|  |
| --- |
| BAB 1 PENGENALAN BAHASA JAVA |

I.Tujuan Instruksional Khusus

Setelah menyelesaikan materi ini mahasiswa 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

# II. Persyaratan / Kebutuhan Sistem Komputer

# Persyaratan / kebutuhan sistem untuk menjalankan program Java adalah sebagai berikut :

# Processor P-II atau yang lebih tinggi,jhjhjhjhj

# Ram minimal 32 Mbyte,

# Free space harddisk minimal 200 Mbyte,

# Mouse ( Serial / PS2 ),

# Sistem Operasi :

# Minimal Windows 98 ( disarankan versi Windows yang lebih tinggi),

# Linux.

**III. Dasar Teori**

*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*. Dikarenakan perangkat tersebut tidak memiliki banyak memori, bahasa harus berukuran kecil dan mengandung kode yang liat. Juga karena manufakturmanufaktur 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 consumer 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. 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 object adalah satu kesatuan hirarki pewarisan

8. Dari bahasa Eiffel, fasilitas *assertion* yang mulai diterapkan di sebagian JDK

*Apa itu Teknologi JAVA?*

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 pada satu sistem operasi, tetapi dikembangkan untuk berbagai sistemoperasi dan bersifat *open source.*

Sebuah *Development Environment*

Sebagai sebuah peralatan pembangun, teknologi Java menyediakan banyak *tools* :

*compiler, interpreter,* penyusun dokumentasi, paket kelas dan sebagainya.

Sebuah Aplikasi

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

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.

*Mengapa Mempelajari JAVA?*

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

1. 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*.

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. Dapat didistribusi dengan mudah

Java dibuat untuk membuat aplikasi terdistribusi secara mudah dengan adanya *libraries* networking yang terintegrasi pada Java.

4. 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.

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. 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.

7. Architecture Neutral

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

8. Portabel

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. 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.

*Sebagian Fitur dari JAVA*

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.

*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.

*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.

***Fase – fase Pemrograman 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 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*

1. Menulis kode program *Text editor* Berkas berekstensi *.java*
2. Kompilasi program Java Compiler Berkas berekstensi .*class*
3. (Java Bytecodes)
4. Menjalankan program Java Interpreter Program Output

**V. Latihan**

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

|  |
| --- |
| **BAB II**  **ELEMEN-ELEMEN DASAR BAHASA JAVA** |

Tujuan Instruksional Khusus

Setelah menyelesaikan materi ini mahasiswa dapat:

1. Menulis program dengan bentuk yang mudah dibaca dan dimengerti.
2. Menerapkan konsep dasar pemrograman bahasa Java.
3. Menyebutkan ketentuan-ketentuan penulisan pengenal pada Java
4. Menggunakan separator dalam program dengan benar.
5. Menyebutkan dan menggunakan tipe-tipe data pada Java
6. Meyebutkan beberapa jenis statemen pada Java
7. Menjelaskan dan menggunakan blok secara benar untuk menulis program
8. Menjelaskan kegunaan komentar program
9. Menyebutkan jenis-jenis komentar program dan menerapkannya

# II. Persyaratan / Kebutuhan Sistem Komputer

# Persyaratan / kebutuhan sistem untuk menjalankan program Java adalah sebagai berikut :

# Processor P-II atau yang lebih tinggi,

# Ram minimal 32 Mbyte,

# Free space harddisk minimal 200 Mbyte,

# Mouse ( Serial / PS2 ),

# Sistem Operasi :

# Minimal Windows 98 ( disarankan versi Windows yang lebih tinggi),

# Linux.

**III. Dasar Teori**

**3.1 Bentuk Penulisan Program Java**

Bahasa Java adalah sebuah bahasa pemrograman yang bentuk penulisannya tidak diatur secara khusus (*free-form language*). Sebagai contoh program dibawah ini yang ditulis dengan beberapa bentuk.

|  |
| --- |
| //PROGRAM BENTUK PERTAMA  class HelloWorld1  {  public static void main(String[] args)  {  System.out.println("Hello World!");  }  } |
| //PROGRAM BENTUK KEDUA  class HelloWorld2{public static void main(String[] args){System.out.println("Hello World!");}} |

|  |
| --- |
| //PROGRAM BENTUK KETIGA  class HelloWorld3  {  public static void main(String[] args)  {  System.out.println("Hello World!");  }  } |

**3.2 Pengenal (*Identifier*)**

Pengenal adalah suatu nama yang biasa dipakai dalam pemrograman untuk menyatakan **nama variabel**, **nama metode, nama *class****.* Serta hal-hal lain yang didekarasikan atau didefinisikan oleh pemrogram.

**3.3 Alat Pemisah (*Separators*)**

Dalam Java, ada beberapa karakter yang digunakan sebagai separator. Separator paling umum adalah **semicolon**, yang biasa digunakan sebagai pemisah statemen. Di bawah ini ditunjukan Tabel 1.1 yang digunakan oleh Java.

**Tabel 1.1** Separator Java

|  |  |  |
| --- | --- | --- |
| **Simbol** | **Nama** | **Keterangan** |
| ( )  { }  [ ]  ;  ,  . | Tanda kurung *(Parentheses)*  Kurung kurawal *(Braces)*  Tanda kurung *(Brackets)*  Titik koma *(Semicolon)*  Koma *(Comma)*  Titik *(Period)* | Digunakan mengisi daftar parameter dalam metode**.** Juga digunakan untuk mendefinisikan prioritas (*precedence*) dalam ekspresi aritmetika.  Digunakan untuk mengisi nilai awal array secara otomatis. Juga digunakan untuk menentukan kode blok, *class,* metode, *local scopes*.  Digunakan untuk mendeklarasikan tipe array.  Pembatas statemen.  Pemisah pengenal (*identifier*)yang berurutan dalam deklarasi variabel.  Digunakan untuk memisahkan nama package dari subpackages  dan class. Juga digunakan untuk memisahkan sebuah variabel atau metode dari variabel referensi. |

### 3.4 Kata Kunci (*Keywords*)

Kata kunci (*keyword*) adalah pengenal system yang mempunyai makna khusus bagi kompiler. Kata kunci ini ditulis dengan menggunakan huruf kecil (*lowercase*). Tabel berikut ini menunjukan kata kunci dalam bahasa Java.

**Tabel 1.2**  Kata kunci bahasa Java

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| abstract | Continue | goto | package | synchronized |
| assert | default | if | private | this |
| Boolean | do | implements | protected | throw |
| break | double | import | public | throws |
| Byte | else | instanceof | return | transient |
| Case | Extends | int | short | try |
| catch | final | interface | static | Void |
| Char | finally | long | strictfp | volatile |
| Class | float | Native | super | while |
| const | for | new | switch |  |

**3.5 Tipe Data**

Dalam bahasa java ada tiga elemen dasar yang digunakan, yaitu : tipe data, variabel, dan array.

**3.5.1 Tipe Integers**

Java memdefinisikan empat tipe integer: **byte**, **short**, **int**, and **long**, yang semuanya adalah bilangan bertanda (*signed integers*), nilai positif dan negatif. Java tidak mendukung bialngan tak bertanda (*unsigned integers*). Lebar dan range dari setiap tipe dapat dilihat pada Tabel 1.3 berikut ini.

**Tabel 1.3**  Tipe data sederhana

|  |  |  |
| --- | --- | --- |
| **Tipe** | **Range** | **Lebar bit** |
| **Long** | 9,223,372,036,854,775,808 … 9,223,372,036,854,775,807 | 64 bit |
| **Int** | 2,147,483,648 … 2,147,483,647 | 32 bit |
| **Short** | 32,768 … 32,767 | 16 bit |
| **Byte** | 128 … 127 | 8 bit |

**3.5.2 Tipe Floating-Point**

Bilangan Floating-point, juga disebut bilangan *real*, digunakan ketika ekspresi memerlukan presisi yang kecil. Ada dua jenis tipe floating-point yaitu, **float** dan **double**, dimana mewakili bilangan presisi tunggal dan presisi ganda. Berikut ini Tabel 1.4 yang menunjukan lebar bit dan jangkauan ‘*range*’ dari jenis tipe floating point:

**Tabel 1.4** Tipe floating point

|  |  |  |
| --- | --- | --- |
| **Tipe** | **Range** | **Lebar bit** |
| **Double** | 4.9e–324 to 1.8e+308 | 64 |
| **Float** | 1.4e−045 to 3.4e+038 | 32 |

**3.5.3 Tipe Characters**

Tipe **char,** digunakan untuk menyimpan karakter.

Meskipun char bukan tipe integer, tetapi dapat dioperasikan seolah-olah merupakan bilangan integer. Hal ini bisa dikerjakan dengan menggunakan nilai kode ASCII, sehingga memungkinkan untuk dilakukan operasi *increment* atau *decrement*. Untuk contoh, perhatikan contoh program berikut ini.

**3.5.4 Tipe Booleans**

Tipe Boolean digunakan untuk operasi logika, yang mempunyai dua nilai yaitu **true** atau **false**. Operasi logika biasanya menggunakan operator relasi, dan juga digunakan pada ekspresi kondisional yang menggunakan **if** dan **for**.

**3.6 Strings**

Dalam Java, *strings* ditangani dengan kelas khusus yang disebut String. Berikut ini beberapa string yang dideklarasikan dengan menggunakan kelas Java String:

String string1;

String string2;

Lebih jauh mengenai kelas string dibahas pada bagian tersendiri pada kelas Package

**3.7 Tipe Casting**

Proses konversi satu tipe data ke tipe data yang lain disebut *casting*.

**Table 1.5** Tipe casting

|  |  |
| --- | --- |
| **Tipe Asal** | **Tipe Tujuan** |
| Byte | Short, char, int, long, float, double |
| Short | int, long, float, double |
| Char | int, long, float, double |
| Int | long, float, double |
| Long | float, double |
| Float | Double |

**3.8 Variabel**

Variabel adalah unit dasar dari penyimpanan dalam program Java atau dengan kata lain variabel adalah sebuah lokasi memori yang dinamai yang ditandai sebuah nilai oleh program kita. Untuk menggunakan variabel harus dideklarasikan sebelumnya. Bentuk dasar dari deklarasi variabel adalah sebagai berikut:

***type identifier* [ = *value*][, *identifier* [= *value*] ...] ;**

Berikut ini contoh pendeklarasian variabel dari bermacam-macam tipe.

|  |
| --- |
| int x, y, z;.  int d = 5, e, f = 9; //variable dengan inisialisasi/pemberian nilai awal  byte z = 26;  double pi = 3.14159; .  char y = 'y'; . |

### 3.9 Statemen ( *Statement* )

Statemen adalah sebuah perintah tunggal (*single command* ). Statemen dapat terdiri dari beberapa baris kode, tetapi kompiler membacanya sebagai satu statemen. Statemen tunggal (*single-line*) diakhiri semicolon (;), dan statemen group (*multi-line*) diakhiri (}). Statemen *Multi-line* secara umum disebut blok kode (code blocks ).

### 

### 3.10 Blok Kode (*Code Blocks*) dan *Scope*

### Dalam Java, *source code* dipecah menjadi bagian-bagian yang dipisahkan oleh *curly braces*: { dan }. Segala sesuatu atau statemen-statemen diantara *curly braces* dianggap sebagai sebuah *block.* Blok ini merupakan sebagai bagian dari sintaks bahasa Java. Jika kita tidak menggunakan *braces*, compiler akan mengalami masalah menentukan dimana satu bagian dari akhir kode dan bagian berikutnya dimulai.

Blok adalah statemen-statemen yang terdapat diantara dua kurung kurawal seperti berikut

|  |
| --- |
| {  blok statemen  } |

Anda juga membuat blok di dalam blok lain atau bisa dikatakan blok secara bertingkat :

|  |
| --- |
| {  <statemen>  {  <statemen>  }  } |

***Scope*** menentukan dimana dalam sebuah program, sebuah variabel adalah diakui. Variabel dibagi menjadi 2 kategori scope utama:

1. Variabel global
2. Variabel local

Berikut ini program yang mengilustrasikan penjelasan di atas.

|  |
| --- |
| class ScopingDemo  {  int x = 0;  void metode1()  {  int y;  y = x; // Metode ini dapat berjalan. metode1 dapat diakses y.  }  void metode2()  {  int z = 1;  z = y; // Metode ini tidak berjalan: y didefinisikan di luar skope method2.  }  } |

Untuk lebih memahami blok, kita ambil contoh kelas HelloWorld. Kelas HelloWorld disusun dari dua blok. Blok luar ‘o*uter block’* dan blok dalam ‘*inner block’.*  Blok luar dari program adalah blok yang mendefinisikan HelloWorld class:

|  |
| --- |
| **class HelloWorld**  {  ...  } |

Blok kelas sangat penting dalam Java. Untuk contoh, metode yang didefinisikan didalam kelas yang mereka miliki. Kelas HelloWorld, blok dalam mendefinisikan kode dalam metode main(), sebagai berikut:

|  |
| --- |
| **public static void main (String args[])**  {  ...  } |

Blok dalam dianggap menjadi  ***nested*** didalam blok luar dari program. Beberapa variabel yang didefinisikan dalam blok dalam adalah bersifat lokal ke blok itu dan tidak *visible* ke blok luar; scope dari variabel didefinisikan sebagai blok dalam.

Manfaat scope menjadi semakin terasa ketika kita menggunakan nesting blocks.

### 

### 3.11 Komentar Program (*Comments*)

1. Secara umum komentar digunakan untuk membantu pemrogram ataupun orang lain dalam memahami program.

Java menggunakan 3 macam komentar:

1. single-line comments
2. multi-line comments,
3. dan Javadoc comments.

**Tabel 1.6** Jenis komentar program

|  |  |  |
| --- | --- | --- |
| Jenis | Simbol | Keterangan |
| Single-  line | // ... | Digunakan untuk memberikan penjelasan singkat pada fungsi atau struktur dari statemen atau ekspresi. Komentar ini hanya memerlukan opening tag (//…) pada awal komentar. Semua karakter sesudah // sampai akhir tidak akan dikenali/diproses oleh program |
| Multi-  line | /\* ... \*/ | Digunakan untuk memberikan penjelasan secara lebih detil lebih dari satu baris pada fungsi atau struktur dari statemen atau ekspresi. Komentar ini hanya memerlukan opening dan closing tag (/\*…\*/) pada awal komentar.Semua karakter diantara tag ini tidak akan diproses oleh program. |
| Javadoc | /\*\* ... \*/ | Sama dengan tag /\* \*/, kecuali bahwa ini tag ini juga dapat digunakan dengan javadoc tool untuk membuat dokumentasi secara otomatis. |

**IV. Langkah Kerja**

1. Jalankan Edit Plus, buat file baru java dan ketikkan kode program berikut :

Untuk contoh, dibawah ini diberikan sebuah program yang menghitung perjalanan cahaya yang ditentukan dalam hari.

|  |
| --- |
| class Light  {  public static void main(String args[])  {  int lightspeed;  long days;  long seconds;  long distance;  lightspeed = 186000;  days = 1000;  seconds = days \* 24 \* 60 \* 60; // konversi ke menit  distance = lightspeed \* seconds; // hitung jarak  System.out.print("Dalam " + days);  System.out.print(" hari cahaya akan berjalan kira-kira");  System.out.println(distance + " miles.");  }  } |

1. Simpan program Java di atas.
2. Eksekusi program Java Pertama dengan mengklik ***Tool*** 🡪 ***Compile Program Java***. Lalu ***Tool*** 🡪 ***Menjalankan Program Java***.
3. Ketikkan kode program Java berikut ini :

|  |
| --- |
| class DoubleDemo  {  public static void main(String args[])  {  double pi, r, Luas;  r = 20.8;  pi = 3.1416;  Luas = pi \* r \* r;  System.out.println("Luas lingkaran= " + Luas);  }  } |

|  |
| --- |
| Class CharDemo  {  public static void main(String args[])  {  char kar1, kar2;  kar1 = 65;  kar2 = 'B';  System.out.println("karakater 1 :"+kar1);  System.out.println("karakater 2 :"+kar2);  }  } |

|  |  |
| --- | --- |
| |  | | --- | | class CharDemo2  {  public static void main(String args[])  {  char kar1,kar2;  kar1 = 'A';  kar2 = 'B';  kar1++;  System.out.println("kar1 = " + kar1 );  kar2--;  System.out.println("kar2 = " + kar2);  }  } | |

|  |  |
| --- | --- |
| |  | | --- | | class BooleanDemo  {  public static void main(String args[])  {  boolean b;  b = false;  if(2 > 3)  b = false;  System.out.println(" 2 > 3 adalah " +b );  //Ditulis dengan bentuk berbeda  if(b)  System.out.println(" 2 > 3 adalah " + (2 > 3));  }  } | |

Di bawah ini ditunjukan contoh program yang menggunakan kelas ***String***.

|  |
| --- |
| class KelasString  {  public static void main (String args[])  {  String String1 = "HALO APA KHABAR";  String String2 = "TEMAN-TEMAN ALUMNI 91 ";  String String3 = "SMA NEGERI SATU PEMALANG";  System.out.println(String1);  System.out.println(String2);  System.out.println(String3);  }  } |

Berikut ini diberikan contoh penerapan tipe casting dalam program Java.

|  |
| --- |
| class CastingDemo  {  public static void main(String[] args)  {  int x=21;  float y;  y=(float) x/2;  System.out.println(y);  }  } |

Program berikut menunjukan bagaimana sebuah variabel dideklarasikan dan bagaimana ditandai sebuah nilai.

|  |  |
| --- | --- |
| |  | | --- | | Class VariabelDemo  {  public static void main(String args[])  {  int num; // mendeklarasikan variable jum  int jum = 500; // memberikan jum nilai 500  System.out.println("Nilai jumlah: " +jum);  jum = jum \* 2;  System.out.print("Hasil perkalian jum \* 2 adalah = ");  System.out.println(jum);  }  } | |

Berikut ini adalah contoh program dengan menggunakan indentation dari blok dalam Java:

|  |
| --- |
| for (int i = 0; i < 5; i++)  {     if (i < 3)  {       System.out.println(i);     }  } |

Contoh program di atas jika ditulis tanpa menggunakan indentations blok:

|  |
| --- |
| for (int i = 0; i < 5; i++)  {  if (i < 3)  {  System.out.println(i);  }  } |

Contoh yang lain dari penggunaan blok kode adalah sebagai target dari looping **for.**

|  |  |
| --- | --- |
| |  | | --- | | class BlockTest  {  public static void main(String args[])  {  int x, y;  y = 20;  // target dari loop adalah sebuah block  for(x = 0; x<10; x++)  {  System.out.println("Nilai x adalah : " + x);  System.out.println("Nilai y adalah : " + y);  y = y - 2;  }  }  } | |

Berikut ini diberikan contoh program untuk lebih memahami Blok Scope.

|  |
| --- |
| Class BlokScopeDemo  {  public static void main (String args[])  {  int i,j;  hitungLuas();  }    public static void hitungLuas()  {  int p=30,l=20;  int Luas=p \* l;  System.out.println("LUAS = "+Luas);  }  } |

Berikut ini diberikan sebuah contoh program dengan menggunakan nesting blok.

|  |
| --- |
| Class NestingBlok  {  public static void main (String args[])  {  int i, j;  System.out.println("Hello, Java!");    for (i = 0; i < 5; i++)  {  int k;  j=i+1;  System.out.print(j);  System.out.println(" Bye!");  }  }  } |

1. Ulangi langkah nomor 3, amati dan catat hasilnya.

**V. Latihan**

1. Jelaskan yang dimaksud dengan *free-form language* pada penulisan program Java
2. Jelaskan pengertian identifier dan ketentuan penulisannya
3. Sebutkan simbol-simbol separator dan jelaskan kegunaan masing-masing simbol tersebut.
4. Jelaskan apa yang dimaksud bahasa Java adalah sebuah bahasa pemrograman bersifat *strongly typed language*
5. Java memiliki delapan tipe data sederhana atau sering disebut sebagai tipe data primitif, jelaskan berdasarkan kelompok dan kegunaan masing-masing.
6. Apa yang yang dimaksud *casting*, kapan kita perlu menggunakan casting, dan bagaimana aturannya
7. Jelaskan apa yang dimaksud variable, statemen dan blok kode

|  |
| --- |
| **BAB 3**  **OPERATOR BAHASA JAVA** |

I. Tujuan Instruksional Khusus

Setelah menyelesaikan materi ini mahasiswa dapat:

1. menjelaskan tentang operator dalam Java.
2. menyebutkan jenis operator.
3. menyebutkan operator presedence.
4. membuat program mengunakan operator.

# II. Persyaratan / Kebutuhan Sistem Komputer

# Persyaratan / kebutuhan sistem untuk menjalankan program Java adalah sebagai berikut :

# Processor P-II atau yang lebih tinggi,

# Ram minimal 32 Mbyte,

# Free space harddisk minimal 200 Mbyte,

# Mouse ( Serial / PS2 ),

# Sistem Operasi :

# Minimal Windows 98 ( disarankan versi Windows yang lebih tinggi),

# Linux.

**III. Dasar Teori**

* 1. Operator

Operator adalah karakter khusus yang memerintahkan *compiler* untuk melakukan operasi terhadap sejumlah *operand*. Perintah operasi dispesifikasikan oleh operator yang operandnya berupa variabel, pernyataan atau konstanta.

Menurut tempatnya dan *operand* yang diproses, operator dibagi menjadi 5 yaitu :

* Operator unary : operator yang mengolah satu operand.
* Operator prefiks : operator yang ditempatkan sebelum operand.
* Operator biner infiks : operator yang ditempatkan diantara dua operand.
* Operator postfiks : operator yang ditempatkan setelah operand.
* Operator ternary : operator yang membutuhkan tiga operand

Menurut jenisnya operator dasar Java ada 4 yaitu : **aritmatika**, **bitwise**, **relasi** dan **logika**.

* 1. **Operator Aritmatika**

Digunakan untuk proses matematik. Operator aritmatika tidak bisa digunakan untuk tipe boolean tetapi dapat digunakan pada tipe char. Anggota dari operator aritmatika sebagai berikut:

|  |  |
| --- | --- |
| **Operator** | **Hasil** |
| + | Penjumlahan |
| += | Persamaan penjumlahan |
| - | Pengurangan (unary minus) |
| -= | Persamaan Pengurangan |
| \* | Perkalian |
| \*= | Persamaan Perkalian |
| / | Pembagian |
| /= | Persamaan Pembagian |
| % | Modulus (sisa pembagian) |
| %= | Persamaan Modulus |
| ++ | Penambahan |
| -- | Pengurangan |

Contoh :

int a = 1 + 1;

int b = a \* 5;

int c = b / 10;

int d = c - 1;

int e = -a;

int x = 42 % 10; // Modulus/pembagian sisa nilai x = 2

i += 4; // artinya i = i + 4

j -= 2; // artinya j = j - 2

k \*= 3; // artinya k = k \* 3

a /= 2; // artinya a = a / 2

b %= 2; // artinya b = b % 2

* 1. **Operator Bitwise Integer**

Tipe numerik integer: long, int, short, char dan byte memiliki operator tambahan yang dapat memodifikasi dan memeriksa bit-bit yang menyusun nilainya. Operatornya adalah sebagai berikut :

|  |  |
| --- | --- |
| **Operator** | **Hasil** |
| ~ | Unary NOT |
| & | AND |
| | | OR |
| ^ | Exclusive OR |
| >> | Shift kanan |
| >>> | Shift kanan, isi dengan nol |
| << | Shift kiri |
| &= | Pernyataan AND |
| |= | Pernyataan OR |
| ^= | Pernyataan exclusive OR |
| >>= | Pernyataan shift kanan |
| >>>= | Pernyataan shift kanan, isi dengan nol |
| <<= | Pernyataan shift kiri |

Operator bitwise mengkonversi bilangan desimal ke dalam bentuk biner kemudian dimanipulasi sesuai dengan tanda operatornya. Contoh :

10 diubah menjadi 1010

Dari kiri:

20x0=0

21x1=2

22x0=0

23x1=8 +

10

~10 hasilnya : NOT 1010 = 0101 (membalikan/invers semua bit)

10 & 2 = 1010 10 | 2 = 1010

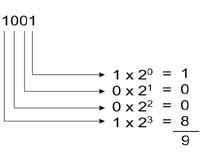
0010 0010

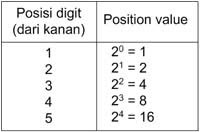
|

&

0010 1010

Biner adalah nama lain dari bilangan base 2. Biner adalah sistem bilangan yang hanya memuat 2 angka, yaitu angka 0 dan 1. Jika dalam desimal, angka disebut dengan Digit. Dalam biner, Angka disebut Bit (Binary Digit). Contoh bilangan biner adalah 1001110, 1011001, dan lain-lain. Konsep penentuan nilai biner sebenarnya sama dengan Desimal. Hanya saja jika di desimal menggunakan perpangkatan angka 10, di biner menggunakan perpangkatan angka 2. Bilangan Biner bisa kita ubah ke bilangan desimal. berikut ini adalah contoh mengubah Bilangan Biner 1001 ke bilangan desimal

[](https://welehcool123.files.wordpress.com/2014/12/4eede-binari-1.jpg)Position Value dalam sistem Bilangan Biner merupakan perpangkatan dari nilai 2 (basis), seperti pada tabel berikut ini :

[](https://welehcool123.files.wordpress.com/2014/12/be05e-binari-2.jpg)

Berarti, Bilangan Biner 1001perhitungannya adalah sebagai berikut :

[8d471-binari-3](https://welehcool123.files.wordpress.com/2014/12/8d471-binari-3.jpg)

1001**2** = (1 x 8) + (0 x 4 ) + (0 x 2) + (1 x 1) = 9**10**. Sehingga 1001 **2** = 9**10**

contoh lainnya adalah mengubah bilangan biner 1011 ke bilangan desimal.  
1011 = (1 \* 2^3) + (0 \* 2^2) + (1 \* 2^1) + (1 \* 2^0) = 8 + 0 + 2 + 1 = 11. Maka bilangan 1011 dalam biner sama dengan bilangan 11 dalam desimal. atau dapat ditulis 1011**2** = 11**10 .**

KIta juga bisa mengkonversi bilangan desimal ke bilangan biner. Untuk mengubah desimal ke biner juga sangat sederhana, Anda hanya membagi nilai desimal dengan 2 dan kemudian menuliskan sisanya, ulangi proses ini sampai Anda tidak bisa membagi dengan 2 lagi. lalu, Sisa-sianya ini kita urutkan dari yang sisa dari pembagian yang hasilnya nol sampai ke pembagian pertama.

misalnya mari kita mengambil nilai desimal 157:

* **157 ÷ 2 = 78** dengan sisa 1
* **78 ÷ 2 = 39** dengan sisa 0
* **39 ÷ 2 = 19** dengan sisa 1
* **19 ÷ 2 = 9** dengan sisa 1
* **9 ÷ 2 = 4** dengan sisa 1
* **4 ÷ 2 = 2** dengan sisa 0
* **2 ÷ 2 = 1** dengan sisa 0
* **1 ÷ 2 = 0**                 dengan sisa 1

Lalu kita urutkan dari paling bawah ke paling atas, sehingga mendapatkan 10011101. Sehingga 157**10** = 10011101 **2** .

* 1. **Operator Relasi**

Digunakan untuk membandingkan dua buah nilai, yang dijabarkan pada tabel berikut :

|  |  |
| --- | --- |
| **Operator** | **Hasil** |
| == | Sama dengan |
| != | Tidak sama dengan |
| > | Lebih besar dari |
| < | Lebih kecil dari |
| >= | Lebih besar sama dengan |
| <= | Lebih kecil sama dengan |

Semua operator di atas akan menghasilkan nilai boolean (true dan false) contoh :

int a = 4;

int b = 2;

boolean c = a < b; // c bernilai false

* 1. **Operator Logika Boolean**

Operator ini bekerja pada operand boolean, yang akan memanipulasi nilai boolean untuk menghasilkan nilai boolean pula, contoh :

|  |  |
| --- | --- |
| **Operator** | **Hasil** |
| & | Logika AND |
| | | Logika OR |
| ^ | Logika XOR |
| || | Hubungan-singkat OR |
| && | Hubungan-singkat AND |
| ! | Hubungan-singkat NOT |
| &= | Pernyataan AND |
| != | Pernyataan OR |
| ^= | Pernyataan XOR |
| == | Sama dengan |
| != | Tidak sama dengan |
| ?: | Operator ternary if-then-else |

boolean a=true;

boolean b=false;

boolean c=a | b;

boolean d=a & b;

Operator hubung singkat (*short circuit logical operator*) digunakan apabila ada dua operan (operan kanan dan kiri) dimana operan kanan bergantung dari operan kiri, baik true maupun false.

Contoh : if (denom != 0 && num / denom > 10)

Keterangan :

* + - Bila nilai denom = 0, maka nilainya adalah false sehingga operan kanan tidak akan diperiksa karena hasilnya false sesuai dengan operan kiri.
    - Bila nilai denom != 0, maka nilainya adalah true dan perlu diANDkan dengan num/denom > 10, sehingga operan kiri dan kanan perlu diANDkan untuk mengetahui hasil logikanya (TRUE atau FALSE)

Operator Ternary if –then-else membutuhkan 3 parameter yang bentuknya adalah sebagai berikut :

Expression ? statement1 : statemen2

Keterangan : expression berupa pernyataan yang memeriksa besaran Boolean. Jika hasilnya true maka statement1 dijalankan, sebaliknya jika hasilnya flase maka statement2 dijalankan. Syaratnya staement1 dan 2 harus menghasilkan tipe yang sama dan tidak boleh void.

Contoh :

Ratio = denom == 0 ? 0 : num / denom;

Ketika dijalankan Java akan memeriksa pernyataan di sebelah kiri ? (denom == 0), jika denom = 0 maka pernyataan diantara tanda ? dan : akan diperiksa dan ditetapkan sebagai hasil/nilai Ratio, sebaliknya jika tidak sama dengan 0 maka pernyataan num / denom yang ditetapkan sebagai hasil/nilai Ratio.

* 1. **Preseden Operator**

Preseden operator menunjukan adanya urutan tingkatan operator (hirarki) yang penjabarannya pada tabel berikut :

|  |  |  |  |
| --- | --- | --- | --- |
| tertinggi | | | |
| () | [] |  |  |
| ++ | -- | ~ | ! |
| \* | / | % |  |
| + | - |  |  |
| >> | >>> | << |  |
| > | >= | < | <= |
| == | != |  |  |
| & |  |  |  |
| ^ |  |  |  |
| | |  |  |  |
| && |  |  |  |
| || |  |  |  |
| ?: |  |  |  |
| terendah | | | |

**IV. Langkah Kerja**

1. Jalankan Edit Plus, buat file baru java dan ketikkan kode program berikut :

|  |
| --- |
| class Aritmatika1  {    public static void main(String[] args)  {  int a = 1 + 1;  int b = a \* 3;  int c = b / 4;  int d = b - a;  int e = -d;    System.out.println("a = " + a);  System.out.println("b = " + b);  System.out.println("c = " + c);}  System.out.println("d = " + d);  System.out.println("e = " + e);  }  } |

1. Simpan program Java di atas.
2. Eksekusi program Java Pertama dengan mengklik ***Tool*** 🡪 ***Compile Program Java***. Lalu ***Tool*** 🡪 ***Menjalankan Program Java***.
3. Ketikkan kode program Java berikut ini :

|  |
| --- |
| class Aritmatika2 {    public static void main(String[] args)  {  int x = 42;  double y = 42.3;  int a = 1;  int b = 2;  int c = 3;  a += 5;  b += 4;  c += a \* b;  c %= 6;    System.out.println("x mod 10 = " + x % 10);  System.out.println("y mod 10 = " + y % 10);  System.out.println("a = " + a);  System.out.println("b = " + b);  System.out.println("c = " + c);  }  } |

|  |
| --- |
| class Aritmatika3 {    public static void main(String[] args)  {  int a = 1;  int b = 2;  int c = ++b; // ++b artinya b = b + 1  int d = a++;  c ++;    System.out.println("a = " + a);  System.out.println("b = " + b);  System.out.println("c = " + c);  System.out.println("d = " + d);  }  } |

|  |
| --- |
| class Bitlogic {    public static void main(String[] args)  {  String binary[] {  "0000", "0001", "0010", "0011", "0100", "0101", "0110",  "0111", "1000", "1001", "1010", "1011", "1100", "1101"  "1110", "1111" };  int a = 3;  int b = 6;  int c = a b;  int d = a & b;  int e = a ^ b;  int f = (~a & b) (a & ~b);  int g = ~a & 0x0f; // nilai ~a agar kurang dari 16  System.out.println("a = " + binary[a]);  System.out.println("b = " + binary[b]);  System.out.println("a b = " + binary[c]);  System.out.println("a & b = " + binary[d]);  System.out.println("a ^ b = " + binary[e]);  System.out.println("~a & ba & ~b = " + binary[f]);  System.out.println("~a = " + binary[g]);  }  } |

1. Ulangi langkah nomor 3, amati dan catat hasilnya.

**V. Pertanyaan**

* 1. Jelaskan apa yang dimaksud operator, operand dan precedence.
  2. Sebutkan jenis operator dalam Java! Dan berikan contoh masing-masing.
  3. Sebutkan hasil dari perhitungan di bawah ini :

A = 10 / 2 \* 5 – 2 >> 3;

B = 4 \* 3 / 4 << 5 – 4;

C = (b == 0) ? 0 : ( a / b);

* 1. Buat program untuk menghitung operasi penjumlahan,pengurangan,perkalian dan pembagian.
  2. Buatlah program untuk menghitung :
     1. luas persegi panjang,
     2. luas lingkaran,
     3. dan luas bujur sangkar.