

Bahasa Pemrograman Java

Pengenalan

by: Yohanes Nugroho

rev: Achmad Imam Kistijantoro,

Saiful Akbar,

Yani Widayani.

Mengenal Java

***Nama Java, Bahasa Pemrograman
Java, API***

Perkenalan Java

- Nama Java
- Java Virtual Machine
- Application Programming Interface

Java adalah...



- Nama bahasa pemrograman
- Nama platform tempat menjalankan program Java, meliputi
 - Virtual Machine
 - API (Application Programming Interface)
- Nama Java sendiri diambil dari Kopi Jawa yang sangat terkenal di kalangan pegawai Sun Microsystem

Sejarah singkat Java ...



- Dulu nama bahasa Java adalah Oak
 - Ternyata namanya sudah ada yang memakai (menurut kantor merk dagang Amerika Serikat)
 - Nama berubah menjadi Java
- Beberapa fakta:
 - Oak sudah mulai dibuat sejak tahun 1991
 - Oak tadinya ditujukan untuk consumer device (television set-top box, mesin cuci, ponsel, dll)
 - Kemudian Web/WWW menjadi populer, yang mempopulerkan Java dan Applet

Five primary goals in the creation of the Java language

- It should be "simple, object oriented, and familiar".
- It should be "robust and secure".
- It should be "architecture neutral and portable".
- It should execute with "high performance".
- It should be "interpreted, threaded, and dynamic".

Bahasa Pemrograman Java

- Bahasa pemrograman Java (untuk selanjutnya disebut bahasa Java) merupakan bahasa dengan sintaks yang mirip C++ tanpa fitur yang kompleks
- Umumnya program dalam bahasa Java dikompilasi menjadi bentuk yang dinamakan bytecode (tidak dalam bahasa mesin *native*)
 - Seperti bahasa assembly, tapi untuk suatu virtual machine
 - bytecode ini dijalankan di Java Virtual Machine
- Bahasa Java dirancang sebagai bahasa yang mendukung OOP

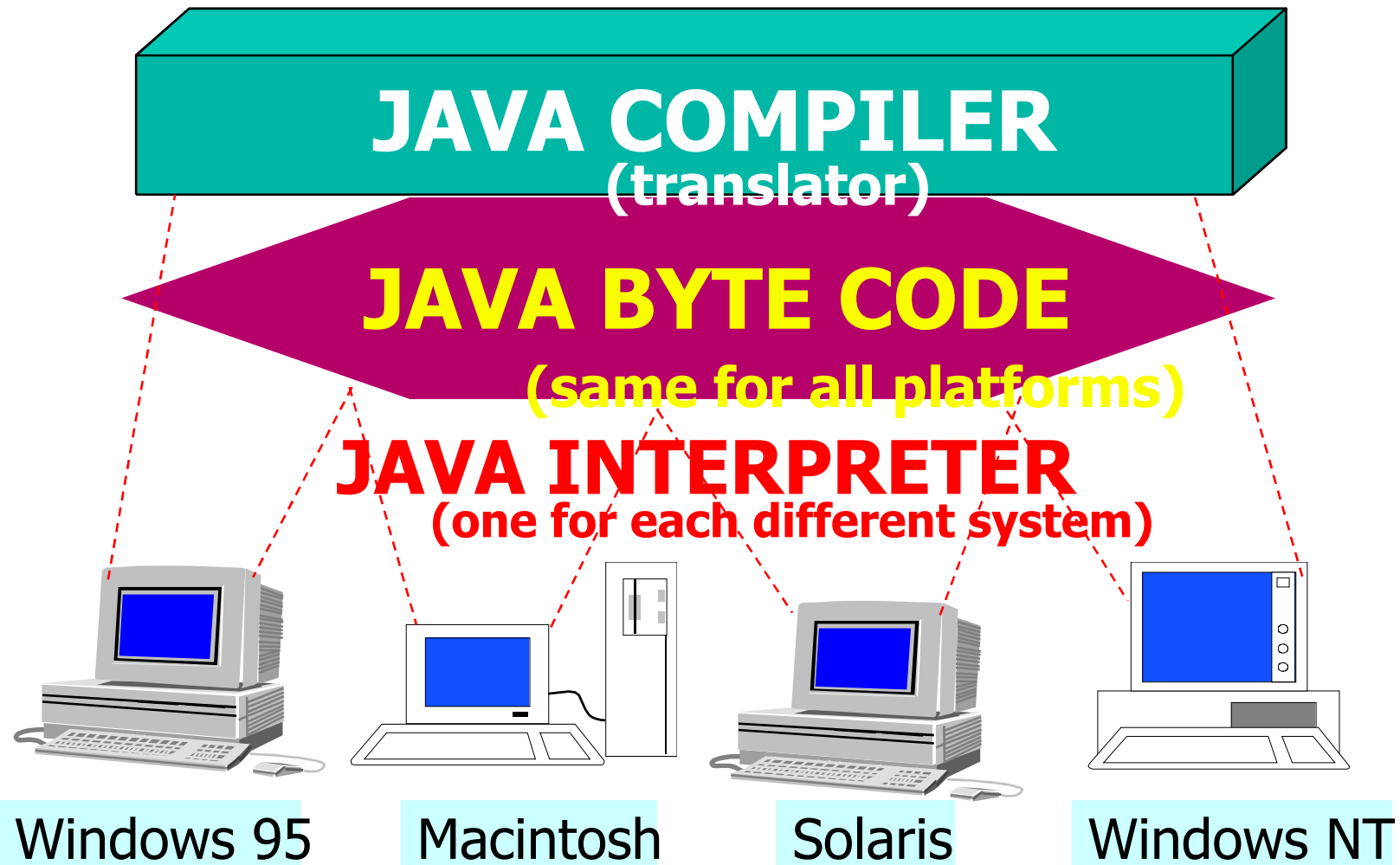
Java Virtual Machine (JVM)

- JVM adalah suatu program yang menjalankan program Java
 - Tepatnya, JVM menjalankan bytecode dengan menginterpretasi bytecode
- Jika tersedia JVM untuk suatu sistem operasi atau device tertentu, maka Java bisa berjalan di sistem komputer tersebut
- Semboyan Java: *"Write Once Run Anywhere"*

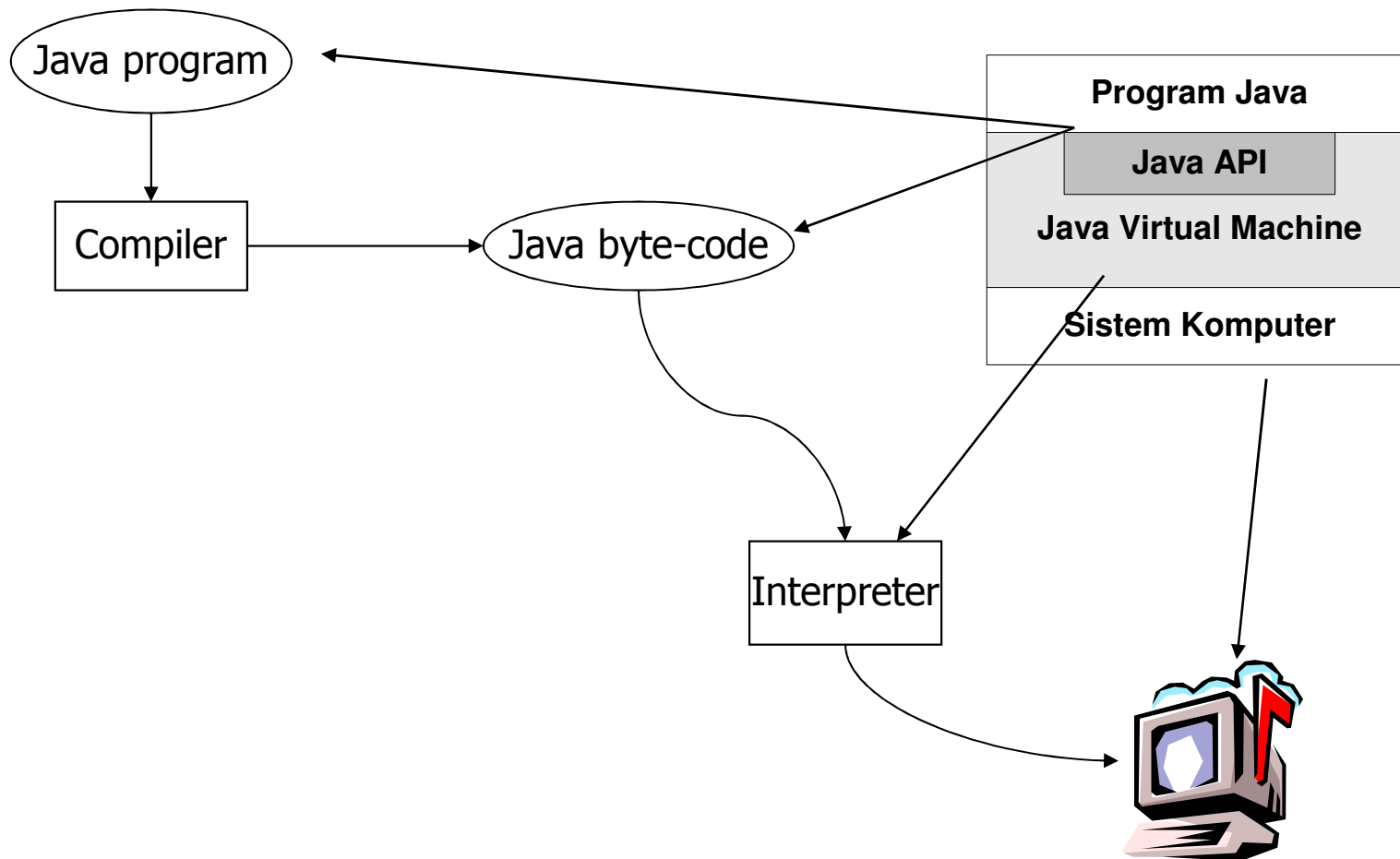
Application Programming Interface

- € Suatu bahasa pemrograman hanya mendefinisikan sintaks dan semantik bahasa tersebut
 - Fungsi-fungsi dasar di suatu bahasa pemrograman disediakan oleh library, misal `printf` di C disediakan oleh library C (bukan oleh bahasa C)
- € Di Java sudah tersedia kumpulan fungsi (dalam Kelas tentunya, karena Java berparadigma OO) yang disebut sebagai Java API
 - Fungsi ini dijamin ada pada setiap implementasi platform Java

Total Platform Independence



Platform Java



Lingkup kuliah

- Meliputi:
 - Bahasa Pemrograman Java
 - Pembahasan didasarkan pada konsep OOP yang sudah diberikan
 - Sedikit API Java
- Sedangkan yang tidak diajarkan
 - Internal JVM
 - API Java yang kompleks
 - Pemrograman Java untuk server

Bahan Bacaan

- Spesifikasi Bahasa Java (The Java Language Specification)
- Java Tutorial
- Dokumentasi API Java
- Semua bisa dilihat di:
<http://java.sun.com>

Hello World

Mengenai Lingkungan Pemrograman Java

Overview Hello World


- Mengerti program hello world
- Entry point program Java
- Mengkompilasi dan menjalankan program Java

Hello World dalam Java

★ Program Java sederhana:

```
class HelloWorld {  
    static public void main(String args[]) {  
        System.out.println("Hello world!");  
    }  
}
```

main: titik awal program



- Nama file harus sama dengan nama kelas

Mengkompilasi dan Menjalankan

- Kompilasi

```
javac HelloWorld.java
```

- Perhatikan suffiks .java
- Jika berhasil, akan terbentuk file HelloWorld.class

- Menjalankan

```
java HelloWorld
```

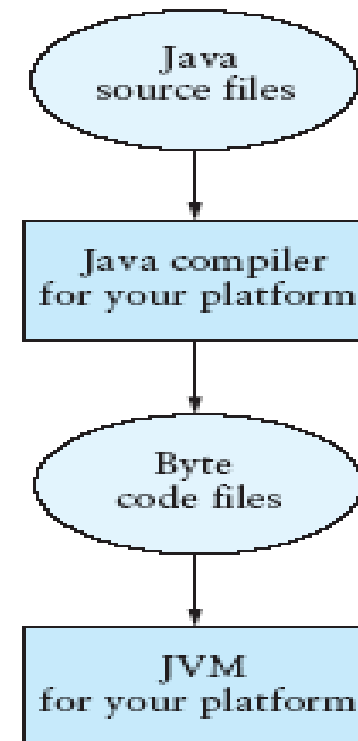
- Perhatikan, tanpa suffiks .class

Gambaran Proses Kompilasi dan Run

- Source code diproses oleh kompilator Java
 - Menghasilkan .class
- File .class diproses oleh JVM
 - Dijalankan

FIGURE A.1

Compiling and Executing a Java Program



Penjelasan Hello World

- Semua program Java merupakan kumpulan kelas
 - Program “hello world” juga merupakan sebuah kelas
- Pada C/C++ entry point adalah main, di Java entry point adalah: method main di sebuah kelas
 - Karena di setiap kelas boleh memiliki main, maka pada sebuah aplikasi (yang punya banyak kelas) boleh ada banyak main (kita/user memilih main yang mana yang dijalankan)

Definisi Main

- Main harus didefinisikan sebagai
 - Method yang publik (bisa diakses dari luar kelas)
 - Method yang statik (bisa dijalankan tanpa instan kelas)
 - Mengembalikan void (bukan int seperti di C)
 - Memiliki parameter (`String arg[]`) yang merupakan parameter dari user
 - Di C: `int main(int argc, char *argv[]);`
 - Array di Java punya informasi panjang: `arg.length` seperti `argc` di C
 - Elemen dari `arg` sama seperti `char *argv[]` di C

Output hello world

- Baris utama:

```
System.out.println("Hello World");
```

- `System` merupakan nama kelas di Java (kelas standar/default)
- `out` merupakan objek dari kelas `PrintWriter`
 - kelas `PrintWriter` akan dijelaskan kemudian pada konsep I/O Java)
- `out` memiliki method `println()` yang mengambil parameter `String`

Dasar Bahasa Java

Tipe Dasar, Loop, Kondisional

Dasar Bahasa Java

- Tipe Primitif dan Reference
- Pendefinisian “variabel”
- Operator
- Kondisional
- Loop

Tipe dasar/primitif

- Tipe dasar/primitif adalah tipe bawaan bahasa Java yang bukan merupakan sebuah kelas
- Java memiliki beberapa tipe dasar seperti di C/C++
 - int (32 bit)
 - long (64 bit)
 - byte (signed 8 bit)
 - char (16 bit UNICODE, tidak seperti C/C++ yang merupakan 8 bit ASCII)
 - float, double

Tipe primitif & reference

★ Primitif:

- tipe dasar seperti char, int, long, byte, float, double
- memori untuk variabel tipe primitif dialokasikan pada saat variabel tersebut dideklarasikan
- operasi assignment pada variabel primitif menghasilkan pengkopian nilai

★ Reference:

- mempunyai semantik serupa dengan pointer pada C/C++
- memori untuk variabel tipe reference tidak dialokasikan pada saat deklarasi, alokasi dilakukan eksplisit dengan operator `new`
- operasi assignment pada variabel reference menghasilkan pengkopian reference (tetap mengacu pada object yang sama)

Perbedaan tipe primitif & reference

```
class Value { int val; }
class Test {
    public static void main(String[] args) {
        int i1 = 3;
        int i2 = i1; // i1 & i2 variabel berbeda dengan nilai sama
        i2 = 4;
        System.out.print("i1==" + i1);
        System.out.println(" but i2==" + i2);
        Value v1 = new Value();
        v1.val = 5;
        Value v2 = v1; // v1 & v2 mengacu ke variabel yg sama
        v2.val = 6;
        System.out.print("v1.val==" + v1.val);
        System.out.println(" and v2.val==" + v2.val);
    }
}
```

Range tipe primitif

Data type	Range of values
byte	-128 .. 127 (8 bits)
short	-32,768 .. 32,767 (16 bits)
int	-2,147,483,648 .. 2,147,483,647 (32 bits)
long	-9,223,372,036,854,775,808 (64 bits)
float	$\pm 10^{-38}$ to $\pm 10^{+38}$ and 0, about 6 digits precision
double	$\pm 10^{-308}$ to $\pm 10^{+308}$ and 0, about 15 digits precision
char	Unicode characters (generally 16 bits per char)
boolean	True or false

Pendefinisian data: tipe & variabel

- ★ variabel harus dideklarasikan dan dialokasikan dahulu sebelum digunakan
- ★ deklarasi: menyatakan tipe variabel tersebut.
- ★ alokasi: pengadaan area memori untuk menampung nilai variabel
- ★ contoh (Java):

```
int X; // variabel dengan nama X, bertipe integer
char myvarChar; // variabel dengan nama myvarChar
String str; // variabel dengan nama str
```
- ★ variabel bertipe non primitif harus alokasi eksplisit:

```
String str;
str = new String("my string");
str = "str 2"; // khusus tipe string, operasi = otomatis alokasi
```

Operator dalam Bahasa Java

- Sifat operator Java sebagian besar sama dengan C/C++:
 - Lihat slide berikut
- Operator berikut ini hanya ada di Java (tambahan):
 - Perbandingan: `instanceof`
 - Bit: `>>>` (*unsigned shift*)
 - Assignment: `>>>=`
- String: `+` penggabungan string
- Operator baru yang ada di Java hanya sedikit dan jarang dipakai (kecuali penggabungan string dengan `+`), sehingga tidak perlu khawatir akan lupa

Operator Java yang sama dengan C/C++

- Matematik: +, -, *, /, % (modulus), unary + -
- Perbandingan: ==, !=, <, >, <=, >=,
- Boolean: &&, ||, !
- Bit: &, |, ~, <<, >>
- Ternary: cond?true-expr:false-expr
- Assignment: =, += -= *= /= <<= >>= &= |=

Operator baru (dibanding C++)

- Ada dua operator yang baru (jika dibandingkan dengan C++) yaitu `>>>` dan `instanceof`
- `>>>`
 - unsigned shift right, sign bilangan (bit terkiri) juga di-*shift* ke kanan
- `x instanceof b`
true jika x (objek) adalah instans dari kelas b

Operasi perbandingan pada primitif

- Operator perbandingan (`==`, `<`, `>`, dll) nilai primitif membandingkan nilai primitif tersebut
- Sifatnya sama dengan C/C++
- Contoh:

```
int a = 5;
```

```
int b = 5;
```

```
if (a==b) { /* Sama */ }
```


Operasi perbandingan pada objek

- Operator `==` terhadap reference membandingkan reference (bukan isi objek)
- Method `.equals()` digunakan untuk membandingkan kesamaan isi objek (termasuk juga objek `String`)

```
String a = "Hello";  
String b = "World";  
if (a.equals(b)) { /*String sama*/ }
```

- Jangan membandingkan string dengan operator `==`

Kondisional

- Java memiliki sintaks `if` dan `switch` yang sama dengan C/C++
- Di Java `integer` tidak sama dengan `boolean`
- Perhatikan bahwa hal berikut tidak boleh

```
int a = 1;
if (a) return;
//integer tdk bisa dikonversi ke boolean
```
- Di Java seharusnya:

```
if (a!=0) return;
```

Loop

- Java memiliki sintaks loop `while`, `for`, `do while` yang sama dengan C/C++
 - Perlu diingat bahwa boolean tidak sama dengan integer di Java
- Di Java 5 ada sintaks loop baru (akan dijelaskan pada materi lain)

Semua di Java adalah Reference

- Java tidak mengenal pointer
 - Semua operasi dan operator pointer yang ada di C/C++ tidak bisa dilakukan di Java (&, *, aritmatika pointer)
- Semua objek di Java berlaku sebagai reference (sifatnya mirip pointer, tapi tanpa * dan &)
- Objek tidak bisa dipertukarkan dengan tipe dasar. Tapi di JDK 1.5 ada autoboxing/unboxing

- <http://docs.oracle.com/javase/tutorial/java/data/autoboxing.html>

Primitive type	Wrapper class
boolean	Boolean
byte	Byte
char	Character
float	Float
int	Integer
long	Long
short	Short
double	Double

Object Wrapper

```
// Create wrapper object for each primitive type Boolean
refBoolean = new Boolean(true);
Byte refByte = new Byte((byte)123);
Character refChar = new Character('x');
Short refShort = new Short((short)123);
Integer refInt = new Integer(123);
Long refLong = new Long(123L);
Float refFloat = new Float(12.3F);
Double refDouble = new Double(12.3D);

// Retrieving the value in a wrapper object
boolean bool = refBoolean.booleanValue();
byte b = refByte.byteValue();
char c = refChar.charValue();
short s = refShort.shortValue();
int i = refInt.intValue();
long l = refLong.longValue();
float f = refFloat.floatValue();
double d = refDouble.doubleValue();
```

OOP di Java

Kelas, Objek, Penurunan

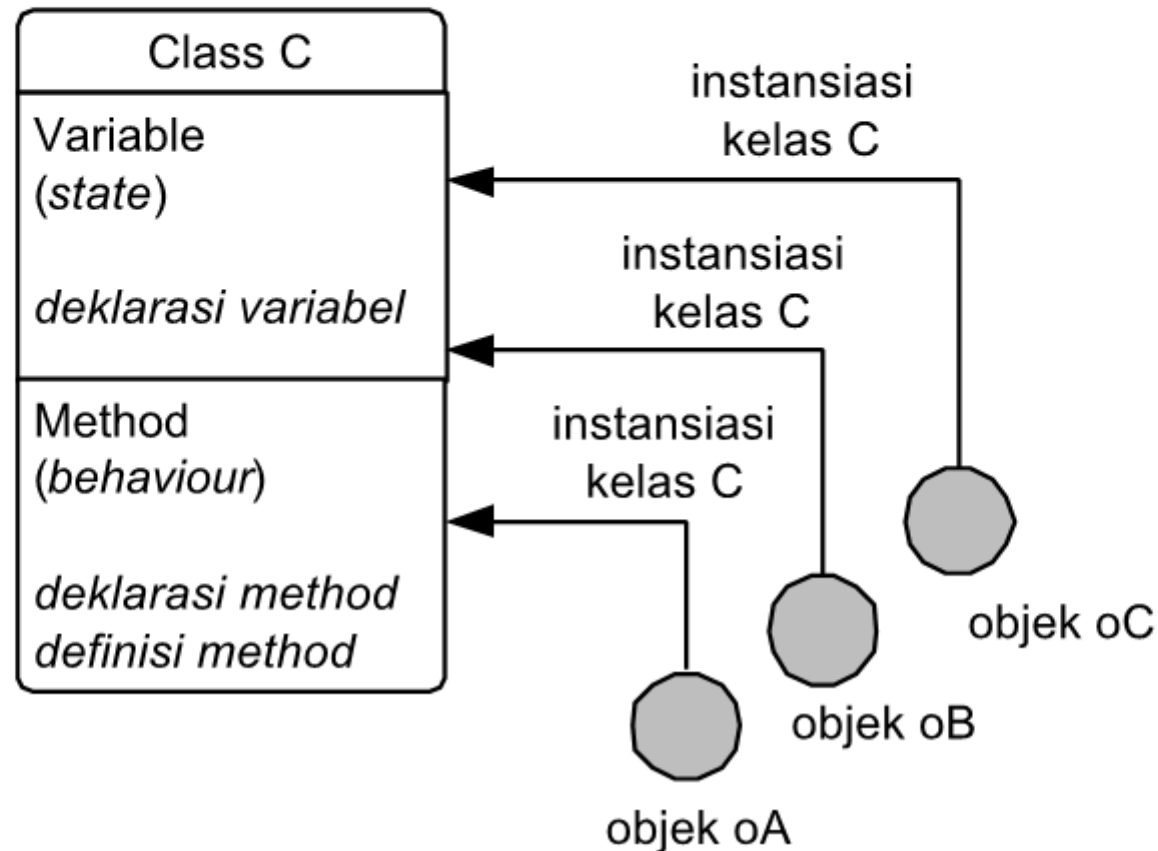
OOP Dengan Java

Bagian-1

- Kelas
- Instansiasi Objek

OO Programming

- ★ Objek: satuan unit, memiliki state & behavior
- ★ Kelas: definisi statik dari objek, menyatakan type objek
- ★ Objek adalah instance dari suatu Kelas



Kelas di Java

- Kelas dituliskan dengan keyword `class`, dengan isi kelas menyatu dengan deklarasinya
 - Di C++ deklarasi dan definisi boleh dipisah boleh disatukan
- Access modifier harus ditulis untuk setiap member (baik data maupun method)

Perhatikan kelas “aneh” ini 😊

```
class Count {  
    public static void main (String args[]) {  
        int i = 0;  
        while(i < 50) {  
            System.out.println(i);  
            i=i+1;  
        }  
    }  
}
```

Kelas sederhana

- ★ contoh: tipe mahasiswa

```
class Mahasiswa {  
    String nama;  
    String alamat;  
    String nim;  
    String jurusan;  
}
```

- ★ class: definisi tipe baru

- ★ variabel (instans) untuk tipe ini disebut object

Contoh kasus

- ★ mendefinisikan object:

```
Mahasiswa mhs1;  
mhs1 = new Mahasiswa();
```

- ★ Deklarasi object (variabel) tipe reference tidak membuat object. Object dibuat dengan perintah `new ClassName()`

- ★ contoh:

```
public class MahasiswaTest {  
    public static void main(String args[]) {  
        Mahasiswa mhs = new Mahasiswa();  
        mhs.nama = "Amir";  
        mhs.alamat = "Dago";  
        mhs.jurusan = "Informatika";  
        System.out.println("nama: "+mhs.nama);  
        System.out.println("alamat: "+mhs.alamat);  
    }  
}
```

Achmad Imam Kistijantoro- Diktat Java Programming

Class method

★ class method: operasi yang disediakan oleh suatu class

★ contoh:

```
class Mahasiswa {  
    String nama;  
    String alamat;  
    String nim;  
    String jurusan;  
  
    String getNama() {  
        return nama;  
    }  
  
    void setNama(String nm) {  
        nama = nm;  
    }  
}
```

Class method

```
public class MahasiswaTest {  
    public static void main(String args[]) {  
        Mahasiswa mhs = new Mahasiswa();  
        mhs.setNama("Amir");  
        System.out.println("nama: "+mhs.getNama());  
    }  
}
```

★ Dibandingkan cara sebelumnya, manakah yg lebih baik ?

Kelas (class)

- ★ definisi internal data, internal method, atribut & behaviour dari object
- ★ Enkapsulasi: membungkus data internal dengan menyediakan interface untuk akses data internal
- ★ sintaks pendefinisian kelas:

```
public class myClass {  
    public myClass() { // konstruktor  
    }  
    // definisi method & atribut  
    private int myVariable;  
    public void methodA() {  
    }  
}
```


Konstruktor

- ★ method khusus yg digunakan untuk membuat object
- ★ memiliki nama sama dengan nama class
- ★ contoh:

```
class Mahasiswa {  
    String nama;  
    String alamat;  
    String nim;  
    String jurusan;  
    public Mahasiswa() {  
    }  
    public Mahasiswa(String nm) {  
        nama = nm;  
    }  
}
```

Konstruktor

```
public class MahasiswaTest {  
    public static void main(String args[]) {  
        Mahasiswa mhs = new Mahasiswa("Amir");  
        System.out.println("nama: "+mhs.getNama());  
    }  
}
```

Konstruktor

- Di Java hanya ada konstruktor
 - Tidak ada Copy Constructor
 - Tidak ada destruktur
 - Otomatis ada garbage collection (pemberesan memori secara otomatis)
 - Ada finalizer `finalize()`
- Tidak ada operator `=`, tapi ada method `clone()`
 - Baca dokumentasi clone di API Java

Deklarasi Konstruktor

- Nama konstruktor sama dengan nama kelas dan tidak memiliki nilai kembalian (sama seperti C++)
- Boleh ada banyak konstruktor (sama seperti C++)
- Contoh konstruktor

```
Point(int x, int y) {  
    this.x = x;  
    this.y = y;  
}
```

Mekanisme Enkapsulasi

★ Java menyediakan mekanisme pendefinisian scope member variable/method dari sebuah class dengan mekanisme:

- private
- protected
- public

★ **private:**

- variable/method hanya dapat diakses oleh kelas itu sendiri

★ **protected:**

- variable/method dapat diakses oleh semua kelas turunan
- variabel tidak dapat diakses dalam pola use, bukan sebagai inheritance (lihat contoh Point pada bagian package (slide no. 23))

★ **public:**

- variable/method dapat diakses oleh semua kelas

method & attribute access modifier

- ★ access modifier: menentukan apakah method/atribut tersebut dapat dipanggil oleh kelas lain
 - private: hanya dapat diakses dari object itu sendiri
 - protected: hanya dapat diakses object lain yang diturunkan dari kelas object tersebut (akan diterangkan kemudian)
 - public: dapat diakses oleh object lain
 - default: hanya dapat diakses oleh object yang berada dalam satu kelas

Contoh public

```
// contoh public variable  
// file name: Point.java
```

```
class Point {  
    public int x, y;  
}
```

```
//file name: Test.java
```

```
public class Test {  
    public static void main(String[] args) {  
        Point p = new Point();  
        System.out.println(p.x + " " + p.y);  
    }  
}
```

Contoh private:

```
// file name: Point.java
public class Point {
    private int x, y;
    public int getX() { return x; }
    public int getY() { return y; }
}
```

```
-----



// file name: Test.java
class Test {
    public static void main(String[] args) {
        Point p = new Point();
        System.out.println(p.x + " " + p.y); // error !, karena
        protected
        System.out.println(p.getX()+ " "+ p.getY() );
    }
}
```


reference this

- ★ Pada Java, this adalah reference yg mengacu ke object itu sendiri. Contoh:

```
// file name: Point.java
public class Point {
    private int x, y;
    public int getX() { return this.x; }
    public int getY() { return this.y; }
}
```

- ★ sama dengan:

```
// file name: Point.java
public class Point {
    private int x, y;
    public int getX() { return  x; }
    public int getY() { return  y; }
}
```

static dan alokasi memori

- ★ atribut dan method hanya dapat diakses jika object telah dibuat

- contoh:

```
Mahasiswa mhs;  
System.out.println(mhs.getNama()); // error  
mhs = new Mahasiswa("Amir");  
System.out.println(mhs.getNama());
```

- ★ atribut & method static dapat diakses tanpa melalui object
- ★ static: alokasi statis, hanya ada satu instans dalam seluruh program

Contoh:

```
class Mahasiswa {
    static int jumlah;
    String nama;
    String alamat;
    String nim;
    String jurusan;
    public Mahasiswa() {
    }
    public Mahasiswa(String nm) {
        nama = nm;
        jumlah = jumlah+1;
    }

    public static int getJumlah() {
        return jumlah;
    }
}
```

Contoh static:

```
public class StaticTest {  
    public static void main(String args[]) {  
        Mahasiswa mhs1 = new Mahasiswa();  
        System.out.println("Jumlah:"+Mahasiswa.getJumlah());  
        Mahasiswa mhs2 = new Mahasiswa();  
        System.out.println("Jumlah:"+Mahasiswa.getJumlah());  
    }  
}
```

Kelas Sederhana [1]

- Kelas paling sederhana merepresentasi type (atribut) dan prototype (method)
- Paket ADT dalam bahasa C dapat ditranslasi menjadi sebuah kelas dengan konvensi sbb
 - File adt.h tidak perlu ditulis
 - File adt.c langsung menjadi sebuah kelas
 - File madt.c menjadi sebuah kelas terpisah
 - Setiap atribut dan method harus diberi “access modifier” secara eksplisit

Contoh kelas ADT

- Lihat contoh program kecil
 - Point
 - Stack

Penempatan Access Modifier

- € akses modifier dilakukan dengan menempatkannya langsung di depan nama **setiap** property atau method
 - di C++ ada tanda ':' setelah access modifier dan akses method berlaku untuk semua method/property sampai akses modifier berikutnya

```
class Point3D {  
    private int x, y;  
    public int z;  
}
```

Kelas Point

- Perhatikan kelas berikut

```
class Point {  
    private int x, y;  
    public int getX() { return x; }  
    public int getY() { return y; }  
};
```

- Kelas di atas dapat dicompile sebagai C++ ataupun Java

Instansiasi Objek

€ Deklarasi

```
Point p;
```

Perhatikan, ini hanya deklarasi

€ Alokasi

```
p = new Point(); //konstruktor dipanggil
```

€ Akses field dan method dengan titik (bukan ->)

```
p.x = 5;
```

```
int x = p.getX();
```

Method di Java

- Hanya ada sedikit perbedaan antara method di C++ dengan Java
 - Java tidak memiliki parameter default
 - method tidak bisa ditandai const
- Tidak ada pointer/reference untuk tipe dasar
 - Konsekuensi paling sederhana: tidak bisa membuat method untuk menukar dua tipe dasar

Passing parameter

- Passing parameter Selalu by value
- Tipe primitif
 - Dibuat salinannya
 - Variabel parameter boleh diubah/dipakai, tapi tidak mengubah nilai/variabel pemanggil
- Tipe reference (objek)
 - Pass by reference, method yang dipanggil mempengaruhi objek pada parameter
 - Jika diassign nilai baru, maka tidak akan terlihat efeknya oleh pemanggil

Bandingkan kedua potongan kode

```
class A {  
    int v = 0;  
    void incA(){v++;}  
    void hello(A  
        objekA, int z) {  
        objekA.incA();  
        z = 9; }  
}  
A x = new A();  
int m = 2;  
hello(x, m);  
//x.v = 1, m tetap 2
```

```
class A {  
    int v = 0;  
    void incA(){v++;}  
    void hello(A  
        objekA, int z){  
        objekA = new A();  
        objekA.incA();  
        z = 9;}  
}  
A x = new A();  
int m = 2;  
hello(x, m);  
//x.v = 0, m tetap 2
```