PEMBAHASAN BAHASA *JAVA* PADA JANTUNGNYA PEMPROGRAMAN

Warno
Program studi Teknik Informatika, Fakultas Tehnik, MIPA Universitas Indraprasta PGRI
Jl. Nangka No.58C Tanjung Barat (TB Simatupang), Jagakarsa, Jakarta Selatan
warnooke@gmail.com

Abstrak

Java adalah bahasa pemrograman adalah kasus sensitif, yang meanswriting huruf besar dan huruf kecil memiliki arti yang berbeda. Namun, spasi dan baris baru tidak penting dalam Javalanguage kecuali untuk memisahahkan antara kata-kata. Parameter String [] args parameter adalah wajib dalam metode utama. Men kitakan statis mengatakan bahwa metode utama tidak akan mengubah kelas tersebut yang objek Hello. Instruksi atau pernyataan dalam laporan-metode utama yang berada di antara {} - dieksekusi satu demi satu pernyataan andevery diakhiri dengan simbol (;).

Kata kunci: bahasa java, pemrograman, instruksi

Pendahuluan

Pada tahun 1991**, Sun** dipimpin **Patric Naughton** dan **James Gosling** ingin merancang bahasa computer untuk perangkat *consumer* seperti *cable TV Box*. Karena perangkat itu tidak mempunyai banyak memori, bahasa harus berukuran kecil dan menghasilkan kode program yang liat. Juga karena manufaktur-manufaktur berbeda memilih pemroses-pemroses yang berbeda, maka bahasa harus bebas dari arsitektur manapun. Proyek ini diberi nama kode "**Green**".

Kebutuhan untuk kecil, liat dan kode netral terhadap *platform* mengatur tim mempelajari implementasi pascal yang pernah dicoba. **Niklaus Wirth**, pencipta bahasa Pascal telah merancang bahasa *portable* yang menghasilkan kode *intermediate* untuk mesin hipotetis. Mesin ini sering disebut *Virtual machine*. Kode antara ini kemudian dapat digunakan disembarang mesin yang memiliki *interpreter*. Proyek **Green** menggunakan *virtual machine* untuk mengatasi isu utama netral terhadap arsitektur mesin.

Karena orang-orang di proyek Green berbasis C++ bukan Pascal maka kebanyakan Sintaks diambil dari C++, serta mengadopsi orientasi obyek bukan *procedural*. Mulanya bahasa yang diciptakan diberi nama "**Oak**" kemudian diganti "***Java***"karena telah ada bahasa pemrograman bernama "**Oak**". Produk pertama proyek **Green** adalah "*7", sebuah kendali jauh yang sangat cerdas. Karena pasar masih belum tertarik dengan produk *consumer* cerdas maka proyek **Green** harus menemukan pasar lain dari teknologi yang diciptakan. Kemudian, penerapan mengarah menjadi teknologi yang berperan di web.

Pada 1995, **Netscape** memutuskan membuat *browser* yang dilengkapi dengan *Java*. Setelah itu diikuti oleh **IBM,Symantec,Inprise,** bahkan **Microsoft**. Setelah itu *Java* mulai didengar. Dengan strategi terbukanya, banyak industri yang meliriknya. Bersamaan itu disusul berbagai universitas Amerika, Jepang, dan Eropa yang mengubah pengenalan bahasa pemrograman komputer menjadi *Java*, meninggalkan C++. *Java* lebih sederhana dan telah mengakomodasikan hampir seluruh fitur penting bahasa-bahasa pemrograman yang ada semenjak perkembangan komputasi modern.

Java pertama kali diluncurkan sebagai bahasa pemrograman umum (*general purpose programming language*) dengan kelebihan dia bisa dijalankan di web *browser* sebagai *applet*. Sejak awal, para pembuat *Java* telah menanamkan visi mereka ke dalam *Java* untuk small embedded customer device)seperti TV, telepon, radio, dan sebagainya supaya dapat berkomunikasi satu sama

lain. Langkah pertama yang diambil oleh Sun Microsistem adalah dengan membuat JVM (*JavaVirtual machine*) yang kemudian diimplementasikan dalam bentuk JRE (*Java Runtime Environment*). JVM adalah lingkungan tempat eksekusi program *Java* berlangsung dimana para obyek saling berinteraksi satu dengan yang lainnya. *Virtual machine* inilah yang menyebabkan *Java* mempunyai kemampuan penanganan memori yang lebih baik, keamanan yang lebih tinggi serta portabilitas yang besar. Apabila kita hanya ingin menjalankan program *Java*, maka kita cukup memiliki JRE saja.Tapi seandainya kita ingin mengembangkan perangkat lunak sendiri, JRE saja tidak cukup.

Untuk lebih meningkatkan produktivitas pengembang perangkat lunak, Sun juga meluncurkan SDK (*Standard Development Kit*) yang berisi kakas dan API untuk membuat program aplikasi berbasis *Java*. Pada tahun 1999 Sun meluncurkan J2EE (*Java 2 Enterprise Edition*) sebagai *framework* untuk membuat aplikasi *enterprise* berskala besar. Pada tahun 2001, Sun meluncurkan J2ME yang kelak menjadi salah satu standar pemrograman di dalam PDA maupun *handphone.*

Sederhana, semudah C dan seampuh C++: berlawanan dengan anggapan orang-orang bahwa bahasa *Java* sulit untuk dipelajari, *Java* gampang untuk dipelajari terutama untuk orang yang sudah mengenal pemrograman tapi belum terlalu terikat pada paradigma pemrograman prosedural. Tentu saja ini berarti bahwa kita harus siap mempelajari salah satu teknologi yang berkembang paling cepat di dunia dalam dua tahun terakhir ini dengan banyak membaca tentunya baik dari buku maupun melalui web.

Sangat berorientasi obyek (OOP) dengan implementasi yang sangat baik sehingga kita bukan hanya belajar bagaimana membuat program yang baik (*reusable, scalable*, dan *maintanable*) tetapi juga kita belajar bagaimana cara berfikir yang baik untuk mengenali struktur masalah yang sedang kita hadapi dan memecahkannya secara sistematis dengan pola-pola tertentu (*patterns OpenPlatform*, *Write Once Run Anywhere* (WORA), *portable* atau *multiplatform*, program yang kita buat dapat dijalankan di Windows, Linux/Unix, Solaris, dan Macintosh tanpa perlu diubah maupun di kompilasi ulang. *Java* adalah juga bahasa yang paling sesuai digunakan bersama dengan XML yang membuat data menjadi *portable*, ini karena kelahiran XML tidak terlepas dari dukungan parser-parser berbahasa *Java*.

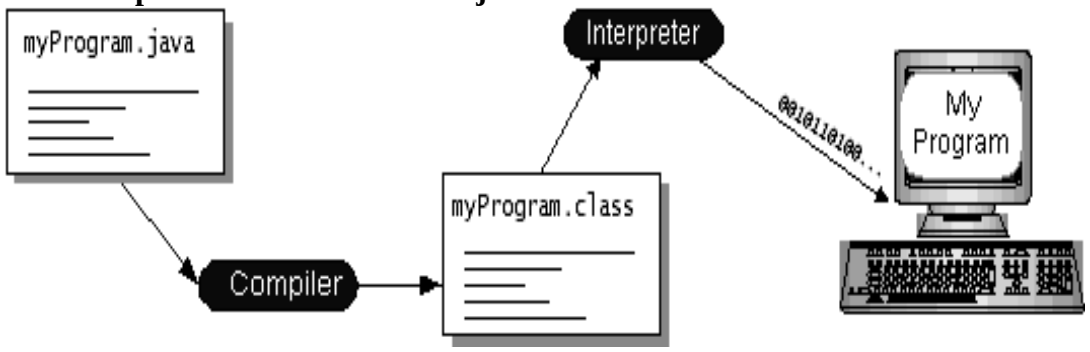
Arsitekturnya yang kokoh dan pemrograman yang aman. Dalam *Java* program yang kita buat tidak mudah untuk "*hang*" karena konflik pada memori biasanya diselesaikan dengan mengumpulkan obyek-obyek yang sudah tak terpakai lagi secara otomatis oleh *garbage collector*. Penanganan kesalahan juga dipermudah dalam *Java* dengan konsep *Exception*.

1. Bukan sekedar bahasa tapi juga *platform* sekaligus arsitektur. *Java* mempunyai portabilitas yang sangat tinggi. Ia dapat berada pada *smartcard*, *pager*, POS (*Point of Service*), *handphone,* PDA, palm, TV, *Embedded device* (PLC, *micro controller*), laptop, pc, dan bahkan server). Menyadari akan hal ini Sun membagi arsitektur *Java* membagi tiga bagian, yaitu:
   a. *Enterprise Java* (J2EE) untuk aplikasi berbasis web, aplikasi sistem tersebar dengan beraneka ragam klien dengan kompleksitas yang tinggi. Merupakan superset dari Standar *Java*
   b. *Standard Java* (J2SE), ini adalah yang biasa kita kenal sebagai bahasa *Java*, dan merupakan fokus kita sekarang.
   c. *Micro Java* (J2ME) merupakan subset dari J2SE dan salah satu aplikasinya yang banyak dipakai adalah untuk *wireless device*/*mobile device*
2. Program *Java* dijalankan menggunakan *interpreter* melalui *Java Virtual machine* (JVM). Hal ini menyebabkan *source codeJava* yang telah dikompilasi menjadi *Java bytecodes*dapat dijalankan pada *platform* yang berbeda-beda.
3. Fitur-fitur utama yang lain:
   a. Mendukung *multithreading.*
   b. Selalu memeriksa tipe obyek pada saat *runtime.*
   c. Mempunyai *automatic garbage collection* untuk membersihkan obyek yang tidak terpakai dari memori
   d. Mendukung *exception* sebagai salah satu cara penanganan kesalahan

Bagaimana Java Bekerja?

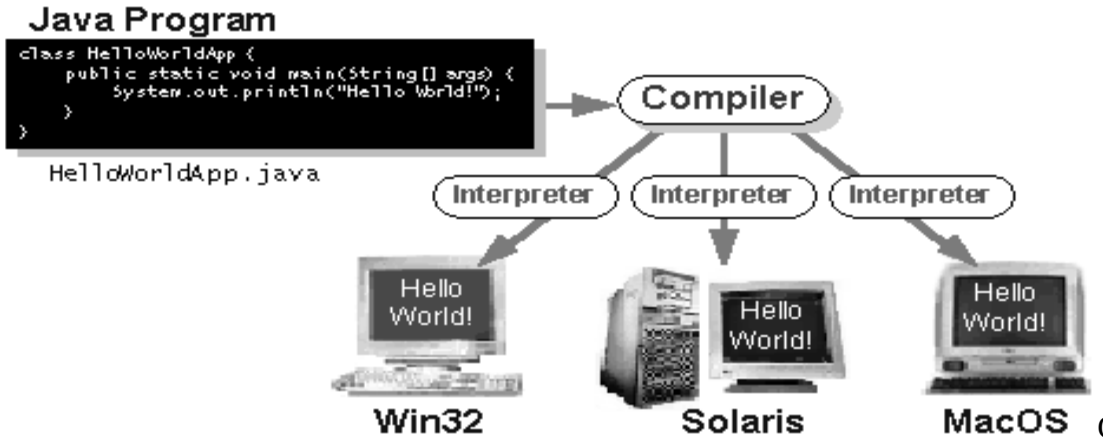
Lingkungan pemrograman pada *Java* menggunakan *compiler* sekaligus *interpreter* agar dapat berjalan pada *platform* yang berbeda. *Java compiler* melakukan kompilasi pada *source code* (***.Java***) menjadi *Java bytecodes* (**.class**) seperti ditunjukkan oleh Gambar 1 berikut.

Mekanisme Kompilasi Bahasa *Java* Bekerja



Gambar 1
Mekanisme Kompilasi dan Eksekusi Program *Java*

Java bytecodes merupakan instruksi mesin yang tidak spesifik terhadap *processor*. Oleh karena itu, program *Java* hasil kompilasi akan dapat dijalankan pada berbagai *platform* sistem komputer dengan menggunakan *Java Virtual machine* (JVM), *"write once, run anywhere"* (lihat Gambar 2). JVM disebut juga *bytecodes interpreter* atau *Java runtime interpreter*.



Gambar 2
Error! No text of specified style in document.
Konsep *Write Once, Run Anywhere* pada *Java*

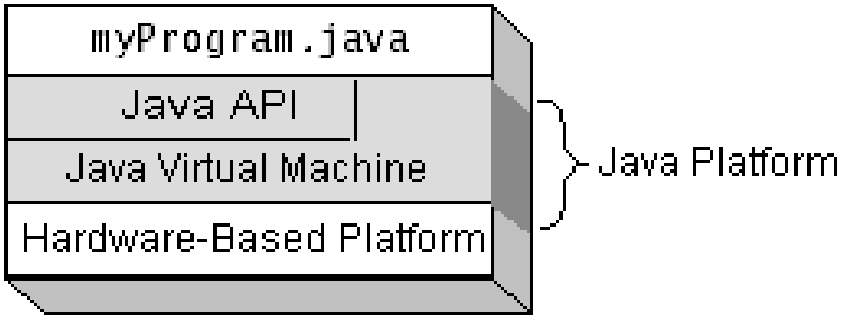
Platform Java

Platform dapat diartikan sebagai lingkungan perangkat keras atau perangkat lunak dimana program dijalankan. Umumnya *platform* dinyatakan berdasarkan nama sistem operasi yang digunakan, misalnya Windows 2000, Linux, Solaris, atau MacOS. Tidak seperti bahasa pemrograman lainnya, *platform Java* mempunyai dua komponen, yaitu:

1. *Java Virtual machine* (*Java* VM)
   Merupakan fondasi untuk *platform Java* yang dapat digunakan di berbagai *platform* perangkat keras.
2. *Java Application Programming Interface* (*Java* API)
   Kumpulan komponen-komponen perangkat lunak siap pakai (*ready-made software components*)

untuk berbagai keperluan penulisan program, seperti *graphical user interface* (GUI).
Gambar 3 berikut memperlihatkan gambaran program yang berjalan diatas di kedua komponen *platform Java*.

Platform dalam bahsa *Java* :



Gambar 3
Platform Java

Aplikasi Java

Ada dua tipe aplikasi pada *Java* yang umumnya sering ditulis, yaitu aplikasi *stand alone* dan *applets*. Aplikasi *stand alone* merupakan aplikasi yang dijalankan langsung diatas *platform Java*. Sedangkan *applets* adalah aplikasi yang dijalankan melalui *web browser* ataupun *applet viewer*.
Perbedaan *applets* dengan *Java stand alone* adalah:

a. *Applets* melakukan *extends* dari *class applets*.
b. *Applets* tidak mempunyai fungsi main().
c. *Applets* mempunyai beberapa batasan keamanan, seperti tidak diperbolehkan membaca atau menulis file pada sistem komputer.
d. *Applets* tidak dapat menjalankan program lain pada komputer yang menjalankan *applets*.

Selain kedua tipe aplikasi tersebut, ada beberapa tipe aplikasi *Java* yang lain yaitu:

a. Aplikasi berbasis Windows, misalnya dengan menggunakan fasilitas *Swing*.
b. *Java Servlet*, yaitu aplikasi seperti *applets* tetapi dijalankan di sisi *server*.
c. *Java Server Pages*, yaitu aplikasi di sisi *server* yang ditulis dalam bahasa *script* yang melekat pada halaman HTML untuk memberikan efek tampilan pada *web browser*.
d. *Java Beans* dan *Enterprise Java Beans*, yaitu aplikasi untuk program *multitiers*.
e. *Java Micro Edition*, yaitu aplikasi yang diperuntukan bagi perangkat genggam (*handheld devices*), seperti telepon genggam.

Identifier di Java

Identifier adalah nama yang diberikan kepada variabel, *method*, kelas, paket, dan interface untuk unik mengidentifikasikan bagi kompilator dan memberikan nama yang berarti bagi pemrogram. Adapun tatacara penamaan *identifier*:

1. *Case sensitive*, huruf kapital dan kecil dibedakan
2. *Identifier* yang diberikan oleh pemrogram tidak boleh sama dengan *keyword* yang ada di *Java*
3. Dimulai dengan huruf atau *underscore* (garis bawah) atau tanda ($). Namun sebisa mungkin diawali dengan huruf karena mungkin *identifier* dengan awalan *underscore* dan ($) digunakan unruk pemrosesan internal dan *fileimport*
4. Karakter berikutnya dapat berupa huruf atau angka 0 sampai 9. Simbol-simbol seperti '+' dan spasi tidak dapat digunakan

Keyword di Java

Keyword adalah *identifier* yang digunakan *Java* untuk suatu tujuan khusus. Daftar *keyword Java* adalah sebagai berikut:

Tabel 1
Keyword dalam *Java*

abstract	do	implements	private	this
boolean	double	*import*	protected	throw
break	else	*instance*of	public	throws
byte	extends	int	return	transient
case	false	interface	short	true
catch	final	long	static	try
char	finally	native	strictfp	void
class	float	new	super	volatile
continue	for	null	switch	while
default	if	package	synchronized	

Tipe Data di Java

Tipe data dalam *Java* dibagi dalam dua kategori:

1. Sederhana
2. Komposit

Tipe data sederhana

Tipe data sederhana merupakan tipe inti. Tipe sederhana tidak diturunkan dari tipe lain. Tipe ini adalah tipe data primitif. Terdapat delapan tipe data primitif di *Java*:

1. Empat tipe adalah untuk bilangan bulat: ***byte, short, int, long***
2. Dua untuk tipe angka pecahan (*floating point*): ***float, double***
3. Satu untuk tipe karakter, yaitu ***char***
4. Satu untuk tipe ***boolean*** yang berisi nilai logika: *true/false*

1. Tipe data *integer*

Tipe data *integer* memiliki jangkauan nilai sebagai berikut:

Tabel 2
Tipe data Integer

Panjang Integer	Tipe data	Jangkauan Nilai
8 bit	*byte*	-27 to 27-1
16 bit	*short*	-215 to 215-1
32 bit	*int*	-231 to 231-1
64 bit	*long*	-263 to 263-1

Pada kebanyakan situasi, tipe ***int*** paling banyak dipakai. Untuk data yang berukuran besar, digunakan tipe data ***long***. Tipe ***short***dan***byte*** terutama digunakan untuk aplikasi khusus yang menangani file level rendah.

2. Tipe data *Floating point*

Tipe data ini digunakan untuk perhitungan yang melibatkan bilangan pecahan, seperti perhitungan kosinus, akar persamaan, dan sebagainya. *Java* mengimplementasikan standar himpunan tipe dan operator titik mengambang IEEE-754. Keakuratan nilai untuk tipe data *floating point* adalah sebagai berikut:

Tabel 3
Tipe data *Floating Point*

Panjang Float	Tipe data	Nilai terbesar
32 bit	*Float*	3.40282e+38
64 bit	*Double*	1.79769e+308

Masing-masing tipe data *floating point* memiliki kebutuhan memori yang berbeda. Tipe data ***float*** memerlukan 32 bit sebagai *single-precision*, sedangkan tipe data ***double*** memerlukan 64 bit sebagai *double precision*.

Nama ***double*** mengacu pada presisinya yang sebesar dua kali dibandingkan ***float***. Presisi ***float*** kebanyakan tidak memadai untuk banyak aplikasi komputasi. Angka literal bertipe ***float*** berakhiran F, contoh 3.14F sedangkan kalau tidak diberi akhiran F akan dipandang sebagai bertipe ***double*.**

3. Tipe data *Char*

Tipe data ***char*** merupakan tipe data yang direpresentasikan dengan 16-bit *Unicodecharacter*. Literal dari *char* harus berada diantara *singlequotes* (‘ ‘)
Contohnya :

‘a’	huruf a
‘\t’	karakter tab

Unicode dirancang untuk menangani semua karakter di dunia dalam kode 2 byte. Kode 2 byte memungkinkan 65.536 karakter, dimulai dari nilai byte 0 sampai 65.535. Himpunan karakter ASCII dipandang sebagai bagian dari *Unicode* dan ditempatkan sebagai 256 karakter pertama dari *Unicode*. Terdapat pula beberapa barisan escape untuk karakter *Unicode* yang spesial, seperti berikut:

Tabel 4
Karakter *Unicode*

Barisan *Escape*	Nama	Nilai *Unicode*
\b	*Backspace*	\u008
\t	*Tab*	\u009
\n	*Linefeed*	\u00a
\r	*Carriagereturn*	\u00d
\”	Petik ganda	\u0022
\’	Petik tunggal	\u0027
\\	*Backslash*	\u005c

4 Tipe data Boolean

Tipe data *boolean* memiliki 2 literal yaitu : *true* dan *false*. Contoh, *State*men :
boolean truth = true;
mendeklarasikan variabel *truth* sebagai tipe data boolean dan memberikan nilai *true*

5. Tipe data komposit

Tipe data komposit merupakan tipe data yang disusun dari tipe data sederhana atau tipe komposit lain yang sudah ada. Tipe ini dapat berupa *array, string,* kelas, dan *interface*.
Khusus untuk *String* pada *Java* dikenali sebagai kelas, bukan sebagai *array of character*. String pada *Java* diapit oleh tanda petik ganda (“.....”), contoh:
String s=”Saya makan nasi”;

Operator di Java

Java memiliki beberapa jenis operator di antaranya:

- *Operator unary:*

Tabel 5
O*peratorunary*

Nama Operator	Simbol	Definisi
Increment	++	Akan menambahkan nilai sejumlah satu
Decrement	--	Akan mengurangi nilai sejumlah satu

Contoh penggunaan:
int x = 5;
int y = x++;

Pada kasus di atas nilai y akan berisi 5 dan nilai x akan berisi 6 karena nilai y akan mengambil nilai x dahulu setelah itu baru nilai x ditambahkan satu, berbeda kasusnya pada contoh di bawah ini:
int x = 5;
int y = ++x;

Pada kasus di atas, nilai y akan berisi 6 dan x berisi 6 karena nilai x akan ditambahkan satu dahulu baru kemudian dimasukkan ke variabel y.

- *Operator aritmatika:*

Tabel 6
Operator aritmatika

Nama Operator	Simbol	Deskripsi
Penambahan	+	Menambahkan dua buah nilai
Pengurangan	-	Pengurangan dua buah nilai
Perkalian	*	Perkalian dua buah nilai
Pembagian	/	Pembagian dua buah nilai
Sisa bagi	%	Sisa pembagian dua buah nilai

- *Operator relasi:*

Tabel 7
Operator relasi

Simbol	Deskripsi
<	Kurang dari
>	Lebih dari
<=	Kurang dari atau sama dengan
>=	Lebih dari atau sama dengan
==	Sama dengan
!=	Tidak sama dengan

- *Operator boolean:*

Tabel 8
Operator Boolean

Simbol	Deskripsi
&&	AND
||	OR
^	XOR
!	NOT

Hasil dan Pembahasan

Program dapat di aplikasikan berbagai macam bahasa program dan atauran main sendiri-sendiri tidak boleh menyimpang dalam aturan tersebut dan harus sepesifik dalam penulisan syntak programnya. Dalam pembahasan basaha program *Java* yang harus sesuai struktur bahasa program *Java* dan sebagai jantungnya program untuk menyelesaikan suatu permasalahan tersebut maka ada beberapa perintah yang sering di gunakan dalam pembuatan program di antaranya :
Perintah : if…else, for…, while…do untuk memnuktikan algoritma dan contoh program adalah sebagai berikut :

Percabangan di Java

Percabangan di *Java* menggunakan dua jenis Sintaks:

1. Sintaks if

Sintaks *if-else*, sebagai berikut :

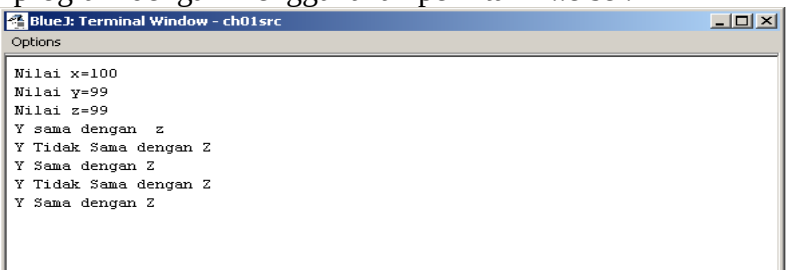
```
if (boolean expression) {
        Statement or block
} else if (boolean expression) {
        Statement or block
} else {
        Statement or block
}
```

Contoh syntak program dengan menggunakan perintah if—else

```
class oprelasi{
public static void main(String[] args){
int x,y,z;
x=100;
y=99;
```

```
z=99;
System.out.println("Nilai x="+x);
System.out.println("Nilai y="+y);
System.out.println("Nilai z="+z);
if(y==z){
      System.out.println("Y sama dengan  z");
   }else{
      System.out.println("Y Tidak Sama dengan Z");
   }
   if(x!=y){
      System.out.println("Y Tidak Sama dengan Z");
   }else{
      System.out.println("Y Sama dengan Z");
   }
   if(x<=y){
      System.out.println("Y Tidak Sama dengan Z");
   }else{
      System.out.println("Y Sama dengan Z");
   }
   if(x>=y){
      System.out.println("Y Tidak Sama dengan Z");
   }else{
      System.out.println("Y Sama dengan Z");
   }
     if(x<=y){
      System.out.println("Y Tidak Sama dengan Z");
   }else{
      System.out.println("Y Sama dengan Z");
   }}}
```

Hasil syntak program dengan menggunakan perintah if..else :



Gambar 4
Hasil syntak program perintah **for**

Perulangan di Java

Perulangan di *Java* menggunakan 3 jenis Sintaks:

1. Algoritma Perulangan for

Sintaks *for* sebagai berikut :

```
for (init_expr;boolean testexpr;alter_expr) {
      Statement or block
}
```

Contoh :

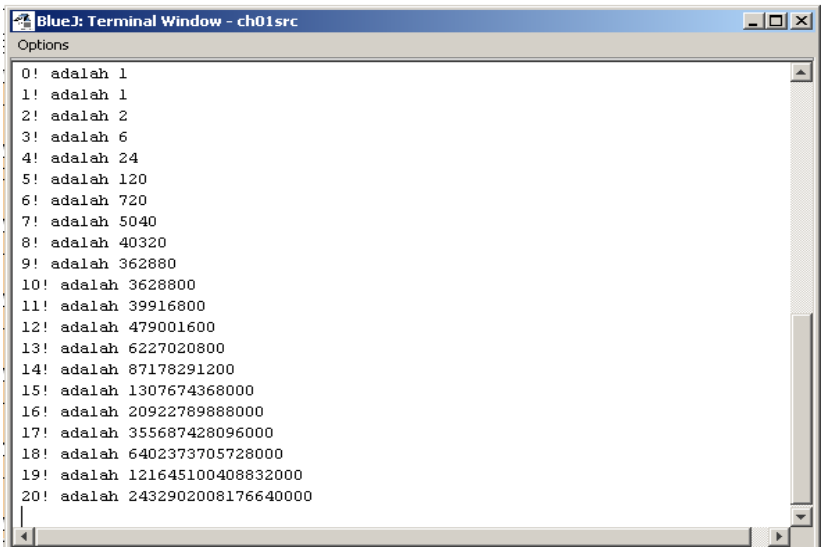
```
for (int i=0;i<10;i++) {
      Sistem.out.println(i);
}
```

Syntak program yang menggunakan perintah for..(perulangan).

```
public class faktorial
{
   public static void main(String[] args)
   {
      long limit=20;
      long faktorial=1;

for(int i=0;i<=limit;i++)
      {
         faktorial=1;
for(int faktor=2;faktor<=i;faktor++)
         faktorial *= faktor;
         System.out.println(i + "!" + " adalah " +faktorial);
      }}}
```

Hasil program perintah for tersebut diatas setelah di jalankan atau di run adalah :



Gambar 5
Hasil syntak program perintah **for**

2. Perulangan *while*

Sintaks *loopingwhile* sebagai berikut :

```
while (boolean testexpr) {
      Statement or block
}
```

Contoh :

```
int i = 0;
while (i<10) {
      Sistem.out.println(i);
      i++;
}
```

3. Perulangan *do....while*

Algoritma perulangan dengan perintah while … do
Sintaks do/*whileloop*, sebagai berikut

```
do {
    Statement or block
} while (boolean testexpr)
```

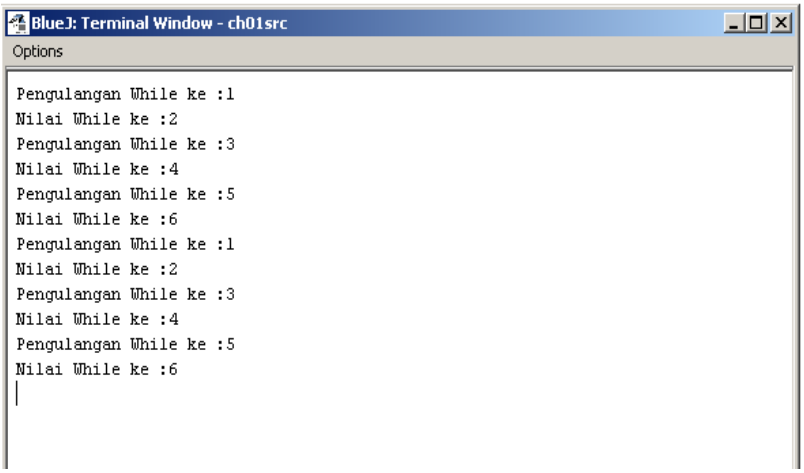
Contoh :

```
int i = 0;
do {
    Sistem.out.println(i);
    i++;
} while (i<10);
```

Program *Java* yang menggunakan perintah while(pengulangan)

```
public class loopwhile{
public static void main(String[] args){
boolean benar=true;
int a=1;
while(benar)
{
System.out.println("Pengulangan While ke :"+a++);
if(a==6){
benar=false;
}else{
benar=true;
}
if(a<=6){
System.out.println("Nilai While ke :"+a++);
}else{
System.out.println("Nilai While ke :"+a--);
}}}}
```

Hasil program setelah di jalankan:



Gambar 6
Hasil syntax program perintah while-do

Kesimpulan

Dalam pemrograman sehari-hari, terutama bagi programmer yang baru belajar dan bekerja dengan objek, penurunan kelas akan sering digunakan. Salah satunya adalah untuk memperluas kegunaan suatu kelas, yang disesuaikan dengan situasi dan kondisi permasalahan yang kita hadapi. Kita bisa membuat kelas baru yang merupakan turunan kelas yang sudah ada, menambah beberapa variabel dan metode instansi baru, yaitu dengan operator **extends** seperti dijelaskan sebelumnya. Untuk mempermudah dalam memahami bahasa program *Java* dengan membahas pemrograman berorentasi objek sesuai yang di jelaskan pada pembahasan di atas.Pemrograman sangat berpengaruh terhadap perkembangan teknologi saat ini yang sudah menyebar luas dalam kalangan dunia informatika saat dan di gemari oleh para mahasiswa Dalam meningkatakan kualitas pemrograman bahasa *Java*. Pemprograman *Java* sangat perpengaruh pada perkembangan teknologi saat ini untuk itu sangat dibutuhkan oleh seluruh mahasiswa dan dapat menciptakan hal yang baru sehingga semua kalangan tahap belajar sangat membutuhkan bahasa pemrograman.

Daftar Pustaka

Deitel, H.M. *Java* How To Program, sixth Edition. 2004, Prentice Hall, 2004

Exercise 10, Introduction to Computer Science Januari 2003, TU Darmstadt, 2003

H.M Deitel, *Java* How to Program, Sixth Edition. Prentice Hall, August, 2004

Rickyanto, Isak, *Dasar Pemrograman Berorientasi Objek*, Yogyakarta : Andy , 2002

Purnama, Rangsang, *Tuntutan Pemrograman Java*, Jakarta : Prestasi Pustaka Publisher
(Sumber: http://*Java*.sun.com/docs/books/tutorial/)

http://cs.bilgi.edu.tr/pages/standards_project/*Java*_CodingStyle.pdf

http://cs-education.stanford.edu/class/cs107/handouts/42SimpleInheritance.pdf

http://faculty.ksu.edu.sa/Shammami/Documents/DocCSC113/Lectures%20Notes/ChapterRelationship. pdf