

Dasar Pemrograman Java di Ubuntu

Arya Adhyaksa Waskita

21 Nopember 2010

Daftar Isi

1	Pendahuluan	2
1.1	Program Java Sederhana	2
2	Lebih jauh tentang Java	5
2.1	Berorientasi obyek	5
2.2	Obyek Java Serdehana	5
2.3	Dokumentasi	7
3	Menggunakan ant	10
3.1	Pendahuluan	10
3.2	Komponen ant	10
3.3	Menggunakan ant	11

Bab 1

Pendahuluan

Java adalah bahasa pemrograman yang bersifat multiplatform. Sifat ini diperoleh Java dengan cara mengkompilasi kode sumber menjadi byte code yang bersifat sebagai kode antara. Untuk menjalankannya, *byte code* diinterpretasi oleh *interpreter* yang sesuai dengan sistem operasi yang digunakan. *Interpreter* yang berbeda untuk setiap sistem operasi tersebut, akan bekerja menginterpretasi byte code sesuai sistem operasi yang digunakan.

Sebelum memulai pemrograman dengan Java di Ubuntu, pastikan semua pustaka yang dibutuhkan telah tersedia. Sebagai perbandingan, tulisan ini menggunakan Ubuntu 10.10. Java Development Kit (OpenJDK) yang diperlukan untuk memprogram dengan Java diinstal dengan perintah berikut.

```
sudo apt-get install openjdk-6-jdk openjdk-6-jre openjdk-6-doc
```

openjdk-6-jdk adalah pustaka yang akan digunakan dalam memprogram dengan Java. Dalam pustaka tersebut, telah tersedia beragam class dasar untuk antarmuka pengguna (user interface), I/O, akses database, dan sebagainya. Paket openjdk-6-jdk harus ada di mesin yang digunakan untuk mengembangkan aplikasi berbasis Java.

Paket openjdk-6-doc akan memberikan informasi mengenai berbagai class dalam pustaka Java dan cara menggunakannya. Gambar 1.1 memberikan ilustrasi tentang petunjuk penggunaan berbagai class dalam pustaka Java. Sedangkan paket openjdk-6-jre adalah interpreter atau disebut juga mesin virtual Java. Paket ini harus tersedia di mesin yang akan menjalankan aplikasi berbasis Java.

1.1 Program Java Sederhana

Ada baiknya jika kita menggunakan editor teks yang mampu memberikan fasilitas *syntax highlight* seperti pada Geany¹. Untuk menginstalnya gunakan perintah berikut.

```
sudo apt-get install geany
```

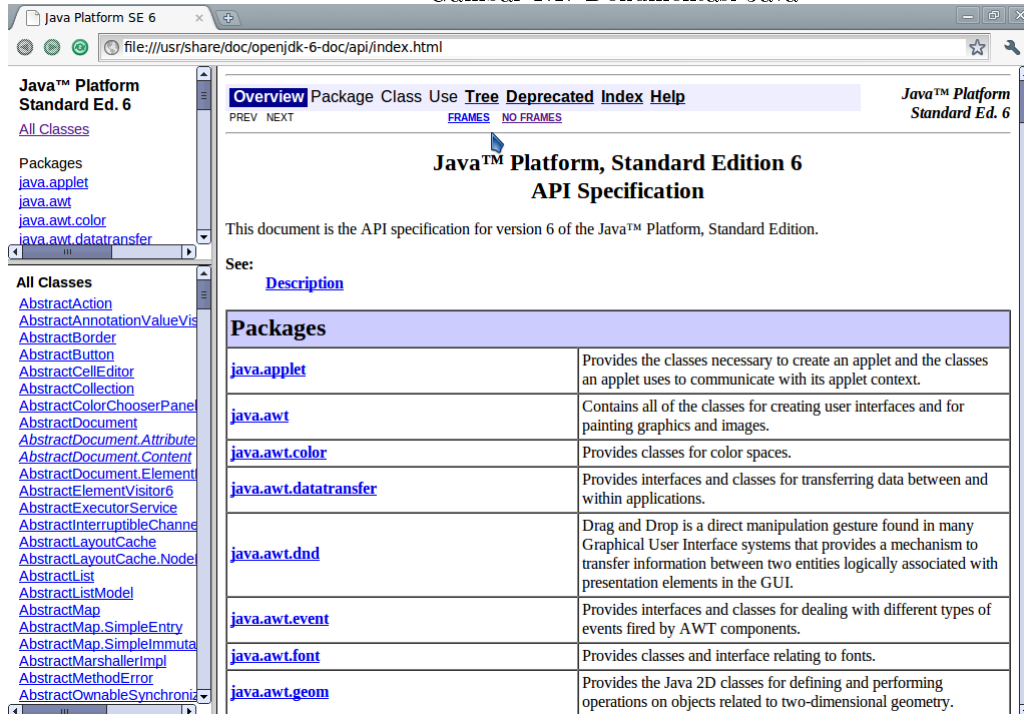
Sedangkan tampilan editor teks Geany adalah seperti Gambar 1.2

Sebagai awal mempelajari Java, coba buat program sederhana berikut ini. Kemudian, simpan ke dalam berkas dengan nama yang sama dengan nama class yang didefinisikan ditambah ekstensi *.java.

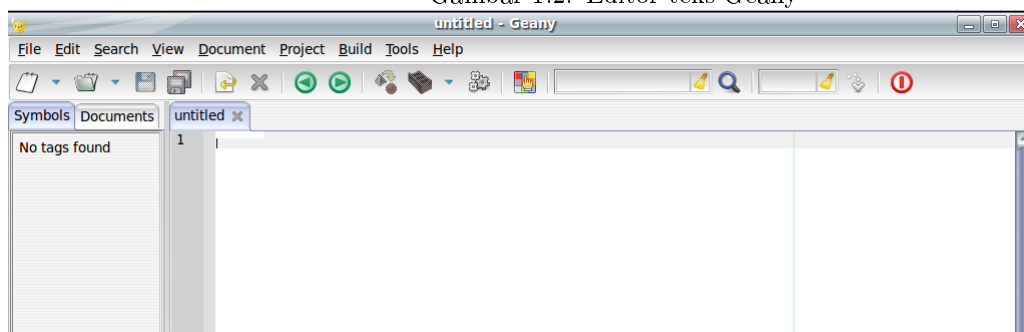
```
1 public class Hello {  
2     public static void main(String[] args) {  
3         System.out.println("Apa_kabar_rekan_Umet?");  
4     }  
5 }
```

¹<http://www.geany.org/>

Gambar 1.1: Dokumentasi Java



Gambar 1.2: Editor teks Geany




Mari kita bahas satu persatu baris dalam kode tersebut

- Java adalah bahasa pemrograman berorientasi obyek, sehingga pada setiap kode Java digunakan kata **class** seperti pada baris ke-1.
- Java membutuhkan sebuah class yang memiliki fungsi dengan nama **main** untuk dapat dieksekusi seperti pada baris ke-2.
- Pada baris ke-2, fungsi **main** membawa parameter array bertipe String, ditandai dengan karakter "String[] args".
- Baris ke-3 adalah cara Java menuliskan karakter dalam tipe data String ke layar (*standard output*).

Jika sudah dibuat, bagaimana mengeksekusinya? Kompilasi program yang telah dibuat dengan perintah berikut.

```
javac Hello.java
```



Jika kita menggunakan Geany, kita dapat menggunakan tombol . Selanjutnya, akan terbentuk berkas dengan nama Hello.class yang merupakan byte code dari kode sumber Java. Berkas tersebut kemudian diinterpretasi dengan perintah berikut. Perhatikan, tidak ada ekstensi pada eksekusi perintah tersebut.

```
java Hello
```

Bab 2

Lebih jauh tentang Java

2.1 Berorientasi obyek

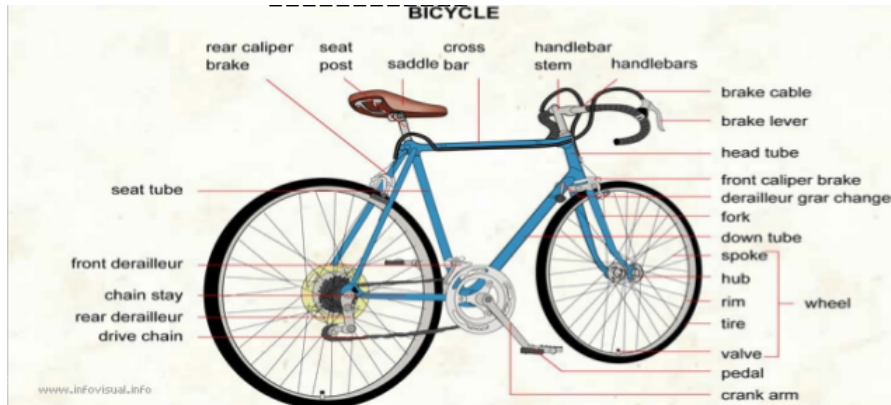
Seperti telah disebutkan di awal, Java adalah bahasa pemrograman berorientasi obyek. Ini adalah konsep pemrograman yang memandang sebuah proses bisnis bukan dari tahapan proses melainkan dari para pelakunya. Data dan fungsi dienkapsulasi dalam sebuah class. Konsep ini seperti konsep umum dalam kehidupan sehari-hari kita. Perhatikan Gambar 2.1 berikut.

Setiap elemen dalam Java dikemas dalam sebuah class sebagaimana elemen-elemen pada Gambar 2.1 dikemas dalam obyek sepeda. Elemen-elemen tersebut terkait dengan apa yang dimiliki (*attributes*) serta apa yang mampu dilakukan (*methods*). Dari contoh obyek sepeda tersebut, setiap sepeda pasti memiliki elemen seperti ban, rantai, saddle, dan elemen lainnya. Akan tetapi, antara sepeda yang satu dengan sepeda lainnya memiliki nilai berbeda untuk masing-masing elemen. Sebagai contoh, sepeda gunung akan memiliki ukuran ban yang berbeda dengan sepeda balap. Class yang mengemas berbagai elemen sepeda tersebut dapat dianggap sebagai cetakan. Sedangkan dari cetakan tersebut dibuat sepeda A, sepeda B, dan sebagainya.

2.2 Obyek Java Serdehana

Setelah mengetahui konsep obyek pemrograman Java, mari mencoba menerapkannya. Karena merupakan obyek, nama sebuah class juga harus merupakan kata benda. Sebagai contoh, mari membuat sebuah class yang kita beri nama Salam. Perhatikan kode berikut.

```
1 public class Salam {
2     private String namaTeman;
3     public Salam(String nama) {
4         namaTeman=nama;
5     }
6
7     public String berikanSalam() {
8         return "Assalammualaikum," + namaTeman + "!";
9     }
10
11     public void gantiNama(String nama) {
12         namaTeman=nama;
13     }
14 }
```

Gambar 2.1: Sepeda, diambil dari <http://romisatriawahono.net/publications/2000/romi-ooad.pdf>

Penjelasan kode tersebut adalah sebagai berikut.

- Baris ke-1 adalah deklarasi sebuah class. *Keyword* **public** menunjukkan batasan (*scope*) dapat dipanggilnya sebuah class.
- Baris ke-2 menunjukkan deklarasi atribut, dinamai sebagai **namaTeman** dengan tipe data String. *Keyword* **private** juga menunjukkan batasan (*scope*) dapat dipanggilnya atribut tersebut.
- Baris ke-3 sampai ke-5 merupakan deklarasi **Constructor**, sebuah fungsi yang diberi nama yang sama persis dengan nama class-nya. Method constructor akan mendeskripsikan seperti apa obyek yang tercetak dari class tersebut.
- Baris ke-7 sampai ke-9 merupakan deklarasi fungsi yang mengembalikan sebuah nilai, yang dalam hal ini bertipe String.
- Baris ke-11 sampai ke-13 merupakan deklarasi fungsi yang dalam eksekusinya membutuhkan parameter masukan tapi tidak memberikan nilai luaran.

Lalu, bagaimana menggunakannya? Seperti telah dijelaskan, sebuah class Java hanya dapat dieksekusi apabila class tersebut memiliki fungsi dengan nama **main**. Pada kode di atas, kita tidak memiliki fungsi bernama **main**, sehingga class Salam tidak dapat dieksekusi. Untuk dapat menggunakannya, kita perlu class lain yang memiliki atribut class Salam tersebut. Perhatikan kode berikut.

```

1 public class BeriSalam {
2     private static Salam umet;
3     public static void main(String[] nama) {
4         String salam;
5         umet = new Salam("Dendy");
6         salam = umet.berikanSalam();
7         System.out.println(salam);
8     }
9 }

```

Penjelasannya adalah sebagai berikut.

- Baris ke-1 adalah deklarasi class.
- Baris ke-2 adalah deklarasi atribut bertipe `Salam` dan diberi nama `umet`. Tipe atribut bisa tipe data primitif atau class lainnya.
- Baris ke-3 sampai 8 adalah deklarasi fungsi **main**. Di dalamnya ada deklarasi variabel bertipe `String` dengan nama `salam` yang dideklarasikan di baris ke-5.
- Baris ke-5 adalah deklarasi obyek `umet`, obyek dari class `Salam`. Obyek ini membawa parameter `String` seperti didefinisikan di class `Salam`.
- Baris ke-6 adalah pernyataan yang memerintahkan obyek `umet` menjalankan fungsinya, yaitu `berikanSalam`. Karena fungsi tersebut mengembalikan variabel bertipe `String`, kembalian tersebut disimpan dalam variabel `salam`.
- Baris ke-7 adalah pernyataan untuk menampilkan karakter pada layar. Karakter tersebut adalah variabel `salam` yang diberikan nilai di baris ke-6.

2.3 Dokumentasi

Java memberikan fasilitas untuk membuat dokumentasi untuk class yang kita kembangkan. Dokumentasi yang dimaksud adalah seperti yang ada pada Gambar 1.1. Perhatikan kode berikut.

```
1 public class Salam {
2     private String namaTeman;
3
4     /**
5      * Membangun obyek Salam, yang dapat memberikan
6      * Salam pada seorang rekan
7      * @param Nama seorang rekan yang akan kita beri Salam
8      */
9     public Salam(String nama) {
10         namaTeman=nama;
11     }
12
13     /**
14      * Menyampaikan Salam dengan "Assalammualaikum"
15      * @return Pesan berisi "Assalammualaikum" dan nama
16      * seorang rekan
17      */
18     public String berikanSalam() {
19         return "Assalammualaikum," + namaTeman + "!";
20     }
21
22     /**
23      * Mengganti nama rekan yang akan diberi Salam
24      * @param Nama seorang rekan yang akan kita beri Salam
25      */
26     public void gantiNama(String nama) {
27         namaTeman=nama;
28     }
29 }
```


Gambar 2.2: Dokumentasi yang dihasilkan dengan perintah javadoc



Setelah disimpan, coba jalankan perintah berikut di terminal, di directory di mana Kode tersebut disimpan.

```
javadoc Salam.java
```

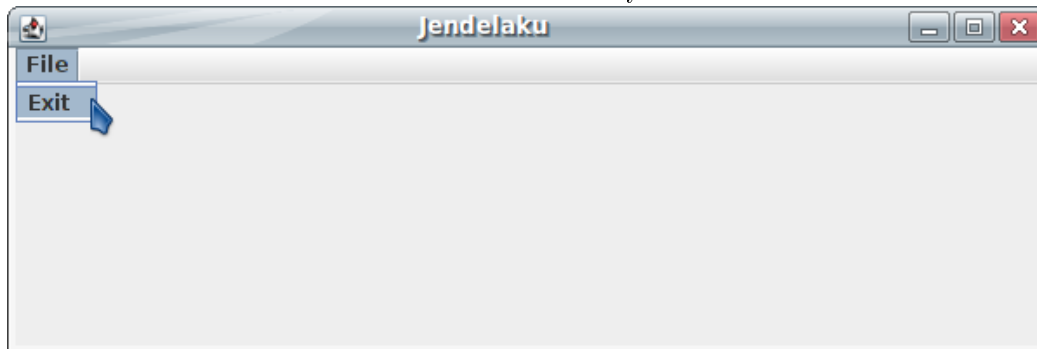
Setelah itu, buka directory di mana kode Salam.java disimpan. Kita akan menemukan beberapa file berekstensi html yang salah satunya adalah index.html. Jika dibuka di *internet browser*, kita akan mendapatkan informasi seperti Gambar 2.2.

Dari potongan gambar tersebut kita dapat melihat alur penyampaian (*passing*) dan pengembalian (*return*) parameter yang kita rancang terefleksikan. Dengan pola yang sedikit dibalik, kita dapat menggunakan pustakan yang sudah disediakan oleh Java untuk membangun aplikasi kita berbasis Java.

Mari perhatikan kode berikut, di mana kita akan membangun sebuah obyek yang kita beri nama Jendela, sebuah obyek antarmuka grafis dari class **JFrame** di package *javax.swing*. Obyek tersebut memiliki karakteristik dapat diberi judul sesuai keinginan kita, berukuran 600x400 pixel, muncul pertama kali di tengah layar dan ketika ditutup, seluruh obyek yang terkait dengan obyek tersebut akan dimatikan.

Setelah selesai simpan, kompilasi dan jalankan. Kita akan mendapatkan obyek jendela kita seperti pada Gambar 2.3. Selanjutnya, kita dapat menambahkan obyek lain seperti menu di mana pengguna dapat berinteraksi dengan obyek jendela kita. Tentu dengan mempelajari dokumentasi berbagai pustaka yang telah disediakan Java.

Gambar 2.3: Obyek Jendelaku



```

1  import javax.swing.*;
2  import java.awt.event.*;
3  public class Jendelaku extends JFrame
4      implements ActionListener {
5      /**
6       * Membangun obyek Frame yang memiliki judul
7       * seperti diberikan melalui parameter
8       * @param Judul halaman yang diberikan
9       */
10     public Jendelaku (String judul) {
11         setTitle(judul);
12         setSize(600,200);
13         setLocationRelativeTo(null);
14         setVisible(true);
15         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
16         JMenuBar menuBar=new JMenuBar();
17         JMenu file=new JMenu("File");
18         JMenuItem exit=new JMenuItem("Exit");
19         file.add(exit);
20         menuBar.add(file);
21         setJMenuBar(menuBar);
22         exit.addActionListener(this);
23     }
24     /**
25      * Fungsi main, membangun obyek Frame yang diinginkan
26      */
27     public static void main(String[] args) {
28         Jendelaku jendelaku=new Jendelaku("Jendelaku");
29     }
30
31     public void actionPerformed(ActionEvent e) {
32         System.exit(0);
33     }
34 }

```

Bab 3

Menggunakan ant

3.1 Pendahuluan

Ant adalah perkakas untuk mengelola proyek pengembangan aplikasi berbasis Java. Ant dikembangkan karena perkakas serupa seperti *make* dipandang memiliki banyak keterbatasan. Ant diklaim lebih baik karena tidak berorientasi perintah shell. Sebaliknya, ant menggunakan format XML dan bersifat multi-platform.

Untuk menggunakannya, jalankan perintah berikut.

```
sudo apt-get install ant ant-doc
```

3.2 Komponen ant

Seperti telah dijelaskan sebelumnya, berkas konfigurasi ant menggunakan format XML. Pada berkas konfigurasi tersebut, ada beberapa komponen penting yang perlu diperhatikan.

- Berkas ant diberi nama **build.xml** dan karenanya diawali dengan baris berikut.

```
<?xml version="1.0"?>
```

- Tag “project”. Setiap berkas ant harus memiliki tag ini. Atribut yang dimilikinya antara lain:
 - name: merupakan atribut yang menunjukkan nama project.
 - default: merupakan atribut yang menunjukkan target default. Berkas ant akan mendeskripsikan beberapa tugas. Dengan atribut default, sebuah tugas pasti akan dikerjakan dengan hanya memanggil perintah *ant* di directory di mana berkas **build.xml** berada.
 - basedir: merupakan atribut yang menunjukkan directory project. Umumnya diset sebagai “.” atau *current directory*.
- Tag “target”. Di tag inilah deskripsi sebuah pekerjaan berada. Jika nilai tag ini sama dengan target yang didefinisikan sebagai default, maka target inilah yang akan dikerjakan saat perintah *ant* dieksekusi. Beberapa atribut pada tag ini antara lain adalah:
 - name: merupakan atribut nama target. Jika hanya pekerjaan yang dideskripsikan pada tag target ini yang akan dikerjakan, nama target ini akan menjadi argumen pada saat perintah ant dieksekusi. Jika tanpa argumen, maka ant akan mengerjakan pekerjaan yang dideskripsikan di target default.

- depends: merupakan atribut yang menunjukkan target yang harus dikerjakan lebih dulu sebelum target pekerjaan di tag target yang sekarang.
- description: merupakan deskripsi target yang akan ditampilkan ke layar ketika ant mengerjakan pekerjaan ini.
- Tag “properties”. Tag ini digunakan untuk mendefinisikan sebuah nilai parameter yang akan sering digunakan, sehingga kita tidak perlu mendeskripsikan nilainya, cukup parameternya saja.
- Tag tugas. Tag ini adalah definisi dari apa yang kita ingin ant bekerja untuk kita. Biasanya seputar buat dan hapus direktori, kompilasi atau memaket ke jar. Karena tugasnya bisa berbeda, maka tag juga berbeda. Dari beberapa tugas yang umum tersebut, berikut adalah tag untuk beberapa tugas berikut.
 - *javac*. Tugas ini ditujukan untuk mengkompilasi program java kita. Tag ini memiliki beberapa atribut berikut ini yang meskipun banyak tidak harus digunakan semua, seperlunya saja.
 - * srcdir: lokasi direktori di mana kode sumber disimpan
 - * destdir: lokasi direktori di mana hasil kompilasi akan disimpan
 - * includes: daftar berkas yang harus disertakan saat kompilasi, dipisahkan dengan spasi atau karakter koma
 - * classpath: lokasi classpath yang digunakan saat kompilasi
 - * sourcepath: nilai defaultnya adalah nilai yang tertera pada atribut srcdir
 - *jar*. Tugas untuk membuat berkas archive java.
 - *javadoc*. Tugas untuk membuat dokumentasi dari program yang kita buat.
 - *copy* dan *mkdir*, membuat dan menyalin berkas/direktori.

3.3 Menggunakan ant

Dengan ant, mari kita coba mengkompilasi aplikasi jendela kita yang sangat sederhana tadi. Tahap pertama, buat directory dengan nama Jendelaku-0.1. Di dalamnya, buat directory lagi dengan nama src, dan pindahkan berkas Jendelaku.java di directory src tersebut. Selanjutnya, buat berkas build.xml seperti kode berikut ini.

```

1 <?xml version="1.0"?>
2 <project name="jendelaku" default="compile" basedir=". ">
3     <property name="src" value="src"/>
4     <property name="build" value="build"/>
5     <property name="main-class" value="Jendelaku"/>
6     <path id="classpath">
7         <fileset dir="${lib.dir}" includes="**/*.jar"/>
8     </path>
9     <target name="compile">
10         <mkdir dir="${build}"/>
11         <javac srcdir="${src}" destdir="${build}"/>
12     </target>
13     <target name="jar" depends="compile">
14         <jar destfile="Jendelaku.jar" basedir="${build}">
15             <manifest>
16                 <attribute name="Main-Class" value="${main-class}"/>
17                 <attribute name="Debian-Java-Home" value="/usr/lib/jvm/java-6-openjdk"/>
18             </manifest>

```

```
19     </jar>
20   </target>
21     <target name="run" depends="jar">
22       <java jar="Jendelaku.jar" fork="true"/>
23     </target>
24     <target name="clean">
25       <delete dir="${build}"/>
26     </target>
27 </project>
```

Untuk menjalankan aplikasi jendela tersebut, gunakan perintah berikut.

ant run

Karena target “run” tergantung pada compile dan jar, memanggil ant dengan argumen run akan secara otomatis mengkompilasi kode Jendelaku.java dan mengemasnya dalam jar untuk kemudian menjalankannya.