BAB 2

Pengenalan Bahasa JAVA

2.1 Tujuan

Pada bab ini akan dibahas secara singkat tentang sejarah JAVA dan definisi teknologi JAVA. Bab ini juga akan sedikit menyinggung tentang fase – fase dalam program JAVA.

Pada akhir pembahasan, diharapkan pembaca dapat :

1. Menjelaskan fitur – fitur teknologi dari Java meliputi Java Virtual Machine (JVM), *garbage collection*, dan *code security*.
2. Menjelaskan perbedaan fase pada pemrograman JAVA

2.2 Latar Belakang JAVA

2.2.1 Sejarah Singkat JAVA

Pada 1991, sekelompok insinyur Sun dipimpin oleh Patrick Naughton dan James Gosling ingin merancang bahasa komputer untuk perangkat konsumer seperti *cable TV Box*. Karena perangkat tersebut tidak memiliki banyak memori, bahasa harus berukuran kecil dan mengandung kode yang liat. Juga karena manufaktur – manufaktur berbeda memilih *processor* yang berbeda pula, maka bahasa harus bebas dari manufaktur manapun. Proyek diberi nama kode ”Green”.

Kebutuhan untuk fleksibilitas, kecil, liat dan kode yang netral terhadap *platform* mengantar tim mempelajari implementasi Pascal yang pernah dicoba. Niklaus Wirth, pencipta bahasa Pascal telah merancang bahasa portabel yang menghasilkan *intermediate code* untuk mesin hipotesis. Mesin ini sering disebut dengan mesin maya *(virtual machine)*. Kode ini kemudian dapat digunakan di sembarang mesin yang memiliki *interpreter*. Proyek Green menggunakan mesin maya untuk mengatasi isu utama tentang netral terhadap arsitektur mesin.

Karena orang – orang di proyek Green berbasis C++ dan bukan Pascal maka kebanyakan sintaks diambil dari C++, serta mengadopsi orientasi objek dan bukan prosedural. Mulanya bahasa yang diciptakan diberi nama ”Oak” oleh James Gosling yang mendapat inspirasi dari sebuah pohon yang berada pada seberang kantornya, namun dikarenakan nama Oak sendiri merupakan nama bahasa pemrograman yang telah ada sebelumnya, kemudian SUN menggantinya dengan JAVA. Nama JAVA sendiri terinspirasi pada saat mereka sedang menikmati secangkir kopi di sebuah kedai kopi yang kemudian dengan tidak sengaja salah satu dari mereka menyebutkan kata JAVA yang mengandung arti asal bijih kopi. Akhirnya mereka sepakat untuk memberikan nama bahasa pemrograman tersebut dengan nama Java.

Produk pertama proyek Green adalah Star 7 (*7), sebuah kendali jarak jauh yang sangat cerdas. Dikarenakan pasar masih belum tertarik dengan produk konsumer cerdas maka proyek Green harus menemukan pasar lain dari teknologi yang diciptakan. Pada saat yang sama, implementasi WWW dan Internet sedang mengalami perkembangan pesat. Di lain pihak, anggota dari proyek Green juga menyadari bahwa Java dapat digunakan pada pemrograman internet, sehingga penerapan selanjutnya mengarah menjadi teknologi yang berperan di web.

Bahasa/Alat pengembangan	Arsitektur Program			
	Modul Web Server	Scripting Web Server	Modul Web Browser	Scripting Web Browser
Java	servlet	JSP	Applet	Javascript
C++	CGI exe		ActiveX*	
Perl	CGI script			
Phyton	CGI script			
PHP		PHP script		
Visual Basic		ASP*	ActiveX*	VB Script*

*) Hanya di landasan Windows, tidak bisa di Linux.

Java telah mengakomodasi hampir seluruh fitur penting bahasa – bahasa pemrograman yang ada semenjak perkembangan komputasi modern manusia :

1. Dari SIMULA, bahasa pada tahun 65-an, bahasa yang paling mempengaruhi Java sekaligus C++. Dari bahasa ini diadopsi bentukan – bentukan dasar dari pemrograman berorientasi objek.
2. Dari LISP – bahasa tahun 55-an. Diadopsi fasilitas *garbage collection,* serta kemampuan untuk meniru *generic list processing*, meski fasilitas ini jarang yang memanfaatkannya.
3. Dari Algol – bahasa pada tahun 60-an, diambil struktur kendali yang dimilikinya.
4. Dari C++, diadopsi sintaks, sebagian semantiks dan *exception handling*
5. Dari bahasa Ada, diambil *strongly type,* dan *exception handling*.
6. Dari Objective C, diambil fasilitas interface.
7. Dari bahasa SmallTalk, diambil pendekatan *single-root class hiérarchie,* dimana objek adalah satu kesatuan hirarki pewarisan
8. Dari bahasa Eiffel, fasilitas *assertion* yang mulai diterapkan di sebagian JDK 1.4

2.2.2 Apa itu Teknologi JAVA?

2.2.2.1 Sebuah Bahasa Pemrograman

Sebagai sebuah bahasa pemrograman, Java dapat membuat seluruh bentuk aplikasi, *desktop, web* dan lainnya, sebagaimana dibuat dengan menggunakan bahasa pemrograman konvensional yang lain.

Java adalah bahasa pemrograman yang berorientasi objek (OOP) dan dapat dijalankan pada berbagai *platform* sistem operasi. Perkembangan Java tidak hanya terfokus oada satu sistem operasi, tetapi dikembangkan untuk berbagai sistem operasi dan bersifat *open source.*

2.2.2.2 Sebuah *Development Environment*

Sebagai sebuah peralatan pembangun, teknologi Java menyediakan banyak *tools* : *compiler, interpreter,* penyusun dokumentasi, paket kelas dan sebagainya.

2.2.2.3 Sebuah Aplikasi

Aplikasi dengan teknologi Java secara umum adalah aplikasi serbt a guna yang dapat dijalankan pada seluruh mesin yang memiliki *Java Runtime Environment* (JRE).

2.2.2.4 Sebuah *Deployment Environment*

Terdapat dua komponen utama dari *Deployment Environment*. Yang pertama adalah JRE, yang terdapat pada paket J2SDK, mengandung kelas – kelas untuk semua paket teknologi Java yang meliputi kelas dasar dari Java, komponen GUI dan sebagainya. Komponen yang lain terdapat pada Web Browser. Hampir seluruh Web Browser komersial menyediakan *interpreter* dan *runtime environment* dari teknologi Java.

2.2.5 Mengapa Mempelajari JAVA?

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

1. Sederhana (*Simple*)
   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*.

2. 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.

3. Terdistribusi (*Distributed*)
   Java dibuat untuk membuat aplikasi terdistribusi secara mudah dengan adanya *libraries* networking yang terintegrasi pada Java.

4. Interpreted
   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.

5. 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.

6. Secure
   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.

7. Architecture Neutral
   Program Java merupakan *platform independent*. Program cukup mempunyai satu buah versi yang dapat dijalankan pada platform berbeda dengan *Java Virtual Machine*.

8. Portable
   Source code maupun program Java dapat dengan mudah dibawa ke platform yang berbeda-beda tanpa harus dikompilasi ulang.

9. 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).

10. Multithreaded
    Java mempunyai kemampuan untuk membuat suatu program yang dapat melakukan beberapa pekerjaan secara sekaligus dan simultan.

11. Dynamic
    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.

2.2.4 Sebagian Fitur dari JAVA

2.2.4.1 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.

2.2.4.2 *Garbage Collection*

Banyak bahasa pemrogaman lain yang mengijinkan seorang pemrogram 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 pemrogram yang mutlak bertanggung jawab akan hal ini. Hal ini dapat menyulitkan bilamana pemrogram 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 pemrogram dan mengurangi atau mengeliminasi sumber kesalahan terbesar yang terdapat pada bahasa yang memungkinkan alokasi dinamis.

2.2.4.3 *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*.

1. 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.
2. Kedua, *bytecode verifier* membaca *bytecode* sebelum dijalankan dan menjamin *bytecode* memenuhi aturan – aturan dasar bahasa Java.
3. 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*.

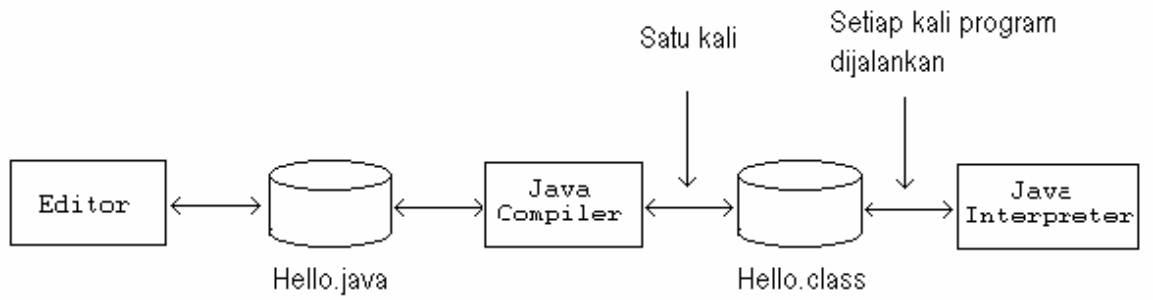
Setelah seluruh proses tersebut selesai dijalankan, barulah kode program di eksekusi.

Java juga menyediakan beragam teknik pengamanan lain :

1. Bahasa dirancang untuk mempersulit eksekusi kode perusak. Peniadaan *pointer* merupakan langkah besar pengamanan. Java tidak mengenal operasi *pointer*. Di tangan pemrogram 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 pemrogram jahat. Pointer merupakan sarana luar biasa untuk pengaksesan tak diotorisasi. Dengan peniadaan operasi *pointer*, Java dapat menjadi bahasa yang lebih aman.
2. 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*.

2.2.5 Fase – fase Pemrograman JAVA

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



Langkah pertama dalam pembuatan 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 tersimpan dalam sebuah berkas berekstensi *.java.*

Setelah membuat dan menyimpan kode program, kompilasi file yang berisi kode program tersebut dengan menggunakan Java Compiler. Hasil dari adalah 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