LAMPIRAN 3

DOKUMENTASI

Rekayasa perangkat lunak merupakan upaya penanganan kompleksitas. Pada rekayasa perangkat lunak, kita seharusnya mendefinisikan dan mendokumentasikan apa yang hendak kita bangun bahkan sebelum proses pembangunan itu sendiri. Bila dokumentasi kita lengkap dan jelas, maka saat memperluas kemampuan perangkat lunak, maka kita cukup memperbarui dari yang telah ada serta menggunakan dokumentasi untuk memulai kerja.

Meski pembuatan dokumentasi merupakan bagian aktivitas yang paling tidak menyenangkan dalam pengembangan perangkat lunak tapi aktivitas ini sangat diperlukan. Aplikasi yang berkualitas dibangun dengan memperhatikan kebutuhan pemeliharan dimana aplikasi didokumentasikan dengan baik serta mengikuti metodologi rekayasa perangkat lunak yang telah terbukti.

Salah satu bentuk dokumentasi adalah dokumentasi kode program. Java menyediakan kakas untuk membuat dokumentasi program yaitu **javadoc**. Utilitas **javadoc** adalah program kecil di JDK untuk membantu menciptakan dokumentasi dari file kode program. Utilitas **javadoc** menggunakan gabungan informasi dari kompilator dan komentar yang dibuat pemrogram untuk membangun dokumentasi berupa file-file HTML yang mendeskripsikan program Java. Selama kita mengelola komentar di kode program maka dokumentasi aplikasi kita akan selalu merefleksikan *state* terakhir program aplikasi.

Untuk menghasilkan halaman HTML yang mendokumentasi program dengan tepat, maka kita harus memasukkan item-item berikut di file kode sumber, yaitu:

- Deskripsi objek
- Deskripsi semua atribut `public`, `protected`, atau `private protected`
- Deskripsi semua metode `public`, `protected`, atau `private protected`

Di antara keunggulan kakas dokumentasi **javadoc** adalah kita dapat menempelkan *HTML tags* di sembarang komentar yang akan muncul di hasil dokumentasi akhir HTML. Kita bahkan dapat menempelkan dokumentasi berupa **applet**! Lebih penting lagi, dokumentasi memungkinkan kita menciptakan *link* ke dokumentasi terkait di luar dokumentasi menggunakan *keyword* `@see`. Batasan yang diberikan adalah kita tidak dapat menempelkan *HTML tag* `<HR>` atau `<H1>` sampai `<H6>` di dalam komentar. Meski penempelan ini tidak membangkitkan kesalahan, penempelan dapat menghasilkan dokumentasi yang tidak diharapkan.

Begitu file sumber telah lengkap – kode dan komentar – maka kita siap membangun dokumentasi aplikasi. Untuk melakukan, kita cukup mengeksekusi perintah `javadoc` dengan file kelas (*class*) sebagai argumen. Perintah `javadoc` menciptakan lima keluaran file HTML dokumentasi kelas.

File yang dihasilkan didaftarkan sebagai berikut:

Nama file	Deskripsi
`YourClassName.html`	Dokumentasi rinci untuk kelas.
`Package-YourPackageName.html`	Untuk masing-masing paket, salah satu file yang diciptakan mendaftarkan semua kelas di paket.
`AllNames.html`	Daftar atribut dan metode untuk semua kelas (di dalam kasus ini, hanya satu kelas) secara terurut alphabet.
`packages.html`	Daftar semua paket di aplikasi.
`tree.html`	Daftar semua kelas yang ditunjukkan sebagai pohon pewarisan.

Kita mungkin ingin mendokumentasi lebih dari satu kelas pada satu waktu. Kita dapat memberi perintah ke **javadoc** untuk menerima *option* sehingga kita dapat menyesuaikan dokumentasi yang kita perlukan.

Perintah eksekusi utilitas `javadoc` berbentuk sebagai berikut:

- `javadoc [options] file`
- `javadoc [options] package`

Option	Komentar
`-author`	Jika terdapat `@author` tags di komentar, daftarkan di dokumentasi.
`-authors`	Sama dengan `-author`.
`-classpath` path list	Secara *manual* men*set classpath* untuk menemukan kode sumber. Variabel lingkungan `CLASSPATH` umumnya diset ke dimana file kelas `.class` berada, sehingga kita hampir selalu harus men-spesifikasikan *option* ini.
`-d` directory	Direktori dimana dokumentasi yang dihasilkan ditempatkan.
`-depend` package list	Daftar kebergantungan paket yang didokumentasikan.
`-version`	Seperti dengan `@author`, kita harus menspesifikasikan informasi versi yang seharusnya dimasukkan ke dokumentasi.
`-verbose`	Mencetak informasi mengenai kemajuan yang dilakukan `javadoc` saat menghasilkan dokumentasi

L3.1 Komentar Umum

Kita dapat meletakkan komentar dengan *tags:*

@author nama
@version teks
@since teks
@deprecated teks
@see link

Masing-masing deskripsi kode Java ditulis sebagai komentar untuk dokumentasi sebelum elemen yang hendak dideskripsikan dokumentasi. Untuk menandai komentar agar bertindak sebagai komentar yang akan diolah oleh kakas `javadoc`, maka kita harus memberi komentar dalam format berikut ini:

```
/** deskripsi */
```

Di halaman berikut adalah *keyword* di komentar `javadoc` beserta arti-artinya.

Keyword	Deskripsi
`@author`	Digunakan di komentar kelas. Mengidentifikasi siapa yang menulis kelas. Kita dapat memiliki banyak `@author` untuk satu kelas.
`@exception`	Digunakan di komentar metode. Mengindentifikasi nama kelas untuk *exception* yang dilempar metode. Kita dapat mempunyai banyak `@exception` untuk metode.
`@param`	Digunakan di komentar metode. Mengidentifikasi parameter metode. Kita seharusnya menspesifikasikan sebanyak parameter yang dimiliki metode.
`@return`	Digunakan di komentar metode. Mendeskripsikan nilai yang dikirim metode.
`@see`	Digunakan di sembarang tipe komentar. Memungkinkan kita mengacu kelas-kelas, atribut-atribut, atau metode-metode lain dengan *hypertext link*.
`@version`	Digunakan di komentar kelas. Mengidentifikasi nomor versi untuk kelas.

Idealnya, kita mendokumentasi kode pada suatu level kejelasan tertentu.

L3.2 Komentar Kelas

Komentar kelas harus ditempatkan setelah pernyataaan *import*, secara langsung sebelum definisi kelas.

Contoh:

```
/**
 * Kelas
 * Kelasi ini merepresentasikan ....
 * @version 1.0
 * @author Bambang Hariyanto
 */
public class C{
...
}
```

L3.3 Komentar Metode

Masing-masing komentar metode harus disusul metode yang dikomentari. Selain *general purpose tags*, kita dapat menggunakan *tag* berikut:

```
@param deskripri variabel
```

Tag ini menambahkan isian bagian "*parameters*" di metode saat itu. Deskripsi ini dapat merentang banyak baris dan menggunakan *HTML tags*. Semua *tag @param* untuk satu metode harus berpasangan dengan:

@return deskripri

Tag ini menambahkan isian bagian "*return*" di metode saat itu. Deskripsi dapat merentang banyak baris dan menggunakan HTML tags.

@throws deskripri kelas

Tag ini menambahkan catatan metode ini melemparkan *exception*.

Contoh komentar metode:

```
/**
 Akar kwadrat
 @param x angka yang akan dicari akar kwadratnya, harus bilangan positif
 @return nilai akar kwadrat dari x
 */
public double akarKwadrat(double x) {
  return MATH.sqrt(x);
}
```

L3.4 Komentar *Field*

Kita hanya perlu mendokumentasikan *field-field* publik, umumnya berupa konstanta-konstanta statik.

```
/**
  PHI nilai phi
*/
 public static final double PHI = 3.14;
```