Pertemuan I

PENGENALAN JAVA

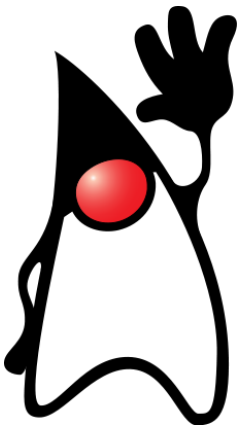
1.1 Pendahuluan

1.1.1 Sejarah

Pada tahun 1991 bahasa java lahir dari *the green project* yang dikerjakan oleh sekelompok insinyur dari *Sun Microsystem, Inc*., yang dipimpin oleh Patrick Naughton dan James Gosling. Proyek ini ditujukan untuk merancang bahasa untuk perangkat konsumer seperti *cable TV box*. Para perancang ini ingin menciptakan sebuah bahasa pemrograman yang dapat dijalankan pada semua peralatan yang tidak tergantung oleh *platform* yang digunakan oleh peralatan tersebut.

Karena orang-orang yang bekerja dalam proyek *Green* memiliki dasar bahasa C++ maka kebanyakan sintaksnya diambil dari C++ serta mengadopsi orientasi obyek bukan prosedural. Pada mulanya, James Gosling memberi nama bahasa pemrogramannya dengan "Oak", setelah terinspirasi dari pohon yang berada di seberang kantornya. Kemudian diganti dengan "Java", karena sudah ada bahasa pemrograman dengan nama Oak. Nama Java terinspirasi ketika mereka sedang minum kopi di sebuah kedai kopi, kemudian salah satu diantara mereka menyebut nama Java yang mengandung arti asal dari biji kopi.

Salah satu hasil dari *green project* adalah maskot java yang dibuat oleh Joe Palrang dan diberi nama Duke (gambar 1.1).



Gambar 1.1. Duke, Maskot Java

Produk pertama proyek *Green* adalah "*7" (*Star Seven*), yaitu sebuah kendali jarak jauh yang cerdas. Karena pasar belum tertarik dengan produk konsumer cerdas maka proyek *Green* harus menemukan pasar lain dari teknologi yang diciptakan. Pada waktu itu perkembangan internet sedang pesat, maka proyek *Green* lebih diarahkan ke teknologi internet. Pada tahun 1995, *Netscape* memutuskan membuat *browser* yang dilengkapi Java. Setelah itu diikuti diikuti IBM, *Symantec* dan *Microsoft*.

Pada awal tahun 1996, Sun secara resmi merilis versi awal Java, sehingga muncul JDK 1.1 (*Java Development Kit versi* 1.1), kemudian muncul Java 2 yang dilengkapi *Swing*, yaitu teknologi GUI (*Graphical User Interface*) yang dapat menghasilkan aplikasi desktop.

Pada tahun 1998 – 1999 lahirlah teknologi java berbasis *enterprise* yang disebut J2EE (*Java 2 Enterprise Edition*) yang diawali dengan *servlet* dan EJB kemudian diikuti JSP. Java juga menjadi lebih cepat populer di lingkungan *server side* dikarenakan kelebihanya di lingkungan *network* dan terdistribusi serta kemampuan *multithreading*. Selain itu java juga mengembangkan J2ME (*Java 2 Micro Edition*) yang dapat menghasilkan aplikasi mobile baik *games* maupun *software* yang dapat dijalankan di peralatan *mobile* seperti ponsel.

Gambar 1.2. Logo Java (Merek dagang dari Sun Microsystems, Inc)

1.1.2 Karakteristik

Berdasarkan *white paper* resmi dari SUN, Java memiliki karakteristik berikut :

a. Sederhana
   Bahasa pemrograman Java menggunakan sintaks mirip dengan C++ namun sintaks pada Java telah banyak diperbaiki terutama menghilangkan penggunaan pointer yang rumit dan *multiple inheritance*. Java juga menggunakan *automatic memory allocation* dan *memory garbage collection*.
b. Berorientasi objek (*Object Oriented*)
   Java mengunakan pemrograman berorientasi objek yang membuat program dapat dibuat secara modular dan dapat dipergunakan kembali. Pemrograman berorientasi objek memodelkan dunia nyata kedalam objek dan melakukan interaksi antar objek-objek tersebut.
c. Dapat didistribusi dengan mudah
   Java dibuat untuk membuat aplikasi terdistribusi secara mudah dengan adanya *libraries* networking yang terintegrasi pada Java.
d. Interpreter
   Program Java dijalankan menggunakan interpreter yaitu *Java Virtual Machine* (JVM). Hal ini menyebabkan *source code* Java yang telah dikompilasi menjadi Java *bytecodes* dapat dijalankan pada platform yang berbeda-beda.
e. Robust
   Java mempuyai reliabilitas yang tinggi. Compiler pada Java mempunyai kemampuan mendeteksi error secara lebih teliti dibandingkan bahasa

pemrograman lain. Java mempunyai *runtime-Exception handling* untuk membantu mengatasi error pada pemrograman.

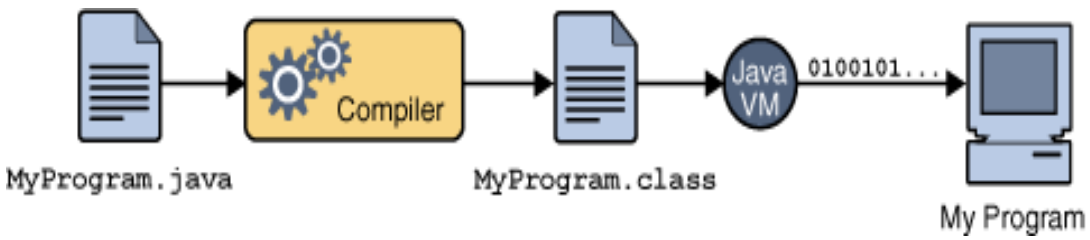
f. Aman
   Sebagai bahasa pemrograman untuk aplikasi internet dan terdistribusi, Java memiliki beberapa mekanisme keamanan untuk menjaga aplikasi tidak digunakan untuk merusak sistem komputer yang menjalankan aplikasi tersebut.
g. Architecture Neutral
   Program Java merupakan *platform independent*. Program cukup mempunyai satu buah versi yang dapat dijalankan pada platform yang berbeda dengan *Java Virtual Machine*.
h. Portabel
   Source code maupun program Java dapat dengan mudah dibawa ke platform yang berbeda-beda tanpa harus dikompilasi ulang.
i. Performance
   Performance pada Java sering dikatakan kurang tinggi. Namun performance Java dapat ditingkatkan menggunakan kompilasi Java lain seperti buatan Inprise, Microsoft ataupun Symantec yang menggunakan *Just In Time Compilers* (JIT).
j. Multithreaded
   Java mempunyai kemampuan untuk membuat suatu program yang dapat melakukan beberapa pekerjaan secara sekaligus dan simultan.
k. Dinamis
   Java didesain untuk dapat dijalankan pada lingkungan yang dinamis. Perubahan pada suatu *class* dengan menambahkan properties ataupun method dapat dilakukan tanpa menggangu program yang menggunakan *class* tersebut.

1.1.3 Sistem Kerja

Java merupakan bahasa pemrograman yang terdiri dari *compiler* dan *interpreter*, *compiler* menerjemahkan kode sumber program java menjadi *bytecode*. Untuk menjalankan *bytecode* hasil *compiler* diperlukan java *interpreter*, sehingga menjadikan java dapat dijalankan di berbagai platform. Java interpreter dapat dijalankan langsung dari *command prompt*; atau *applet viewer* atau *web browser* (untuk *applet*).

Kelemahan adalah kecepatan eksekusi program akan lebih lambat dari program biasa karena program *bytecode* harus diterjemahkan terlebih dahulu oleh *interpreter*, kemudian dijalankan pada *hardware*.

Gambar dibawah ini menjelaskan aliran proses kompilasi dan eksekusi sebuah program Java :



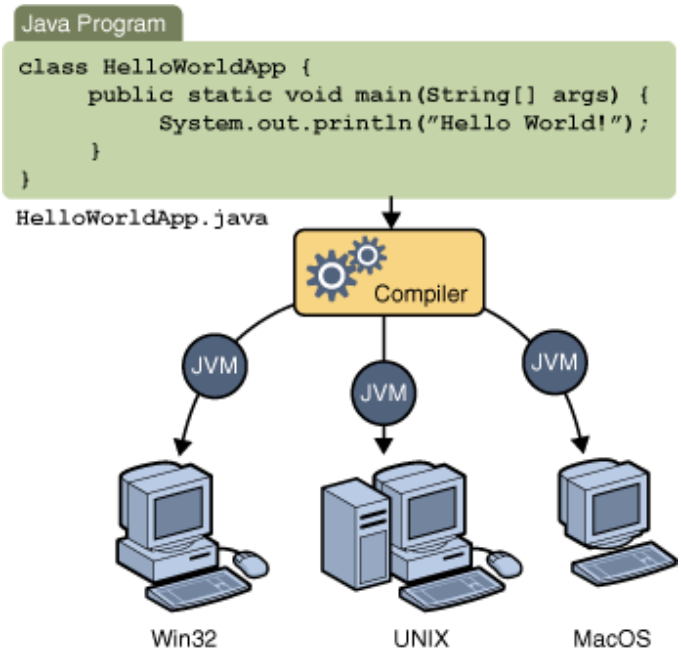
Gambar 1.3. Proses dari sebuah Program Java

Langkah pertama dalam membuat sebuah program berbasis Java adalah menuliskan kode program pada *text editor*. Contoh *text editor* yang dapat digunakan antara lain : notepad, vi, emacs dan lain sebagainya. Kode program yang dibuat kemudian disimpan dalam sebuah berkas berekstensi *.java*.

Setelah membuat dan menyimpan kode program, kompilasi file yang berisi kode program tersebut dengan menggunakan Java Compiler. Hasil dari kompilasi berupa berkas *bytecode* dengan ekstensi *.class*.

Berkas yang mengandung *bytecode* tersebut kemudian akan dikonversikan oleh *Java Interpreter* menjadi bahasa mesin sesuai dengan jenis dan *platform* yang digunakan.

Proses	Tool	Hasil
Menulis kode program	*Text editor*	Berkas berekstensi *.java*
Kompilasi program Java	Compiler	Berkas berekstensi *.class* (Java Bytecodes)
Menjalankan program Java	Interpreter	Program Output



Gambar 1.4. Java *Cross-platform* atau *Multi-platform*

1.1.4 Teknologi

Teknologi java merupakan sebuah bahasa pemrograman dan platform. Bahasa pemrograman java merupakan bahasa tingkat tinggi yang memiliki berbagai karakteristik seperti yang telah dijelaskan sebelumnya.

Platform adalah lingkungan perangkat keras atau lunak dimana program berjalan. Kita sudah mengenal beberapa platform yang paling populer seperti Microsoft Windows, Linux, Solaris OS, dan Mac OS. Kebanyakan platform dapat digambarkan sebagai kombinasi dari sistem operasi dan perangkat keras yang mendasarinya. Platform Java berbeda dari platform lainnya, karena hanya merupakan platform perangkat lunak yang berjalan di atas platform hardware lainnya.

Java platform terdiri dari dua komponen, yaitu *Java Virtual Machine* (JVM) dan *Aplication Programming Interface* (API).

Secara garis besan teknologi Java terbagai menjadi beberapa bagian yaitu seperti di bawah ini :

a. J2SE (*2 Platform Standard Edition*)
   Teknologi Java ini dirancang untuk membuat aplikasi yang berjalan pada PC dan *workstation* yang berada pada platform Windows, Linux, Macintos. J2SE terbagi menjadi dua bagian besar yaitu J2SE *Core* dan J2SE *Desktop*.
   J2SE Core digunakan untuk teknologi *security*, *debugging*, database dan sebagainya. Sedangkan teknologi J2SE Desktop meliputi beberapa teknologi yaitu JRE (*Java Runtime Environment*), JFC (*Java Foundation Classes*), *Java Sound* API dan sebagainya.
b. J2EE (*2 Platform Enterprise Edition*)
   Teknologi Java ini digunakan untuk pengembangan aplikasi-aplikasi *enterprise*, meliputi beberapa teknologi yaitu JSP (*Java Server Pages*), *Java Servlet*, CORBA (untuk aplikasi terdistribusi) dan sebagainya.
c. J2ME (Java *2 Platform Micro Edition*)
   Teknologi Java ini digunakan untuk pengembangan sistem mikro dan sistem *embedded* seperti *handphone*, PDA dan lain sebagainya. Meliputi dua bagian besar yaitu CLDC (MIDP, *BlueTooth* dan lainnya) dan CDC *Technology* (JDBC/teknologi database dan RMI).
d. *Java Web Services*
   Merupakan aplikasi web berbasis *enterprise* dengan standar XML dan protokol tertentu untuk bertukar data dengan klien. Teknologi ini meliputi beberapa API yang dirancang untuk bekerja dengan XML seperti *Java API for XML Based RPC* (JAX-RPC), *Java API for XML Based Messaging* (JAXM), *Java API for XML Processing* (JAXP) dan *Java API for XML Binding* (JAXB).

Penjelasan lebih lanjut tentang komponen-komponen yang ada di beberapa edisi Java adalah sebagai berikut :

A. Untuk tipe JSE (*Java Standard Edition*)
   - JavaBeans : salah satu arsitektur J2SE untuk aplikasi web. Komponen software yang dapat dimanipulasi secara visual pada sebuah tool visual.
   - JFC (*Java Foundation Classes*) : bagian dari library Java yang didasarkan pada platform Java untuk mengembangkan aplikasi GUI (*Graphical User Interface*). JFC dapat digunakan untuk grafik 2D, *image*, *format* dan *printing* teks dengan bantuan AWT (*Abstract Window Toolkit*), *Swing*, dan Java 2D. Dengan bantuan *input method framework*, teknologi JFC akan mempersiapkan aplikasi yang dapat diakses semua pengguna di seluruh dunia dengan bahasa yang berbeda. Fitur *drag* and *drop* akan menjadikan JFC dapat mendukung transfer data dengan aplikasi Java yang lain.
   - Java *Help* : *platform independent* dan salah satu fitur dalam sistem yang menawarkan fasilitas *help* secara otomatis. API JavaHelp 2.0 sangat berguna untuk membuat dokumentasi *online* dan memberikan informasi secara *online* pada para pengguna.
   - Java Web Start : salah satu *framework* dalam Java yang membantu memulai aplikasi secara langsung dari internet dengan menggunakan web browser. Seperti telah diketahui bahwa Java applet dapat berjalan di browser dan selain itu juga dapat berjalan di Java Web Start. Versi pertama dari Java Web

Start mulai dikenalkan pada tahun 2001. Mulai dengan *release* J2SE 1.4 Java Web Start disertakan dengan *Java Runtime Environment*, sehingga tidak perlu melakukan instalasi terpisah.

- JDBC (*Java Database Connectivity*) : merupakan API (*Application Programming Interface*) dan bagian dari *Java Standard Edition* yang digunakan untuk akses data dari database.
- JMF (*Java Media Framework*) : merupakan API yang menjadikan pengembang Java untuk memproses dan menambahkan sumber audio-video ke aplikasi Java dan applet. Fitur ini berguna untuk pengembangan multimedia untuk *capture*, *playback* dan *transcode* format media yang berbeda.

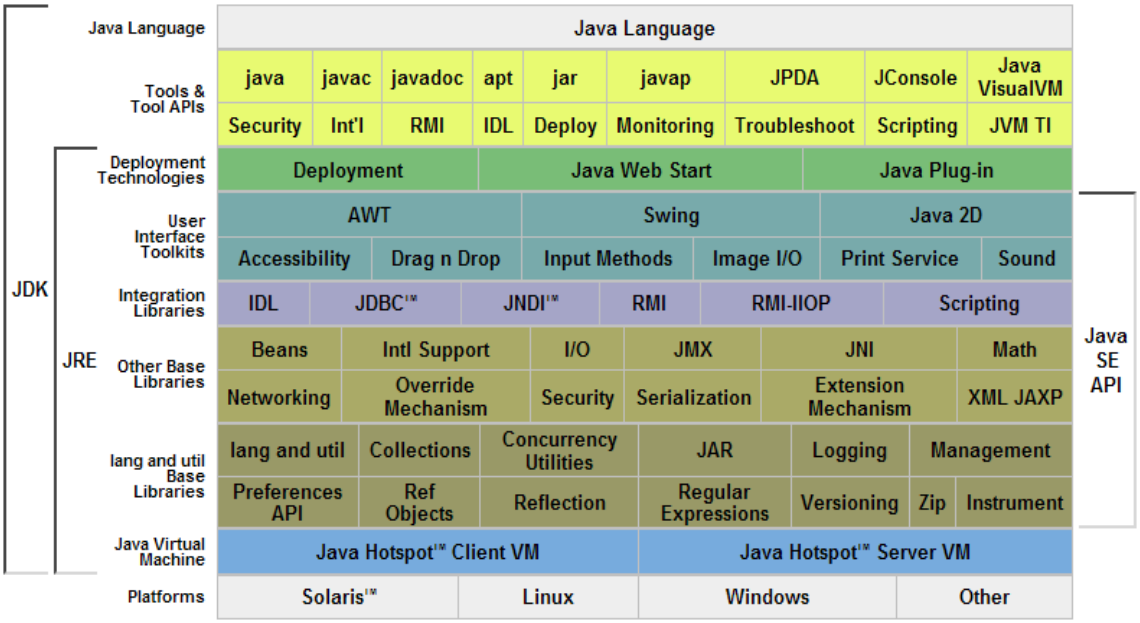
B. Untuk tipe JEE (*Java Enterprise Edition*)
   - EJB (*Enterprise JavaBeans*): platform Java untuk sisi server digunakan untuk aplikasi enterprise. Dengan menggunakan teknologi ini akan membantu membuat aplikasi yang kecil, mudah didistribusikan dan aman.
   - Java Mail : teknologi API Java Mail akan membantu membuat aplikasi mail dan pesan juga didukung platform dan protokol yang *independent*. Teknologi ini berada pada platform JSE dan JEE. Java Mail menggunakan platform ekstensibel untuk mentransfer semua jenis *Multimedia Internet Mail Extension* (MIME).
   - *Java Message Services* (JMS): digunakan untuk mengirim pesan antar pengguna aplikasi. Fasilitas ini merupakan tool untuk membuat aplikasi enterprise. JMS API merupakan kombinasi teknologi Java dengan tool pesan untuk membuat aplikasi berdasar pesan pendek. Ada dua model yang digunakan yaitu model *Point-to-Point* dan model *Publishing* and *Subscribing*.
   - *Java Server Pages* (JSP) : membantu para pengembang web dalam mengembangkan dan memperbaiki isi web dalam format seperti HTML dan XML. Dengan bantuan JSP, akan lebih mudah dalam membangun aplikasi web di server. Teknologi ini menggunakan tag HTML dan XML yang memberikan solusi logic untuk isi web. Terdapat pemisahan antara *interface* dan isi satu sama lain, sehingga akan mempermudah *designer* mengubah *layout* tanpa mengubah isi web secara keseluruhan.
   - Java Servlets : mempermudah pengembang menambah isi ke web server dengan menggunakan platform Java. Fitur ini menambah fungsi dari web server. Servlets menyediakan platform *independent* dan komponen berdasarkan aplikasi web tanpa adanya program CGI.

C. Jenis JME (*Java Mobile Edition*)
   - *Connected Limited Device Configuration* (CLDC) : konfigurasi dari *Java Micro Edition*. CLDC memberikan gambaran kemampuan JVM, inti dari kumpulan API untuk alat-alat seperti *pager* dan *mobile phone*. Ada dua versi CLDC yaitu versi 1.0 yang di keluarkan pada tahun 2000 yang dikenal sebagai *Java Specification Request* (JSR) 30. Versi selanjutnya adalah 1.1 atau JSR 139. CLDC dan MIDP (*Mobile Information Device Profile*) menyediakan aplikasi dengan platform Java berjalan pada peripheral dengan *power-supply* yang minim.
   - MIDP (*Mobile Information Device Profile*) : salah satu konfigurasi pada *Java Micro Edition* yang biasanya digabungkan dengan CLDC, menyediakan

lingkungan *java Runtime Environment* untuk variasi peralatan *mobile* dan PDA (*Personal Digital Assistance*). Dengan bantuan MIDP, pengembang Java dapat membuat aplikasi yang dapat di distribusikan secara *mobile* dalam waktu yang singkat. Fungsi yang mendukung antara lain *user interface*, koneksi jaringan data penyimpanan dan proses aplikasi secara keseluruhan. Ada dua versi MIDP yaitu yang pertama MIDP 2.0 atau JSR 118 dan yang kedua MIDP 1.0 atau JSR 37.

- CDC (*Connected Device Configuration*) : standar kerangka kerja dari teknologi Java yang digunakan untuk membangun dan mendistribusikan aplikasi Java yang dapat di-*sharing* dalam jaringan yang luas dan alat seperti *pager*, *mobile phones*, *set top box*, PDA. Ada dua versi yang tersedia yaitu yang pertama CDC 1.0 atau JSR 36 dan yang kedua CDC 1.1 (JSR 218).

1.1.5 Arsitektur



Gambar 1.5. Arsitektur J2SDK

Seperti gambar diatas, lapisan terbawah adalah system operasi yang menjalankan *Java Virtual Machine* (JVM), diatasnya terdapat *Core APIs* (*Application Programming Interfaces*) dan integration APIs yang menyediakan kumpulan library.

Lapisan berikutnya adalah *Swing* dan AWT (*Abstract Windowing Toolkit*) untuk membuat tampilan aplikasi yang bagus. Lapisan selanjutnya adalah utilitas untuk menjalankan aplikasi pada lingkungan desktop. Lapisan teratas adalah utilitas yang digunakan oleh para *programmer* untuk pengembangan, kompilasi, *debug* dan membuat dokumentasi program.

Beberapa fitur yang ditawarkan Java API (*Application Programming Interface*) antara lain sebagai berikut :

a. Applet

Program Java yang dapat berjalan di atas browser, yang dapat membuat halaman HTML lebih dinamis dan menarik.

b. *Java Networking*
   Sekumpulan API (*Application Programming Interface*) yang menyediakan fungsi–fungsi untuk aplikasi–aplikasi jaringan, seperti penyediaan akses untuk TCP, UDP, IP *Adrress* dan URL. Tetapi Java *Networking* tidak menyediakan akses untuk ICMP (*Internet Control Message Protocol*) dikarenakan alasan sekuriti dan pada kondisi umum hanya *administrator* (root) yang bisa memanfaatkan protokol ICMP.
c. *Java Database Connectivity* (JDBC)
   JDBC menyediakan sekumpulan API yang dapat digunakan untuk mengakses database seperti Oracle, MySQL, PostgreSQL, Microsoft SQL Server.
d. *Java Security*
   Java Security menyediakan sekumpulan API untuk mengatur *security* dari aplikasi Java baik secara *high level* atau *low level*, seperti *public/private key management* dan *certificates*.
e. Java Swing
   Java Swing menyediakan sekumpulan API untuk membangun aplikasi–aplikasi GUI (*Graphical User Interface*) dan model GUI yang diinginkan bisa bermacam–macam, bisa model Java, model Motif/CDE atau model yang *dependent* terhadap *platform* yang digunakan.
f. Java RMI
   Java RMI (*Remote Method Invocation*) menyediakan sekumpulan API untuk membangun aplikasi–aplikasi Java yang mirip dengan model RPC (*Remote Procedure Call*), jadi objek-objek java bisa dipanggil secara remote pada jaringan.
g. Java 2D/3D
   Java 2D/3D menyediakan sekumpulan API untuk membangun grafik – grafik 2D/3D yang menarik dan juga akses ke printer.
h. *Java Server Pages*
   Berkembang dari Java Servlet yang digunakan untuk menggantikan aplikasi–aplikasi CGI, JSP (Java Server Pages) yang mirip ASP dan PHP merupakan alternatif terbaik untuk solusi aplikasi Internet.
i. JNI (*Java Native Interface*)
   JNI menyediakan sekumpulan API yang digunakan untuk mengakses fungsi–fungsi pada library (*.dll atau *.so) yang dibuat dengan bahasa pemrograman yang lain seperti C, C++, dan Basic.
j. *Java Sound*
   Java Sound menyediakan sekumpulan API untuk manipulasi sound.
k. Java IDL + CORBA
   Java IDL (*Interface Definition Language*) menyediakan dukungan Java untuk implementasi CORBA (*Common Object Request Broker*) yang merupakan model *distributed-Object* untuk solusi aplikasi besar di dunia *networking*.
l. *Java Card*
   Java Card utamanya digunakan untuk aplikasi–aplikasi pada smart card, yang sederhana wujudnya seperti SIM Card pada *handphone*.
m. JTAPI (*Java Telephony* API)
   Java Telepony API menyediakan sekumpulan API untuk memanfaatkan *devices–devices telepony*, sehingga akan cocok untuk aplikasi–aplikasi CTI (*Computer Telephony Integration*) yang dibutuhkan seperti ACD (*Automatic Call Distribution*), PCPBX dan lainnya.

1.1.6 Sebagian Fitur di Java

Beberapa fitur penting yang dimiliki oleh java antara lain :

a. *Java Virtual Machine* (JVM)

JVM adalah sebuah mesin imajiner (maya) yang bekerja dengan menyerupai aplikasi pada sebuah mesin nyata. JVM menyediakan spesifikasi hardware dan platform dimana kompilasi kode Java terjadi. Spesifikasi inilah yang membuat aplikasi berbasis Java menjadi bebas dari *platform* manapun karena proses kompilasi diselesaikan oleh JVM.

Aplikasi program Java diciptakan dengan *file* teks berekstensi *.java*. Program ini dikompilasi menghasilkan satu berkas *bytecode* berekstensi *.class* atau lebih. *Bytecode* adalah serangkaian instruksi serupa instruksi kode mesin. Perbedaannya adalah kode mesin harus dijalankan pada sistem komputer dimana kompilasi ditujukan, sementara *bytecode* berjalan pada *java interpreter* yang tersedia di semua *platform* sistem komputer dan sistem operasi.

b. *Garbage Collection*

Banyak bahasa pemrogaman lain yang mengijinkan seorang programmer mengalokasikan memori pada saat dijalankan. Namun, setelah menggunakan alokasi memori tersebut, harus terdapat cara untuk menempatkan kembali blok memori tersebut supaya program lain dapat menggunakannya. Dalam C, C++ dan bahasa lainnya, adalah programmer yang mutlak bertanggung jawab akan hal ini. Hal ini dapat menyulitkan bilamana programmer tersebut alpa untuk mengembalikan blok memori sehingga menyebabkan situasi yang dikenal dengan nama *memory leaks.*

Program Java melakukan *garbage collection* yang berarti program tidak perlu menghapus sendiri objek–objek yang tidak digunakan lagi. Fasilitas ini mengurangi beban pengelolaan memori oleh programmer dan mengurangi atau mengeliminasi sumber kesalahan terbesar yang terdapat pada bahasa yang memungkinkan alokasi dinamis.

c. *Code Security*

Code Security terimplementasi pada Java melalui penggunaan Java Runtime Environment (JRE). Java menggunakan model pengamanan 3 lapis untuk melindungi sistem dari *untrusted Java Code*.

- Pertama, *class-loader* menangani pemuatan kelas Java ke *runtime interpreter.* Proses ini menyediakan pengamanan dengan memisahkan kelas–kelas yang berasal dari *local disk* dengan kelas–kelas yang diambil dari jaringan. Hal ini membatasi aplikasi Trojan karena kelas–kelas yang berasal dari *local disk* yang dimuat terlebih dahulu.
- Kedua, *bytecode verifier* membaca *bytecode* sebelum dijalankan dan menjamin *bytecode* memenuhi aturan–aturan dasar bahasa Java.
- Ketiga, manajemen keamanan menangani keamanan tingkat aplikasi dengan mengendalikan apakah program berhak mengakses sumber daya seperti sistem file, *port* jaringan, proses eksternal dan sistem *windowing*.

Setelah seluruh proses tersebut selesai dijalankan, barulah kode program di eksekusi.

Java juga menyediakan beragam teknik pengamanan lain :

- Bahasa dirancang untuk mempersulit eksekusi kode perusak. Peniadaan *pointer* merupakan langkah besar pengamanan. Java tidak mengenal operasi *pointer*. Di tangan programmer handal, operasi pointer merupakan hal yang luar biasa untuk optimasi dan pembuatan program yang efisien serta mengagumkan. Namun mode ini dapat menjadi petaka di hadapan programmer jahat. Pointer merupakan sarana luar biasa untuk pengaksesan tak diotorisasi. Dengan peniadaan operasi *pointer*, Java dapat menjadi bahasa yang lebih aman.
- Java memiliki beberapa pengaman terhadap *applet*. Untuk mencegah program bertindak mengganggu media penyimpanan, maka *applet* tidak diperbolehkan melakukan *open, read* ataupun *write* terhadap berkas secara sembarangan. Karena Java *applet* dapat membuka jendela *browser* yang baru, maka jendela mempunyai logo Java dan teks identifikasi terhadap jendela yang dibuka. Hal ini mencegah jendela *pop-up* menipu sebagai permintaan keterangan *username* dan *password*.

1.2 Persiapan Pemrograman Java

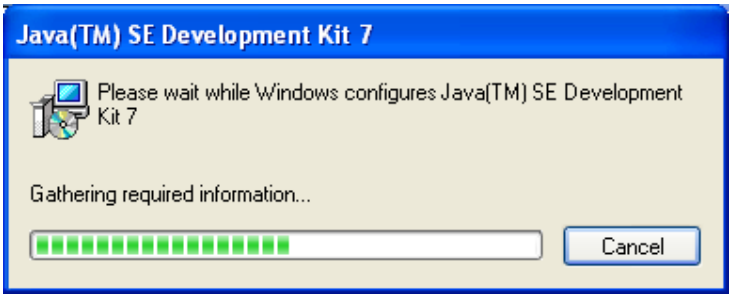
Untuk mengembangkan program java kita harus menginstall aplikasi yang dibutuhkan terlebih dahulu. Semua aplikasi pengembangan yang dibutuhkan telah dijadikan suatu paket yang disebut *Java Development Kit* (JDK). *Java Development Kit* (JDK) dapat di-*download* di web site Oracle, yaitu http://www.oracle.com/.

JDK terdiri dari beberapa komponen, antara lain :

a. Kompilator/*compiler* (javac)
b. *Intrepreter* program java (java)
c. *Applet viewer* (appletviewer)
d. *Debugger* (jdb)
e. *Class file disassembler* (javap)
f. *Header and stub file generator* (javah)
g. *Documentation generator* (javadoc)
h. *Applet demo*
i. Kode sumber java API

1.2.1 Instalasi JDK

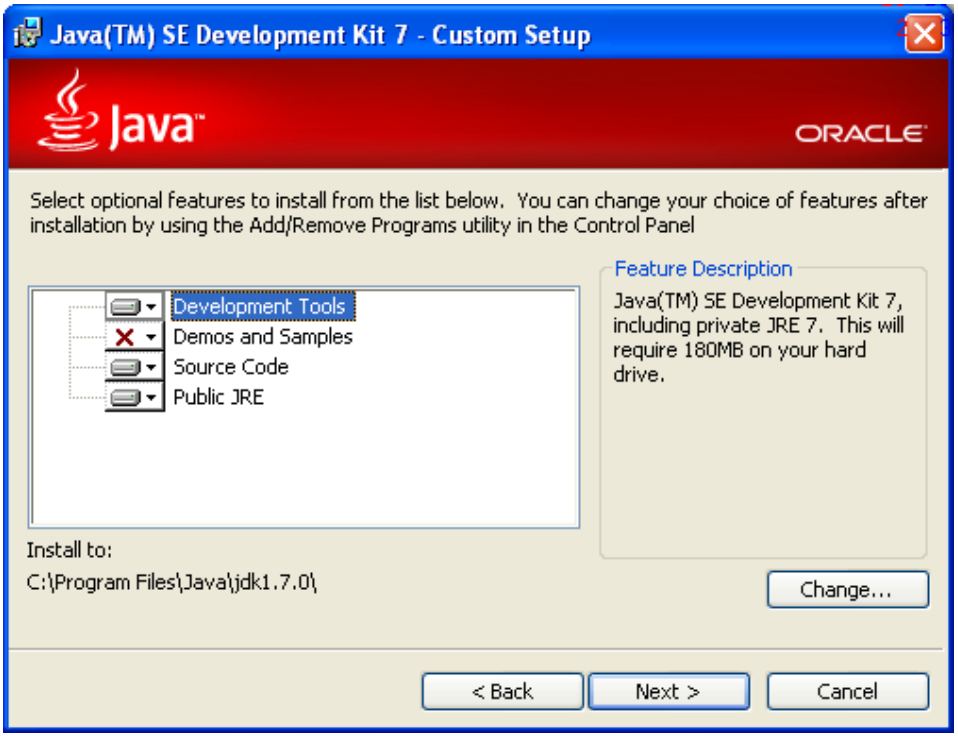
Pada contoh instalasi JDK ini menggunakan versi 7, setelah kita selesai *download* maka akan mendapat file jdk-7-windows-i586.exe. Jalankan file tersebut, kemudian akan menampilkan proses instalasi sebagai berikut :



Gambar 1.6. Instalasi JDK 7

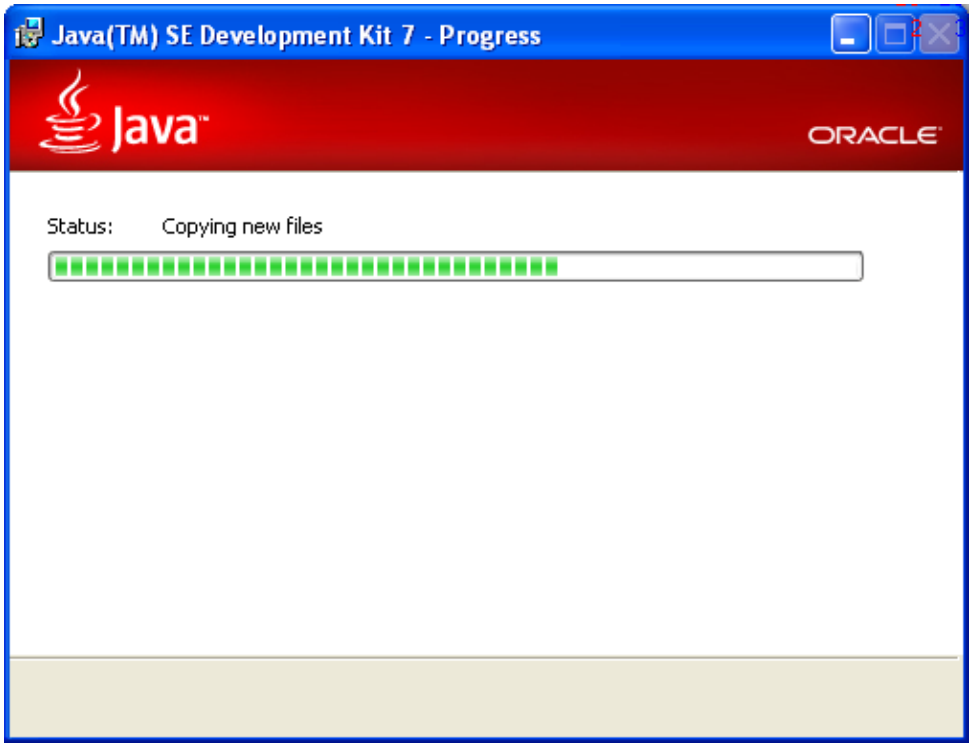
Gambar 1.7. Kotak dialog memulai instalasi JDK 7

Pilih Next > untuk memulai instalasi JDK 7.

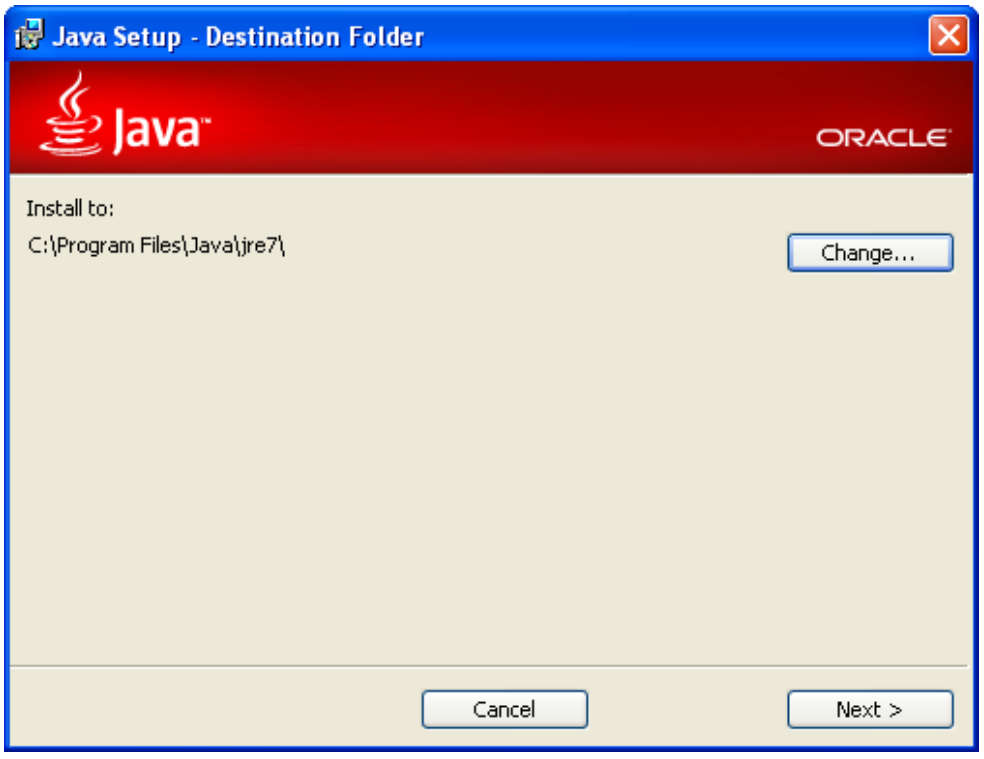


Gambar 1.8. Kotak dialog *Custom Setup* instalasi JDK 7

Pilih fitur yang akan di-*install* sesuai kebutuhan dan tentukan tujuan hasil instalasi, kemudian pilih tombol Next >.



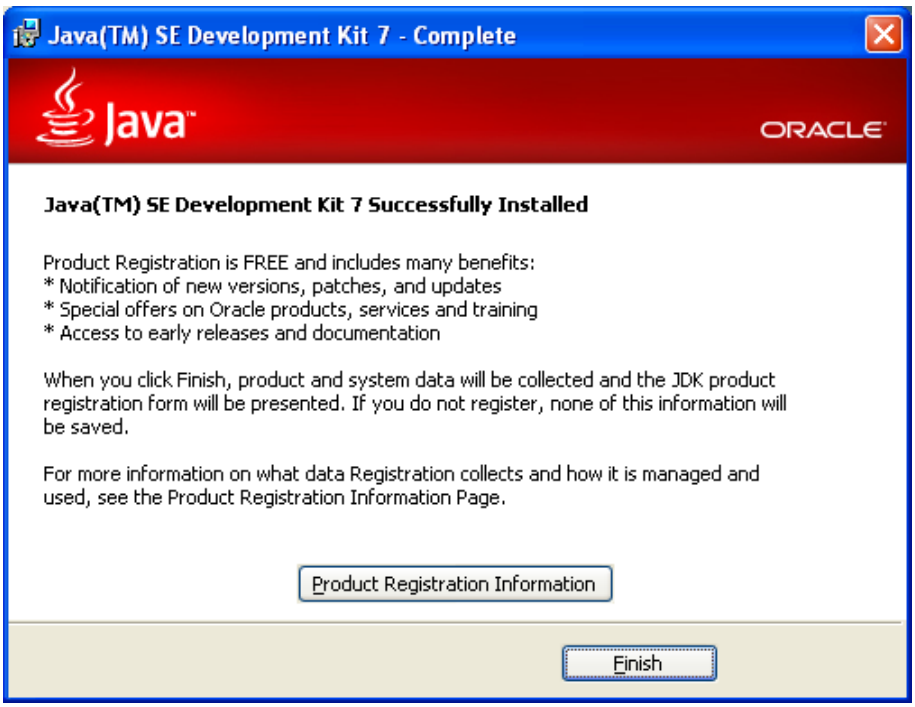
Gambar 1.9. Kotak dialog *progress* instalasi JDK 7



Gambar 1.10. Menentukan tujuan instalasi JRE 7

Pada saat menginstall JDK, kita juga harus menginstall JRE (*Java Runtime Environment*) yang akan kita butuhkan ketika menjalankan program yang kita buat. Tentukan tujuan instalasi JRE 7, kemudin pilih tombol Next >.

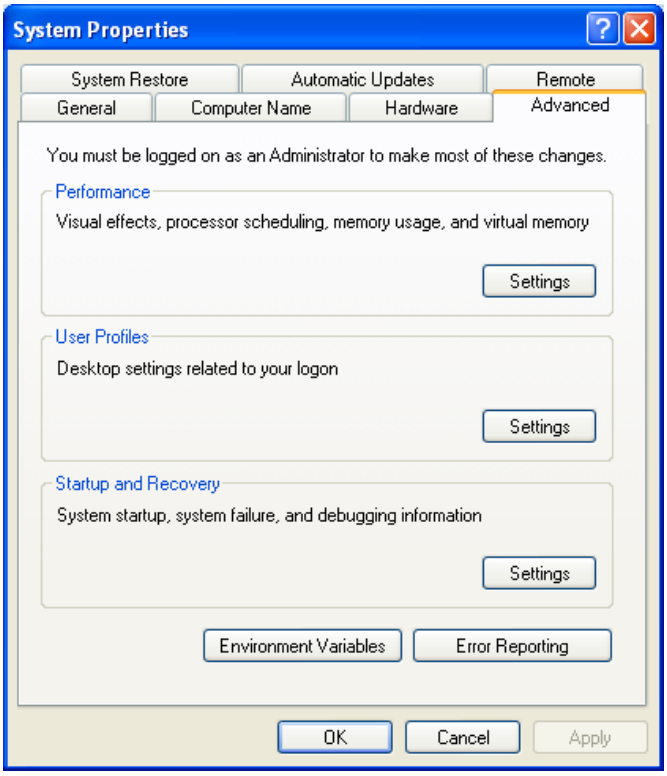
Gambar 1.11. Kotak dialog *progress* instalasi JRE 7



Gambar 1.12. Tampilan akhir instalasi JDK 7

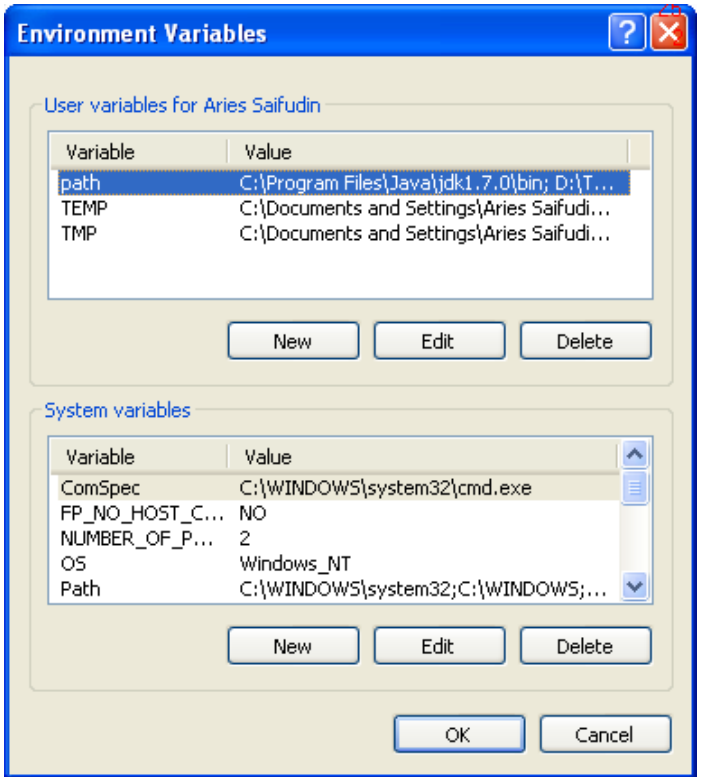
Tekan tombol Finish untuk mengakhiri proses instalasi.

Agar file-file JDK hasil instalasi dapat dipanggil dari *folder* mana saja maka *setting Path* harus diarahkan ke *folder* JDK-nya. Untuk windows XP, klik menu *Start*, pilih *Control Panel*, pilih *System*, pilih Tab *Advanced*, sehingga tampil seperti gambar 1.13 berikut ini :



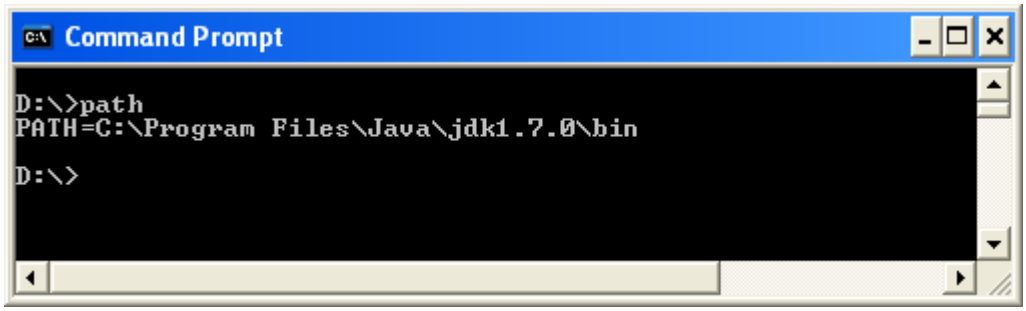
Gambar 1.13. Tampilan *System Properties*

Selanjutnya klik tombol *Envirentment Variables* dan muncul tampilan seperti gambar 1.14 berikut ini :



Gambar 1.14. Tampilan *Environment Variables*

Klik Tombol *New* dan isikan PATH pada *Variable Name* dan folder BIN dari JDK yang telah diinstall (C:\Program Files\Java\jdk1.7.0\bin) pada *Variable Value*. Setelah selesai jalankan *command promt* dan ketik PATH, jika folder JDK sudah tampil maka *setting path* selesai. Jika belum tampil, *restart* komputernya terlebih dulu.

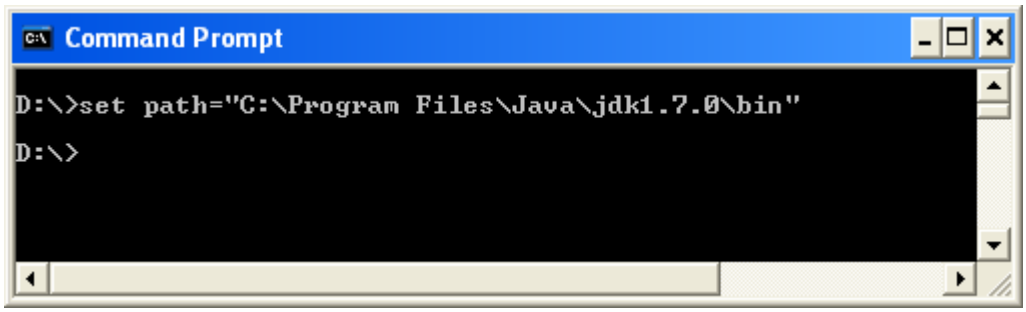


Gambar 1.15. Tampilan pengecekan *path* dari *command prompt*

Jika ingin melakukan *setting path* dari *command prompt*, dapat menggunakan pernyataan berikut ini :

```
set path=”C:\Program Files\Java\jdk1.7.0\bin”
```

atau dapat dilihat pada gambar 1.16 berikut ini :



Gambar 1.16. Tampilan *setting path* dari *command prompt*

1.2.2 Kompilator

Untuk mengkompilasi kode sumber java menjadi kelas *bytecode* menggunakan program *javac.exe*. Kode sumber java berekstensi *.java*, sedangkan kompilator *javac* menghasilkan file *bytecode* dengan ekstensi *.class*. Kompilator menghasilkan satu file *class* untuk tiap kelas yang didefinisikan di file sumber. Sehingga dimungkinkan dalam satu file sumber java ketika dikompilasi menghasilkan lebih dari satu file *class*.

Sintaks untuk menggunakan kompilator adalah :

```
javac [Options] FileName.java
```

- *FileName* adalah nama file kode sumber yang akan dikompilasi.
- *Options* bersifat opsional, digunakan untuk menspesifikasikan cara kompilator memproses kode sumber program java.

1.2.3 Interpreter

Interpreter merupakan modul utama sistem java yang digunakan aplikasi java. *Interpreter* digunakan untuk menjalankan program *bytecode* java. *Interpreter* bertindak sebagai *tool* baris perintah untuk menjalankan program java non-grafis. Untuk program grafis memerlukan tampilan yang didukung *browser* dan sistem operasi.

Sintaks untuk menggunakan *interpreter* adalah :

```
java [Options] ClassName [Arguments]
```

- *ClassName* adalah nama kelas yang ingin dieksekusi. Ketika *interpreter* mengeksekusi suatu kelas, maka yang dilakukan adalah mengeksekusi metode *main*() di dalam kelas itu. *Interpreter* selesai ketika metode *main*() dan *thread-thread* yang diciptakan telah selesai dieksekusi.
- *Options* bersifat opsional, digunakan untuk menspesifikasikan cara *interpreter* mengeksekusi program java.
- *Arguments* bersifat opsional, digunakan untuk mengirimkan data *string* yang dapat diproses dalam program java.

1.2.4 Applet Viewer

Applet viewer adalah *tool* yang digunakan untuk pengujian *java applet* secara minimal.

Sintaks untuk menggunakan *applet viewer* adalah :

```
appletviewer [Options] URL
```

- *Options* bersifat opsional, digunakan untuk menspesifikasikan cara menjalankan *java applet.* Hanya ada satu *option* yang didukung *applet viewer*, yaitu *–debug* yang menyatakan menjalankan *applet viewer* di java *debugger* yang memungkinkan melakukan *debugging* terhadap *java applet*.
- *URL* digunakan untuk menspesifikasikan dokumen URL berisi halaman HTML dengan *embedded java applet*.

1.2.5 Java Debugger

Java debugger (jdb) adalah utilitas baris perintah untuk melakukan *debugging* aplikasi java.

Sintaks untuk menggunakan *Java debugger* adalah :

```
jdb [options] [class] [arguments]
```

- *Options* digunakan untuk menspesifikasikan cara menjalankan jdb.
- *ClassName* adalah nama kelas yang ingin di-*debug*.
- *Arguments* digunakan untuk mengirimkan data *string* yang dapat diproses di dalam program java.

1.2.6 Java Class File Disassembler

Java Class File Disassembler (javap) digunakan untuk menguraikan (*disassemble*) file *class*. Keluaran *default* aktivitas *disassemble* berisi daftar dari data publik (*public data*)

dan metode publik (*public method*) di kelas. *Class file disassemble* berguna ketika kita tidak mempunyai kode sumber dari kelas. Dengan demikian, kita dapat mengetahui data dan metode publik sehingga kita dapat menggunakannya.

Sintaks untuk menggunakan *Java Class File Disassembler* adalah :

```
javap [Options] ClassNames
```

- *Options* digunakan untuk menspesifikasikan cara menjalankan *disassemble*.
- *ClassNames* adalah nama satu atau lebih kelas yang di-*disassemble*.

1.2.7 Java Header and Stub File Generator

Java Header and Stub File Generator (javah) untuk menghasilkan C header dan file kode sumber untuk implementasi metode-metode java dalam bahasa C. File-file yang dihasilkan dapat digunakan untuk mengakses variable anggota obyek yang ditulis dengan bahasa C. *Java Header and Stub File Generator* menghasilkan struktur C dengan *layout* kelas java.

Sintaks untuk *Java Header and Stub File Generator* adalah :

```
javah [Options] ClassName
```

- *Options* digunakan untuk menspesifikasikan cara file-file sumber dihasilkan.
- *ClassName* adalah nama kelas yang perlu dihasilkan file-file sumber bahasa C.

1.2.8 Java Documentation Generator

Java Documentation Generator (javadoc) adalah *tool* untuk menghasilkan dokumentasi API secara langsung dari kode sumber java. *Java Documentation Generator* melakukan parsing terhadap file sumber java dan menghasilkan halaman HTML berdasarkan deklarasi dan komentar di file sumber.

Sintaks untuk menggunakan *Java Documentation Generator* adalah :

```
javadoc Options FileName
```

- *Options* memungkinkan kita mengubah perilaku *default* dari *javadoc*.
- *FileName* digunakan untuk menspesifikasikan paket atau file kode sumber java.

1.2.9 Demo

JDK juga berisi beragam contoh program java, semuanya disertai kode sumber.

1.2.10 Kode Sumber Java API

JDK disertai kode sumber secara lengkap untuk semua kelas yang membentuk java API. Kode sumber API secara otomatis tersimpan di *hard drive* saat melakukan dekompresi JDK, tetapi masih dalam bentuk file yang dikompresi, karena diasumsikan tidak semua orang menggunakannya. Kode sumber API terdapat di file bernama *src.zip*, file tersebut disimpan di direktori java yang diciptakan di *harddisk* ketika instalasi JDK.

1.3 Tes Hasil Instalasi JDK

Setelah kita selesai melakukan instalasi JDK, kita bisa membuat dan menjalankan program java. Kita dapat mencoba dengan beberapa program sederhana di bawah ini.

1.3.1 Program Mode Teks

Untuk melakukan tes hasil instalasi JDK, kita buat program sederhana sebagai berikut :

TestHello.java

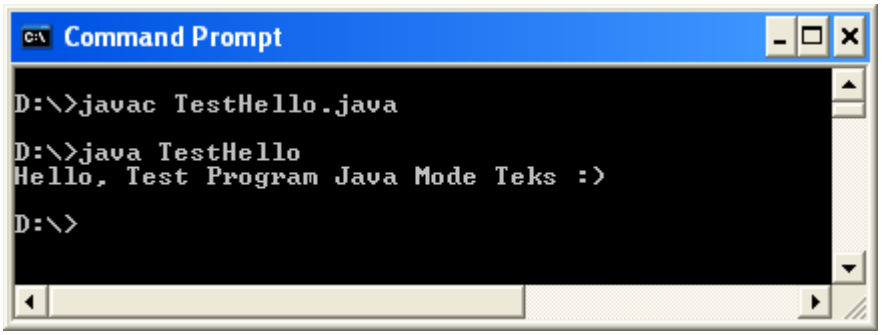
```
public class TestHello {
  public static void main (String args[]) {
    System.out.println("Hello, Test Program Java Mode Teks :)");
  }
}
```

Untuk editornya dapat menggunakan notepad, vi atau text editor lainnya, kemudian simpan dengan nama `TestHello.java` (huruf besar/kecil harus sama dengan yang ditulis pada nama class). Untuk mengkompilasi gunakan perintah :

```
Javac TestHello.java
```

Jika tidak ada kesalahan, berarti aplikasi JDK sudah terinstall dan sintak dalam program tidak ada kesalahan. Untuk menjalankan gunakan perintah :

```
Java TestHello
```



Gambar 1.17. Mengkompilasi dan menjalankan program TestHello.java

1.3.2 Program Mode GUI

Untuk melakukan tes program GUI (*Graphical User Interface*), dapat dilakukan dengan program sederhana berikut ini :

TestGUI.java

```
import javax.swing.*;
class TestGUI{
  public static void main (String args[]){
    JOptionPane.showMessageDialog(null, "Membuat program java sangat
mudah !   :)");
  }
}
```

Gambar 1.18. Mengkompilasi dan menjalankan program TestGUI.java

1.3.3 Program Java Applet

Tes java applet dilakukan dengan membuat program java applet sebagai berikut :

TestApplet.java

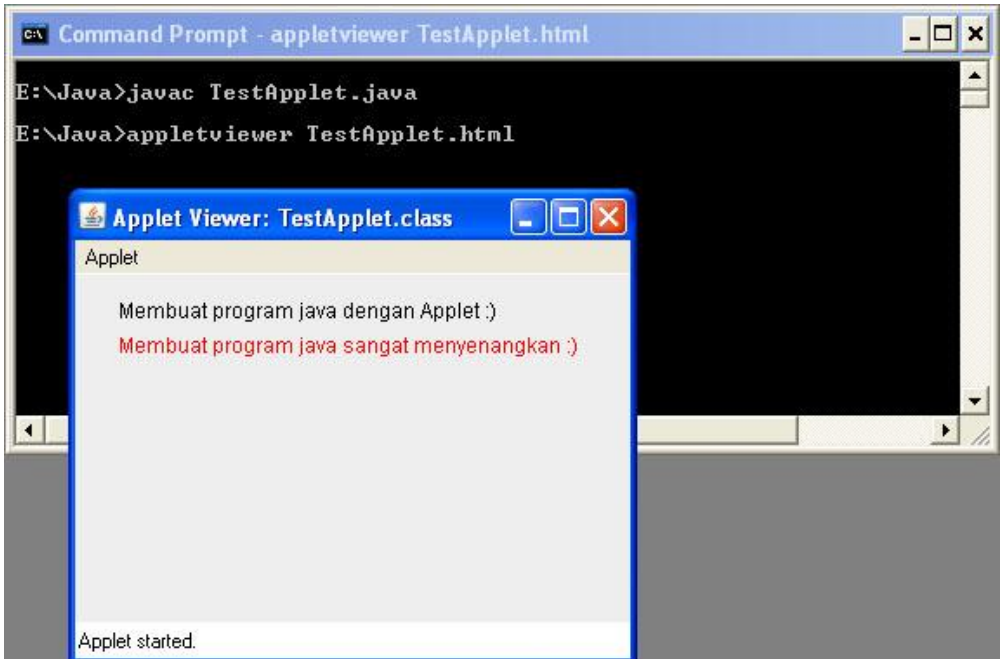
```
import java.awt.Graphics;
import javax.swing.JApplet;
import java.awt.Color;

public class TestApplet extends JApplet {
  public void paint (Graphics g) {
    super.paint(g);
    g.drawString("Membuat program java dengan Applet :)",25,25);
    g.setColor(Color.red);
        g.drawString("Membuat program java sangat menyenangkan
:)",25,45);
   }
}
```

Kemudian buat file HTML untuk memproses program java applet yang telah dibuat :

TestApplet.html

```
<html>
   <p> Ini adalah file html yang disisipi Applet Java </p>
   <applet code="TestApplet.class" height=200 width=320>
     No Java?!
   </applet>
 </html>
```

Setelah selesai disimpan, *compile* dan jalankan dengan *appletviewer* :



Gambar 1.19. Mengkompilasi dan menjalankan program TestApplet.java

Atau dengan membuka file HTML menggunakan browser :

Gambar 1.20. Menjalankan program applet dari browser

Referensi :

1. Hariyanto, Bambang, (2007), *Esensi-esensi Bahasa Pemrograman Java*, Edisi 2, Informatika Bandung, November 2007.
2. Utomo, Eko Priyo, (2009), *Panduan Mudah Mengenal Bahasa Java*, Yrama Widya, Juni 2009.
3. Tim Pengembang JENI, JENI 1-6, Depdiknas, 2007
4. http://www.java.net/, diakses tanggal 28 September 2010
5. http://www.oracle.com/, diakses tanggal 28 September 2010