



JAVA APPLETS (1)



Language Features

Simple

- Sintaks berdasar pada C++ (transisi lebih mudah bagi programmer)
- Menghilangkan feature yang jarang dipakai
contoh : explicit pointers, operator overloading, automatic coercions
- Penambahan memory management (mengacu pada count/garbage collection hybrid)

Object-oriented

- Fasilitas OOP mirip dengan C++
- OOP murni – seluruhnya adalah class, tidak ada independent functions

Robust

- ketiadaan pointers dan memori manajemen menghindari banyak error
- *libraries of useful, tested classes increases level of abstraction
arrays & strings are ADTs, well-defined interfaces

Portable

- byte kode akan run pada versi manapun dari Java Virtual Machine (JVM)



Language Features (cont.)

- Platform independence
 - Dapat menjalankan Java code pada multiple platforms
 - Netralitas dicapai dengan mencampur compilation & interpretation
 - Java programs diterjemahkan ke dalam *byte code* oleh Java compiler
 - byte code is a generic machine code
 - byte code kemudian dieksekusi oleh interpreter (Java Virtual Machine)
 - must have a byte code interpreter for each hardware platform
 - Sebuah Applet adalah sebuah special form dari Java application
 - byte code di download pada page, JVM di-embedded pada browser
- Architecture-neutral
 - Tidak ada implementation bergantung pada features (contoh : size dari primitive types)
- High-performance
 - Lebih cepat daripada traditional interpretation karena byte code mendekati native code
 - Masih sedikit lebih lambat daripada compiled language (contoh : C++)



Language Features (cont.)

Distributed

- Extensive libraries untuk mengatasi TCP/IP protokol seperti HTTP & FTP
- Aplikasi Java dpt mengakses remote URL'S sama seperti halnya mengakses file lokal

Multi-threaded

- Sebuah *thread* seperti sebuah program terpisah, dijalankan secara berbarengan
- Dapat menulis program Java dimana beberapa pekerjaan dpt dilakukan sekaligus dgn mendefinisikan multiple threads (same shared memory, but semi-independent execution)
- Threads penting untuk multi-media, web applications

Secure

- Aplikasi Java applications tidak dapat langsung mengakses ke lokasi memory
 - Akses memory adalah virtual, dipetakan oleh JVM ke lokasi fisik
 - Downloaded applet tidak dapat membuka, membaca atau menyalin local files
- JVM memeriksa autentifikasi dari class juga memeriksa autentifikasi dari class yg di-load
- *Sun meng-klaim: Model execution memungkinkan virus-free*, tamper-free* systems*



Bagaimana Java Bekerja...!

Compile-time Environment

Java Source
(.java)

Java
Compiler

Java
Bytecode
(.class)

Java
Bytecodes
move locally
or through
network

Compile-time Environment

Class
Loader
Bytecode
Verifier

Java
Class
Libraries

Java
Interpreter

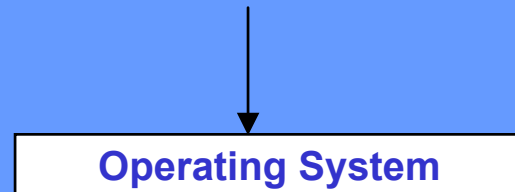
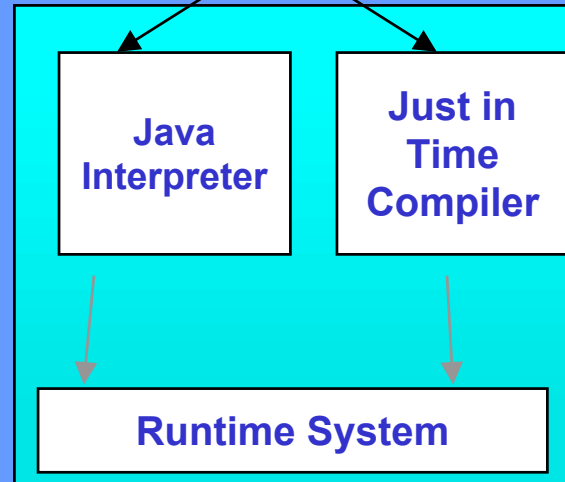
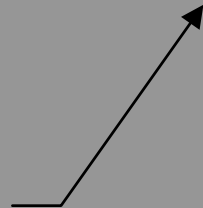
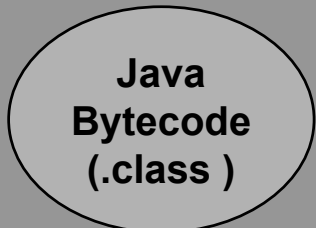
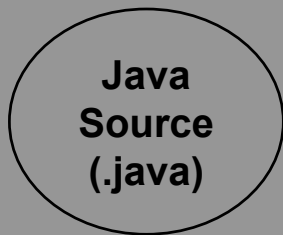
Just in
Time
Compiler

Java
Virtual
machine

Runtime System

Operating System

Hardware





Bagaimana Java Bekerja ...! (lanj)

Java independent hanya untuk satu alasan :

- Hanya bergantung pada Java Virtual Machine (JVM),
- Code dikompiled ke *bytecode*, yang di-interpreted oleh resident JVM,
- JIT (just in time) compilers mencoba untuk meningkatkan kecepatan.



Keamanan - Java

- Pointer denial – mengurangi kesempatan virulent programs merusak host
- Applets lebih terbatas lagi -
 - Tidak bisa
 - Menjalankan local executables,
 - Read atau write ke local file system,
 - Berkomunikasi dengan beberapa server lain selain dengan originating server.



Object-Oriented

- Java mendukung OOD
 - Polymorphism
 - Inheritance
 - Encapsulation
- Program Java berisi tak lain hanya definisi dan instantiation dari class
 - Semuanya di-encapsulate dalam sebuah class!



Keuntungan Java

- Portable - Write Once, Run Anywhere
- Keamanannya sudah dipikirkan secara matang
- Memory management sempurna
- Didesain untuk network programming
- Multi-threaded (berbagai tugas simultaneous)
- Dynamic & extensible (loads of libraries)
 - Class disimpan pada file yang terpisah
 - Loaded hanya jika dibutuhkan



Java Programming Models

- Java *applications* are stand-alone programs
 - Harus dikompilasi menjadi Java byte code dengan Java compiler
 - Dieksekusi oleh sebuah interpreter (Java Virtual Machine)
- Java *applets* provide for client-side programming
 - dikompilasi menjadi Java byte code, kemudian di-*downloaded* sebagai bagian dari sebuah *Web page*
 - Dieksekusi oleh JVM *embedded* dalam *Web browser*
 - Tidak seperti JavaScript, Java full-featured dengan extensive library support
 - Java dan APIs-nya telah menjadi standard industri
 - Pendefinisian language dikontrol oleh Sun, untuk meyakinkan compatibility
 - Applications Programming Interfaces standardize the behavior of useful classes and libraries of routines*
- Java *servlets* provide similar capabilities on the server-side
 - Merupakan alternative dari CGI programs, lebih terintegrasi dengan Web server



Java Applets

- Important point : Java applets & applications look different!
 - Jika ingin mendefinisikan stand-alone application, buat sebuah aplikasi membutuhkan `public static void main` function, sama dengan C++ main
 - Jika ingin meng-embed code pada sebuah Web page, buat sebuah applet membutuhkan `public void paint`, `public void init`, ...
- As with JavaScript, security is central
 - Ketika sebuah Java applet di-downloaded, pemeriksa bytecode dari JVM memeriksa agar dapat melihat pada saat bytecode berisi byte yang terbuka, terbaca dan tertulis dalam lokal disk.
 - Java applet dapat membuka sebuah window baru dengan Java logo untuk pencegahan yaitu dengan menyembunyikan system windows (contohnya: pencurian passwords)
 - Java applet tidak membolehkan untuk terhubung ke server lain kecuali ke host.
 - Kondisi yang aman/secure ini disebut *sand box model*



First Java Applet

```
import java.awt.*;
import java.applet.*;

/**
 * This class displays "Hello world!" on the applet window. */
public class HelloWorld extends Applet
{
    public void paint(Graphics g)
    {
        g.drawString("Hello world!", 10, 10); // writes starting 10 pixels over & down
    }
}
```

libraries: Java menyediakan provides extensive library support dalam bentuk class

- Library dipanggil menggunakan import (mirip dengan `#include` di C++)

`java.awt`: berisi contains Abstract Window Toolkit (untuk GUI classes & routines)

`java.applet`: berisi definisi applet class

Comments : `//` dan `/* */` berfungsi sama seperti pada C++

`/** */` menandakan komentar



First Java Applet

```
import java.awt.*;
import java.applet.*;
/**
 * This class displays "Hello world!" on the applet window. */
public class HelloWorld extends Applet
{
    public void paint(Graphics g)
    {
        g.drawString("Hello world!", 10, 10); // writes starting 10 pixels over & down
    }
}
```

Pendefinisian class di Java

- Sama dengan pada C++ (tetapi tidak ada titik koma di akhir)
Dapat berisi instance variables (data fields) & methods(member functions)
Didahului dengan pendefinisian class & method dengan *public* untuk membuatnya tersedia bagi semua program
- Tidak ada fungsi stand-alone di Java*
- Harus disimpan pada sebuah file dengan nama yang sama dengan ekstension .java
Contoh : `HelloWorld.java`



First Java Applet

```
import java.awt.*;
import java.applet.*;
/**
 * This class displays "Hello world!" on the applet window. */
public class HelloWorld extends Applet
{   public void paint(Graphics g)
    {   g.drawString("Hello world!", 10, 10);   // writes starting 10 pixels over & down
    }
}
```

Seluruh applets mewarisi dari Applet class (pada java.applet)

default methods termasuk :

- `init()` : memanggil saat page di-load untuk membuat/inisialisasi variables
by default, does nothing
- `paint(Graphics g)` : called to draw (after init) or redraw (after being obscured)
here, the paint method is overridden to display text on the applet window



Embedding Applet di HTML

to include an applet in a Web page, use either

- **APPLET** tag (deprecated)

CODE menentukan applet name, HEIGHT dan WIDTH menentukan window size
text antara APPLET tags ditampilkan jika tidak dapat dieksekusi (e.g., Java not enabled)

- **OBJECT** tag

Lebih dipilih HTML 4, tetapi tidak secara universal mendukung

```
<html>
<!-- COMP519      hello1.html      18.09.2005 -->
<head> <title>Hello World Page</title> </head>
<body>
<p>
  <applet code="HelloWorld.class" height=100
    width=100>
    You must use a Java-enabled browser to view this
    applet.
  </applet>
</p>
</body> </html>
```



HTML & Applets

```
<html>
<!-- COMP519      hello2.html      18.09.2005  -->
<head>
  <title>Hello World Page</title>
</head>
<body>

<p>
<div align="center">
<table border=1>
<tr><td>

<applet code="HelloWorld.class" height=200
  width=200>
  You must use a Java-enabled browser to view this
  applet.
</applet>
</td></tr>
</table>
</div>
</p>

</body>
</html>
```

Sebuah applet dapat di-embedded dalam HTML elements seperti element lainnya

Berguna untuk format dan layout



Parameters di HTML

```
<html>
<!-- COMP519      hello3.html      18.09.2005 -->
<head>
<title>Hello World Page</title>
</head>
<body>
<p>
<div align="center">
<table border=1>
<tr><td>
<applet code="HelloWorld1.class" height=35 width=300>
  <param name="name" value="Chris">
  <param name="age" value=20>
  You must use a Java-enabled browser to view this applet.
</applet>

</td></tr>
</table>
</div>
</p>

</body> </html>
```

Dapat menentukan parameter **APPLET** ketika di-embedded di HTML

- setiap parameter harus mempunyai **PARAM** tag sendiri dalam **APPLET** element
- Menentukan parameter name dan value



Applet Parameters

```
import java.awt.*;
import java.applet.*;

/**
 * This class displays a message based on parameters. */
public class HelloWorld1 extends Applet
{
    public void paint(Graphics g)
    {
        String userName = getParameter("name");
        int userAge = Integer.parseInt(getParameter("age"));

        String message1 = "Hello " + userName + ".";
        String message2 = "On your next birthday, you will be " +
            (userAge+1) + " years old.";

        g.drawString(message1, 10, 10);
        g.drawString(message2, 10, 30);
    }
}
```

can access parameters passed in from the HTML document

getParameter mengakses nilai dari parameter (must know its name)

- Jika parameter ditunjukkan angka, harus **parseInt** atau **parseFloat**



Java Constructs

- Akan dikenali oleh C/C++ programmers
- Tipe nama yang indentik; Tipe yang dijamin untuk diartikan secara benar dan teliti
- Source dalam .java files, compiled code dalam .class files; downloaded (biasanya) dalam .jar files (Java archive)

Java Development

- Langkah-langkah Pengembangan
 - Buat source files di editor
 - Kompile menggunakan *javac*
 - Jalankan/test menggunakan *java*



Primitive Types dan Variables

- Boolean, char, byte, short, int, long, float, double dsb.
- Tipe dasar ini adalah satu-satunya tipe yang bukan objects
- Ini berarti bahwa kita tidak menggunakan operator baru untuk membuat primitive variable.
- Pendeklarasian primitive variables:

```
float initVal;
```

```
int retVal, index = 2;
```

```
double gamma = 1.2, brightness
```

```
boolean valueOk = false;
```



Initialisation

- Jika tidak ada nilai di berikan sebelumnya untuk digunakan, compiler akan memberikan kesalahan
- Java men-set primitive variables menjadi zero atau false pada kasus dari sebuah boolean variable
- Seluruh object references awalnya di-set null
- Sebuah array adalah sebuah object
 - Set null pada deklarasi
 - Elements to zero false or null on creation



Declarations

```
int index = 1.2;           // compiler error
boolean retOk = 1;        // compiler error
double fiveFourths = 5 / 4; // no error!
float ratio = 5.8f;       // correct
double fiveFourths = 5.0 / 4.0; // correct
```

- 1.2f adalah float value akurasi sampai 7 decimal places.
- 1.2 adalah double value akurasi sampai 15 decimal places.



Assignment

- All Java assignments are right associative

```
int a = 1, b = 2, c = 5
```

```
a = b = c
```

```
System.out.print(
```

```
“a= “ + a + “b= “ + b + “c= “ + c)
```

- What is the value of a, b & c
- Done right to left: `a = (b = c);`



Basic Mathematical Operators

- `*` / `%` + `-` are the mathematical operators
- `*` / `%` have a higher precedence than `+` or `-`

```
double myVal = a + b % d - c * d / b;
```

- Is the same as:

```
double myVal = (a + (b % d)) - ((c * d) / b);
```

Statements & Blocks

- Sebuah statement sederhana adalah pernyataan yang diakhiri titik koma A :
`name = "Fred";`
- Sebuah block adalah gabungan pernyataan yang ditutup dalam tanda kurung kurawal :

```
{  
    name1 = "Fred"; name2 = "Bill";  
}
```
- Blocks dapat terdiri dari beberapa blocks



Flow of Control

- Java menjalankan satu statement berurutan sesuai urutan penulisan
- Beberapa statement Java adalah statement flow of control :

Alternation : if, if else, switch

Looping : for, while, do while

Escapes : break, continue, return

If – The Conditional Statement

- Statement if meng-evaluasi sebuah expression & jika evaluasi benar maka tindakan tertentu akan dijalankan. Misal jika nilai dari x lebih kecil dari 10, maka x sama dengan 10

```
if ( x < 10 ) x = 10;
```

- Hal ini dapat ditulis seperti :

```
if ( x < 10 )
```

```
x = 10;
```

- Atau, alternative lain :

```
if ( x < 10 ) { x = 10; }
```



Relational Operators

==	Equal (careful)	!=	Not equal
>=	Greater than or equal	<=	Less than or equal
>	Greater than	<	Less than

If... else

- The if ... else statement evaluates an expression and performs one action if that evaluation is true or a different action if it is false.

```
if (x != oldx) {  
    System.out.print("x was changed");  
}  
else {  
    System.out.print("x is unchanged");  
}
```



Nested if ... else

```
if ( myVal > 100 ) {  
    if ( remainderOn == true) {  
        myVal = mVal % 100;  
    }  
    else {  
        myVal = myVal / 100.0;  
    }  
}  
else  
{  
    System.out.print("myVal is in range");  
}
```



else if

- Berguna untuk memilih beberapa alternative :

```
if ( n == 1 ) {  
    // execute code block #1  
}  
else if ( j == 2 ) {  
    // execute code block #2  
}  
else {  
    // if all previous tests have failed, execute  
    code block #3  
}
```



Peringatan...

WRONG!

```
if( i == j )
    if ( j == k )
        System.out.print("i equals k");
    else
        System.out.print("i is not equal to j");
```

CORRECT!

```
if( i == j ) {
    if ( j == k )
        System.out.print("i equals k");
}
else
    System.out.print("i is not equal to j"); // Correct!
```



Switch Statement

```
switch ( n ) {  
    case 1:  
        // execute code block #1  
        break;  
    case 2:  
        // execute code block #2  
        break;  
    default:  
        // if all previous tests fail then  
        //execute code block #4  
        break;  
}
```



for loop

- Loop n times

```
for ( i = 0; i < n; n++ )  
{  
    // this code body will execute n times  
    // ifrom 0 to n-1  
}
```

- Nested for:

```
for ( j = 0; j < 10; j++ ) {  
    for ( i = 0; i < 20; i++ ){  
        // this code body will execute 200 times  
    }  
}
```



while loops

```
while(response == 1) {  
    System.out.print( "ID =" + userID[n]);  
    n++;  
    response = readInt( "Enter " );  
}
```

do {... } while loops

```
do {  
    System.out.print( "ID =" + userID[n] );  
    n++;  
    response = readInt( "Enter " );  
} while (response == 1);
```

Berapa kali paling sedikit loop di-executed?

Berapa kali paling banyak loop di-executed?



Break

- Break statement menyebabkan keluar dari innermost yg berisi while, do, for or switch statement.

```
for ( int i = 0; i < maxID, i++ ) {  
    if ( userID[i] == targetID ) {  
        index = i;  
        break;  
    }  
} // program jumps here after break
```

Continue

- Hanya dapat digunakan dengan while, do atau for.
- Continue statement menyebabkan innermost loop mulai perulangan berikutnya

```
for ( int i = 0; i < maxID; i++ ) {  
    if ( userID[i] != -1 ) continue;  
    System.out.print( "UserID " + i + " :" + userID);  
}
```



Classes ARE Object Definitions

- OOP - object oriented programming
- code dibangun dari object-object
- Java seperti ini disebut **classes**
- Setiap pendefinisian class di-codekan dalam file .java yang berbeda
- Nama dari setiap object harus sama dengan nama class/object