabila ada versi yang lebih baru, dia akan secara otomatis mengupgrade dirinya sendiri.

Ini akan memudahkan kita untuk mendistribusikan dan memaintain aplikasi.

Perangkat pembantu

Untuk meningkatkan produktivitas dan mempercepat development, ada beberapa perangkat yang perlu kita ketahui.

Ant

Ant adalah build tool. Dapat diperoleh secara cuma-cuma di http://ant.apache.org. Ant memiliki kemampuan untuk:

- melakukan kompilasi
- menjalankan aplikasi
- mengkompres folder
- menjalankan tes dan membuat laporannya

- berinteraksi dengan aplikasi version control
- melakukan deploy/undeploy
- menyalakan/mematikan application server
- dan sebagainya

Dengan Ant, kita dapat mengotomasi pekerjaan yang sering kita lakukan sehingga dapat dijalankan dengan satu perintah saja.

Eclipse

Eclipse adalah Integrated Development Editor (IDE) open source yang disponsori IBM. Fiturnya sangat lengkap dan canggih. Eclipse dibangun dengan arsitektur berbasis plugin, sehingga banyak orang yang menyediakan pluginnya baik gratis maupun komersil.

Netbeans

Netbeans adalah IDE open source yang disponsori oleh Sun Microsystem. Engine Matisse yang dimilikinya sangat legendaris. Dengan Matisse, kita dapat membuat aplikasi desktop semudah Visual Basic atau Borland Delphi.

Hypersonic SQL

Hypersonic SQL adalah aplikasi database yang mungil. Ukurannya cuma 620 KB, tidak sampai satu megabyte. Hypersonic SQL dapat dijalankan sebagai server, dan juga bisa dijalankan secara *embedded* oleh aplikasi kita.

Dengan ukurannya yang ringan, kita dapat melakukan development dengan lebih cepat.

Tomcat

Tomcat adalah servlet engine, web server yang mengerti bagaimana menjalankan servet dan JSP. Tomcat tersedia secara gratis di http://tomcat.apache.org.

Jetty

Sama seperti Tomcat, Jetty adalah servlet dan JSP engine. Keistimewaan Jetty dibandingkan Tomcat adalah Jetty sangat kecil. Ukurannya hanya 420 KB. Bandingkan dengan Tomcat yang memakan harddisk sebesar 20 MB.

Sama seperti Hypersonic SQL, Jetty juga bisa dijalankan secara *embedded*. Kita bisa menjalankan Jetty melalui Ant.

3.Pemrograman Java

Setelah memahami secara umum tentang apa itu Java, sekarang kita akan menyelam ke dunia yang lebih teknis, bahasa pemrograman Java.

Bahasa pemrograman apapun yang kita pelajari, biasanya ada dua hal yang harus kita pahami:

- bahasa pemrograman
- library yang disediakan

Pada bab ini, kita akan mempelajari bahasa pemrograman Java. Yang dibahas di sini adalah bagaimana konsep umum pemrograman diterapkan di Java. Di antara konsep umum tersebut adalah:

- statement (instruksi)
- variabel dan tipe data
- percabangan (if-else, switch)
- perulangan (for, while)

Anatomi kode program

Untuk memahami bagaimana aplikasi Java tersusun, kita perlu memahami class, package, dan class member.

Class

Secara umum, sebuah aplikasi Java terdiri banyak class. Aplikasi skala kecil dapat memiliki puluhan class, sedangkan aplikasi enterprise bisa terdiri dari ribuan class.

Satu file bisa menampung beberapa class, tetapi umumnya hal ini

tidak dianjurkan. Biasanya untuk kasus normal, satu file memuat satu class saja.

Package

Class dikelompokkan dalam package. Adanya nama package menyediakan namespace. Ini bermanfaat agar kita bisa menggunakan nama class yang sama untuk kondisi yang berbeda. Misalnya, untuk tampilan web, kita memiliki class Session yang merepresentasikan kegiatan user yang sedang login. Di sisi lain, kita juga memiliki class Session untuk merepresentasikan koneksi yang kita lakukan ke database. Jadi ada Session untuk tampilan dan ada Session koneksi database.

Dalam urusan Session ini, kita dapat menempatkan kedua Session ke dalam dua package, package database dan package web. Deklarasi class menjadi sebagai berikut:

```
package com.artivisi.latihan.web;
public class Session {
}
```

dan

```
package com.artivisi.latihan.database;
public class Session {
}
```

Dengan demikian, kita dapat memiliki dua class bernama sama untuk keperluan yang berbeda.

Class Member

Sebuah class berisi banyak class member. Jenis-jenis class member adalah:

- Constructor
- Method

- Property
- Inner Class

Penjelasan lebih lanjut tentang masing-masing class member ini akan diberikan di bab selanjutnya.

Berbeda dengan bahasa pemrograman PHP atau Ruby, di Java semua kode program harus diletakkan pada tempatnya. Di PHP misalnya, kita bisa mendeklarasikan variabel atau membuat function di mana saja. Tetapi di Java, kita harus membuat method di dalam class.

Sebagai ilustrasi, dalam satu file `belajar.php`, isinya bisa seperti ini:

```
<?
class Person {
  var $name;
}
function display_person(){
  $p = new Person();
  $p=>name = “Endy”;
  echo($p=>name);
}
echo(“menjalankan function <br>”);
display_person();
?>
```

Pada kode di atas, kita lihat dalam satu file bisa berisi deklarasi class, deklarasi function, dan statement yang memanggil function.

Di Java, kode seperti di atas akan menimbulkan error kompilasi. Untuk melakukan hal yang sama (deklarasi class, deklarasi function, dan mengeksekusi function tersebut) di Java, kita menempatkan kode menurut aturan Java, seperti ini:

```
package belajar.java;
class Person {
  public String name;
}
public class Belajar {
  public static void displayPerson() {
    Person p = new Person();
    p.name = “Endy”;
    System.out.println(p.name);
  }
  public static void main(String[] xx) {
    displayPerson();
  }
}
```

Seperti kita lihat di atas, tidak ada statement yang ada di luar class. Semua deklarasi function (method) dan variabel dilakukan di dalam class.

Statement, Expression, Block

Statement

Statement dalam bahasa Indonesia dapat diartikan menjadi instruksi atau perintah. Setiap statement diakhiri dengan tanda titik koma (;). Ini berbeda dengan VB, dimana statement diakhiri dengan pergantian baris.

Contoh statement adalah sebagai berikut:

```
System.out.println(“Halo”);
```

Ekspresi

Ekspresi adalah rangkaian variabel atau pemanggilan method yang menghasilkan satu nilai. Ekspresi biasanya digunakan dalam

deklarasi variabel atau dalam percabangan.

Coba lihat contoh kode berikut:

```
int a = 5;
if (username.startsWith(“admin”)) {
  // lakukan sesuatu
}
```

Pada contoh kode di atas, yang disebut ekspresi adalah:

```
a = 5
```

dan

```
username.startsWith(“admin”)
```

Ekspresi pertama menghasilkan nilai integer, yaitu 5. Sedangkan ekspresi kedua menghasilkan nilai `true` atau `false`, tergantung apa isi variabel username.

Komentar

Komentar adalah tulisan dalam kode program yang diabaikan oleh compiler. Komentar digunakan untuk memperjelas kode program kita. Misalnya ada algoritma rumit yang panjangnya beberapa belas baris, kita menambahkan satu baris komentar di atasnya untuk menjelaskan kegunaan algoritma tersebut.

Dengan demikian, orang tidak harus membaca keseluruhan baris kode hanya untuk memahami bahwa kode tersebut (misalnya) melakukan formatting untuk tampilan report.

Penulisan komentar di Java sama dengan komentar di C, C++, atau PHP.

Komentar satu baris diawali dengan dua tanda garis miring seperti ini:

```
// apabila user adalah admin, langsung buka halaman admin
if(username.startsWith(“admin”)) {
  // redirect ke admin page
}
```

Komentar yang terdiri dari banyak baris diawali dengan /* dan diakhiri dengan */. Contohnya sebagai berikut:

```
/*
Menjelajahi hasil query dari database.
Bila menemukan pesanan yang bernilai tinggi, catat.
*/
while(rs.next()) {
  if(rs.getDouble(“price”) > 10000.0) {
    // tulis ke log file
  }
}
```

JavaDoc

Selain komentar biasa seperti di atas, di Java dikenal satu jenis komentar khusus yang dinamakan komentar JavaDoc. Berbeda dengan komentar biasa yang diabaikan kompiler, untuk komentar JavaDoc kita dapat menggunakan kompiler khusus untuk mengambil dan memformat isinya menjadi dokumentasi dalam bentuk HTML.

JavaDoc ini biasanya dibuat untuk mendokumentasikan class dan class member.

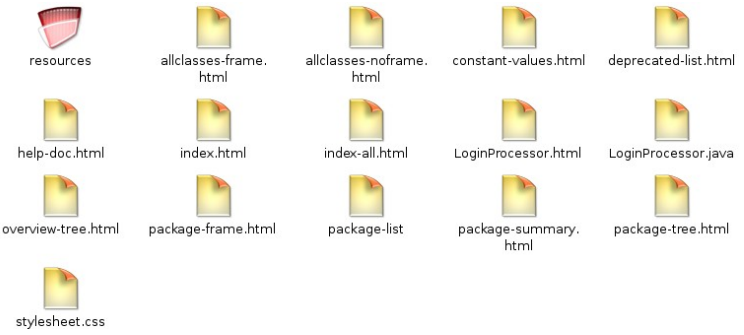
Berikut contoh komentar JavaDoc.

```
/**
* menangani halaman login di website.
* Class ini dapat dipanggil melalui URL
* http://<nama-server>/latihan/login
*/
public class LoginProcessor {
  /**
  * memeriksa username dan password.
  * Method ini akan menerima username dan password
  * dari parameter dan mencocokkannya
  * dengan isi database.
  * @param username nama user yang ingin diperiksa
  * @param password kata kunci yang ingin diperiksa
  * @return true kalau kombinasi username/password
     ada di database, false kalau tidak ada
  */
  public boolean isExist(String username,
                         String password) {
  }
}
```

Kode di atas dapat dikompilasi dengan perintah sebagai berikut:

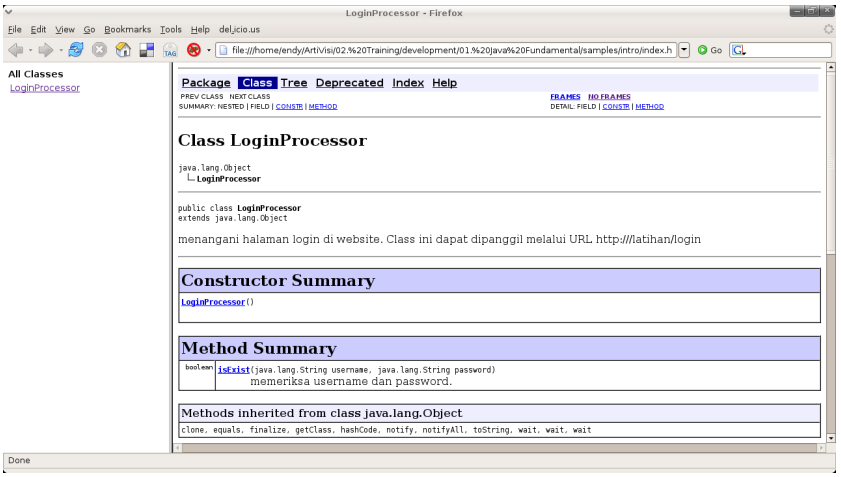
```
C:\> javadoc LoginProcessor.java
Loading source file LoginProcessor.java...
Constructing Javadoc information...
Standard Doclet version 1.6.0
Building tree for all the packages and classes...
Generating LoginProcessor.html...
Generating package-frame.html...
Generating package-summary.html...
Generating package-tree.html...
Generating constant-values.html...
Building index for all the packages and classes...
Generating overview-tree.html...
Generating index-all.html...
Generating deprecated-list.html...
Building index for all classes...
Generating allclasses-frame.html...
Generating allclasses-noframe.html...
Generating index.html...
Generating help-doc.html...
Generating stylesheet.css...
```

Perintah `javadoc` ini akan menghasilkan banyak file di dalam target folder, seperti dapat dilihat di screenshot berikut.



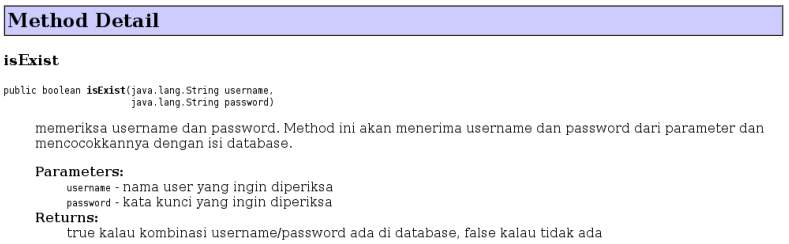
Kita bisa buka file `index.html` dengan menggunakan browser.

Berikut adalah tampilan browser yang telah membuka `index.html`.



Seperti kita lihat, tampilan hasil JavaDoc terdiri dari dua frame. Sebelah kiri berisi daftar class yang berhasil ditemukan `javadoc`, dalam hal ini hanya ada satu class `LoginProcessor`. Di sebelah kanan, disajikan informasi rinci tentang class tersebut. Pada screenshot di atas kita bisa lihat ada bagian Constructor Summary dan Method Summary.

Keterangan tentang method `isExist` dapat dilihat pada screenshot berikut.



Penjelasan method yang kita tuliskan di atas telah disalin dengan rapi ke dalam dokumentasi HTML.

Dengan demikian, kita bisa membuat dokumentasi source code dengan mudah, karena perangkat pembantunya sudah tersedia.

Tipe data

Salah satu hal pertama yang kita perlu pahami dalam bahasa pemrograman apapun adalah bagaimana bahasa tersebut menyimpan data. Java mengenal dua jenis data: primitif dan referensi. Mari kita lihat satu persatu.

Primitif

Tipe data primitif menyimpan tipe data sederhana. Berikut adalah daftar tipe data primitif yang ada dalam Java.

- Bilangan bulat
    - byte. Berukuran 8 bit.
    - short. Berukuran 16 bit
    - int. Berukuran 32 bit
    - long. Berukuran 64 bit
- Bilangan pecahan
    - float. Berukuran 32-bit IEEE 754
    - double. Berukuran 64-bit IEEE 754
- Non bilangan
    - char. Menyimpan satu huruf. Berukuran 16 bit Unicode
    - boolean. Menyimpan nilai `true` atau `false`.

Referensi

Tipe data referensi diklasifikasikan menjadi dua jenis: array dan object. Tipe data array nantinya akan dibahas lebih lanjut dalam bahasan tentang Collection Framework. Sedangkan tipe data object akan dibahas dalam bagian tersendiri tentang Object.

Passing Variabel

Hal penting lainnya dalam belajar bahasa pemrograman adalah mengetahui bagaimana suatu variabel dikirim ke method atau function lain (mekanisme passing). Ada dua mekanisme passing, yaitu *by value* dan *by reference*.

Dalam Java, tipe data primitif dipassing *by value*, sedangkan tipe data array dan object dipassing *by reference*. Untuk mengilustrasikannya, mari kita buat dua buah method, yang satu menerima masukan int, satu lagi menerima masukan array.

```
package tutorial.fundamental.variable;
public class PassingVariables {
  public void cetakDanTampilkan(int x){
    x = x + 1;
    System.out.println("Nilai dalam method: "+x);
  }
  public void cetakDanTampilkan(int[] x){
    x[0] = x[0] + 1;
    System.out.println("Nilai dalam method: "+x[0]);
  }
}
```

Seperti kita lihat, kedua method melakukan hal yang sama. Mereka menambahkan nilai variabel yang dipassing dengan angka 1, kemudian mencetaknya di konsol.

Walaupun sama, tetapi kedua method ini akan berperilaku berbeda jika dijalankan. Mari kita tambahkan method `main` untuk menjalankan contoh ini.

```
public static void main(String[] xx) {
  PassingVariables pv = new PassingVariables();
  int z = 5;
  System.out.println("Sebelum: "+z);
  pv.cetakDanTampilkan(z);
  System.out.println("Setelah: "+z);

  int[] y = {5};
  System.out.println("Sebelum: "+y[0]);
  pv.cetakDanTampilkan(y);
  System.out.println("Setelah: "+y[0]);
}
```

Bila kode ini dijalankan, hasilnya adalah seperti ini.

```
Sebelum: 5
Nilai dalam method: 6
Setelah: 5
Sebelum: 5
Nilai dalam method: 6
Setelah: 6
```

Seperti kita lihat, ada perbedaan di akhir eksekusi method. Variabel bertipe primitif nilainya tetap 5, sedangkan variabel bertipe array nilainya menjadi 6.

Passing by value artinya yang dikirim (passing) adalah nilainya (by value). Jadi, ketika variabel dimasukkan ke dalam method, Java akan membuatkan satu variabel lain bernilai sama. Ketika kita lakukan pertambahan dengan 1, yang ditambah adalah variabel yang baru, **bukan yang asli**. Ini sebabnya ketika kita cetak nilai variabel setelah method dijalankan, nilai variabel tidak berubah.

Passing by reference artinya yang dikirim (passing) adalah alamat memorinya (by reference). Dengan demikian, variabel asli dan

variabel yang dikirim menunjuk ke lokasi memori yang sama. Ketika variabel di dalam method nilainya ditambah satu, variabel yang asli ikut berubah nilainya. Ini sebabnya setelah eksekusi method, variabel asli akan berubah nilainya.

Operator

Operator juga adalah hal yang umum dalam bahasa pemrograman. Mari kita lihat contoh kode berikut.

```
int x = 3;
int y = 5;
int z = x + y;
```

Ada dua operator yang digunakan pada kode di atas, yaitu operator = dan +.

Nilai yang berada di sisi kiri atau sisi kanan operator disebut dengan *operand*. Berdasarkan jumlah operand-nya, kita dapat mengklasifikasikan operator menjadi:

- Unary operator: jumlah operand cuma satu. Contoh:

```
y++.
```

- Binary operator: memiliki dua operand. Contoh:

```
x + y.
```

- Ternary operator: memiliki tiga operand. Contoh:

```
success ? “Ok” : “Not OK”
```

Berdasarkan penggunaannya, kita dapat klasifikasikan operator menjadi:

- Aritmatika
- Relasional
- Logika

- Assignment
- Lain-lain

Mari kita lihat satu persatu.

Aritmatika

Operator ini digunakan untuk melakukan operasi matematika. Berikut adalah daftar operator aritmatika.

Operator	***Kegunaan***	***Contoh***
+	Pertambahan	x + y
-	Pengurangan	x - y
*	Perkalian	x * y
/	Pembagian	x / y
%	Nilai sisa dari pembagian (modulus)	x % y

Pada saat ini mungkin ada pembaca yang berkomentar, bagaimana dengan operasi perpangkatan? Pada bahasa pemrograman lain ada yang menulis 2 pangkat 3 seperti ini:

```
2 ^ 3
```

Di Java, operasi perpangkatan dilakukan dengan menggunakan method pow dari class Math. 2 pangkat 3 ditulis seperti ini dalam Java.

```
Math.pow(2,3)
```

Modulus adalah perhitungan nilai sisa. Apabila kita memiliki kode program seperti ini:

```
int x = 10;
int y = 6;
int z = 10 % 3;
```

Maka z akan bernilai 4, karena 10 dibagi 3 menghasilkan 1 dan sisanya adalah 4.

Selain operator di atas, ada lagi operator aritmatika yang berguna

untuk mempersingkat kode, yaitu operator ++ dan --. Pembaca yang berasal dari latar belakang C, C++, dan PHP tentu sudah tidak asing lagi dengan operator ini.

Operator ++ menambah satu ke operand-nya. Jadi kalau nilai awal variabel z adalah 5, maka z++ sama dengan z = z + 1. Sehingga nilai akhir z adalah 6.

Operator -- bekerja sebaliknya, yaitu mengurangi satu.

Yang perlu diperhatikan adalah, operator ini dapat diletakkan di belakang operand (z++), ataupun di depan operand (++z). Penempatan yang berbeda akan menghasilkan hal yang berbeda pula.

Mari kita ilustrasikan melalui contoh kode.

```
int x = 10;
System.out.println(x++);
System.out.println(x);
```

Bila kode di atas dijalankan, hasilnya adalah:

```
10
11
```

Operator ++ yang diletakkan di belakang operand, akan menambah satu **setelah** operand dievaluasi (dalam hal ini dicetak dengan perintah `println`). Sedangkan bila diletakkan di depan operand, akan menambah satu **sebelum** operand dievaluasi. Berikut contoh kodenya.

```
int x = 10;
System.out.println(++x);
System.out.println(x);
```

Dan hasilnya adalah sebagai berikut.

```
10
11
```

Relasional

Operator relasional membandingkan operand dan menentukan hasil perbandingan tersebut. Operator relasional dalam Java adalah sebagai berikut.

Operator	*Contoh*
>	3 > 5 => false
<	3 < 5 => true
>=	3 >= 5 => false
<=	3 <= 5 => true
!=	3 != 5 => true
==	3 == 5 => false

Kondisional

Operator kondisional berhubungan erat dengan nilai true atau false. Operator ini biasanya digunakan dalam blok if-else atau while. Berikut adalah operator kondisional yang tersedia di Java.

Operator	*Nama*	*Contoh*
&&	And	true && false => false
||	Or	true || false => false
!	Not	!true => false

Selain operator &&, kita juga dapat menggunakan operator & untuk melakukan operasi And. Perbedaan antara && dan & adalah dengan &&, operand kedua tidak diperiksa apabila hasil akhir sudah dapat ditentukan. Perhatikan kode berikut sebagai ilustrasi.

```
if (username.isCorrect() && password.isCorrect()) {}
```

Pada kode di atas, bila `username.isCorrect()` menghasilkan nilai false, sudah pasti hasil keseluruhannya adalah false, apapun hasil dari `password.isCorrect()`. Bila kita menggunakan &&,

`password.isCorrect()` tidak dieksekusi.

Demikian juga untuk operator Or. Kita bisa menggunakan || atau | dengan perilaku yang sama dengan && dan &.

Assignment

Lainnya

Aliran kontrol

Aliran kontrol adalah teknik pemrograman untuk menentukan jalannya program. Fitur ini ada di semua bahasa pemrograman, hanya sintaksnya saja yang berbeda. Jadi, arti dari masing-masingnya tidak dijelaskan lebih lanjut.

if

Contoh penggunaan if di Java adalah sebagai berikut.

```
if (year % 4 == 0) {
  return 29;
} else {
  return 28;
}
```

Apabila pilihannya lebih dari dua, kita gunakan `else if`.

```
if (username.equals(“administrator”)) {
  // lakukan sesuatu
} else if (username.equals(“guest”)) {
  // lakukan sesuatu
} else {
  // lakukan sesuatu
}
```

switch

Berikut adalah contoh kode switch, diambil dari tutorial Java.

```
switch (month) {
  case 1:
  case 3:
  case 5:
  case 7:
  case 8:
  case 10:
  case 12:
    numDays = 31;
    break;
  case 4:
  case 6:
  case 9:
  case 11:
    numDays = 30;
    break;
  case 2:
    if ( ((year % 4 == 0) && !(year % 100 == 0))
                          || (year % 400 == 0) )
      numDays = 29;
    else
      numDays = 28;
    break;
}
```

for

For digunakan untuk perulangan. Berikut contoh kode for, diambil dari Java Tutorial dari Sun.

```
int[] arrayOfInts = { 32, 87, 3, 589 };


for (int i = 0; i < arrayOfInts.length; i++) {
  System.out.print(arrayOfInts[i] + " ");
}
```

while

Bentuk while biasanya digunakan untuk perulangan yang belum dapat ditentukan titik akhirnya, seperti misalnya membaca hasil query database yang jumlahnya belum diketahui.

```
while(resultSet.next()) {
  // lakukan sesuatu
}
```

break

Keyword break digunakan untuk segera keluar dari loop. Contoh kodenya adalah sebagai berikut.

```
public void displayMembers()
  String[] members = {“endy”, “oky”, “anton”};
  for(int i = 0; i<members.length; i++) {
    System.out.println(members[i]);
    if(members[i].equals(“oky”) break;
  }
  System.out.println(“Looping selesai”);
}
```

akan menampilkan:

```
endy
oky
Looping selesai
```

Jadi begitu keyword break dijalankan, loop yang mengelilinginya langsung disudahi.

continue

Kalau keyword break menghentikan loop, keyword continue hanya menghentikan iterasi yang sedang berjalan. Mari kita modifikasi contoh kode di atas untuk mengilustrasikannya.

```
public void displayMembers()
  String[] members = {“endy”, “oky”, “anton”};
  for(int i = 0; i<members.length; i++) {
    if(members[i].equals(“oky”) continue;
    System.out.println(members[i]);
  }
  System.out.println(“Looping selesai”);
}
```

Kode di atas akan menampilkan:

```
endy
anton
Looping selesai
```

return

Keyword return akan langsung mengakhiri method yang mengelilinginya.

Kita akan gunakan contoh kode yang sama, hanya keyword continue yang kita ganti dengan return.

```
public void displayMembers()
  String[] members = {“endy”, “oky”, “anton”};
  for(int i = 0; i<members.length; i++) {
    if(members[i].equals(“oky”) return;
    System.out.println(members[i]);
  }
  System.out.println(“Looping selesai”);
}
```

Kode di atas akan menampilkan:

```
endy
```

Seperti kita lihat, baris `Looping selesai` tidak dijalankan, karena method langsung berakhir.