Bahasa Pemrograman Java

Yohanes Nugroho
rev: Achmad Imam Kistijantoro

Disclaimer

- Materi ini diberikan untuk mahasiswa IF2032 yang telah mendapatkan pelajaran Objek dan C++,
- Materi tidak dimaksudkan untuk digunakan untuk penggunaan lain selain untuk kuliah IF2032 (di mana mahasiswa sudah belajar konsep dan implementasi dalam C++)

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 (mesin cuci, ponsel, dll)
  - Kemudian Web/WWW menjadi populer yang mempopulerkan Java dan Applet

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

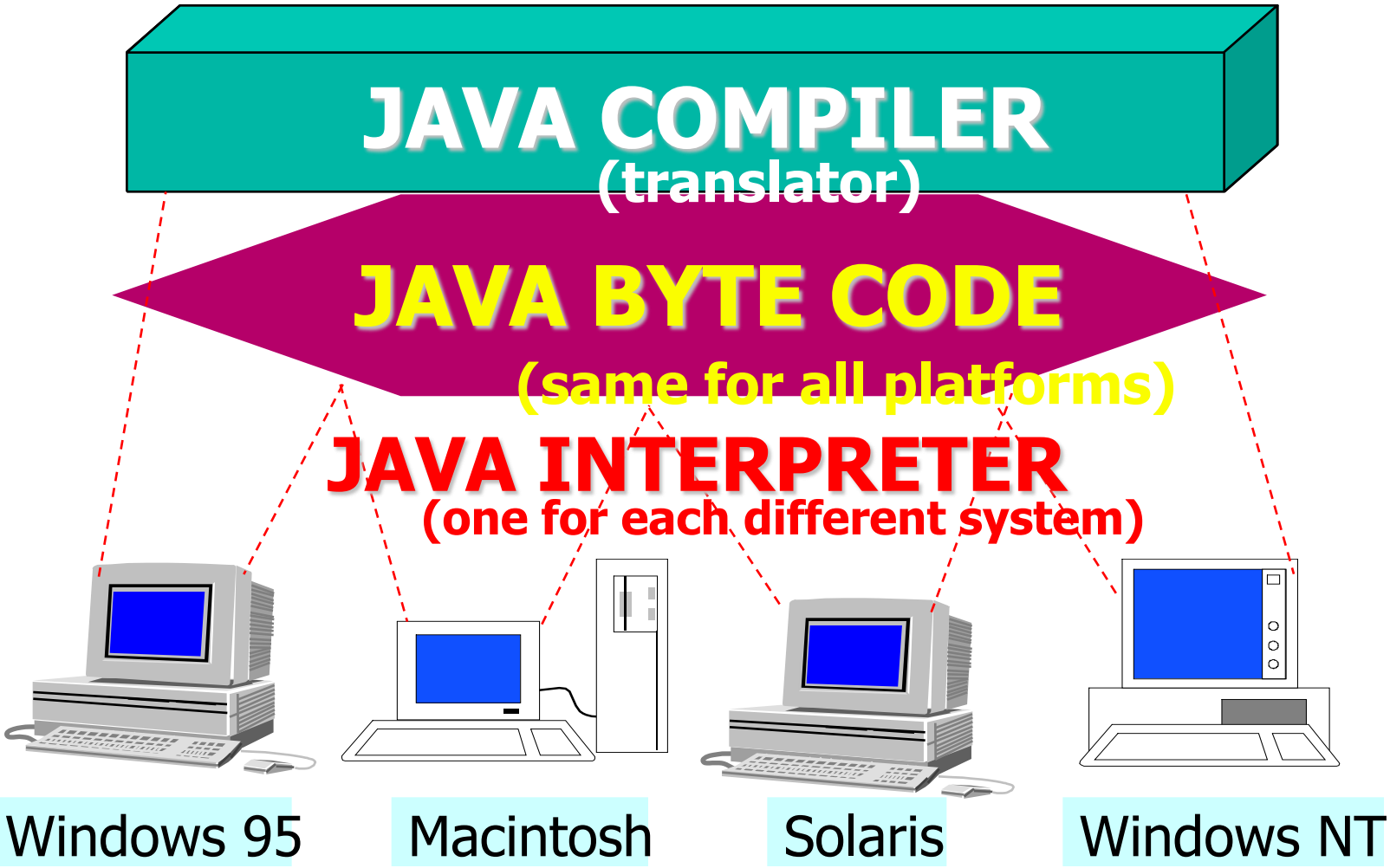
Java Virtual Machine (JVM)

- 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*"

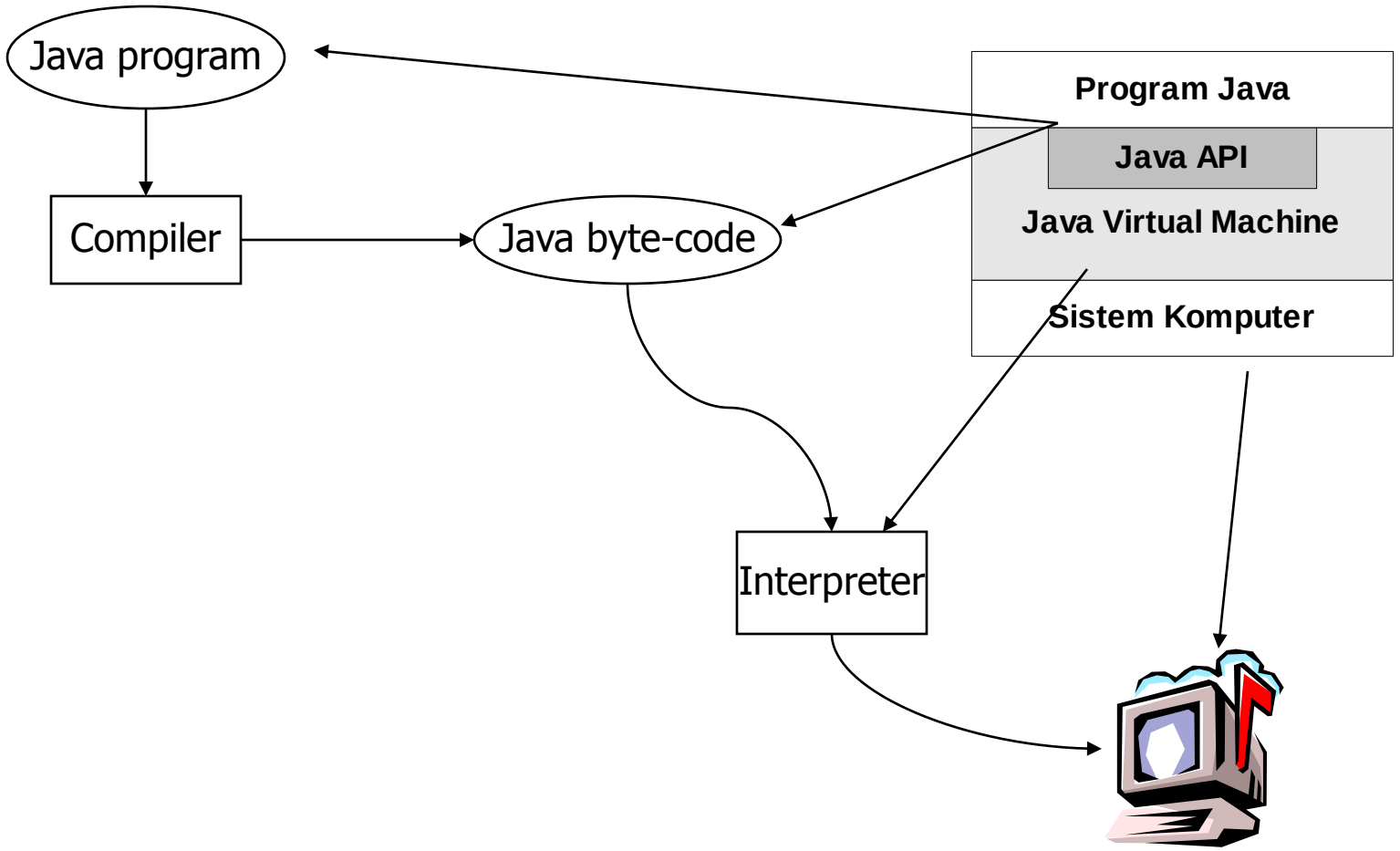
Application Programming Interface

- 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



Yang akan dibahas dalam Paruh kedua kuliah IF2032

- Meliputi:
  - Bahasa Pemrograman Java
    - Pembahasan didasarkan pada C++
  - 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

Mengenal Lingkungan Pemrograman Java

Overview Hello World

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

Hello World dalam Java

- Contoh program Hello World

```
/*program hello world*/
class HelloWorld {
    public static void main(String argv[]){
        System.out.println("Hello World");
    }
}
```

- Nama file harus sama dengan nama kelas
- Sintaks komentar sama dengan C++ (memakai // atau /* */)

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

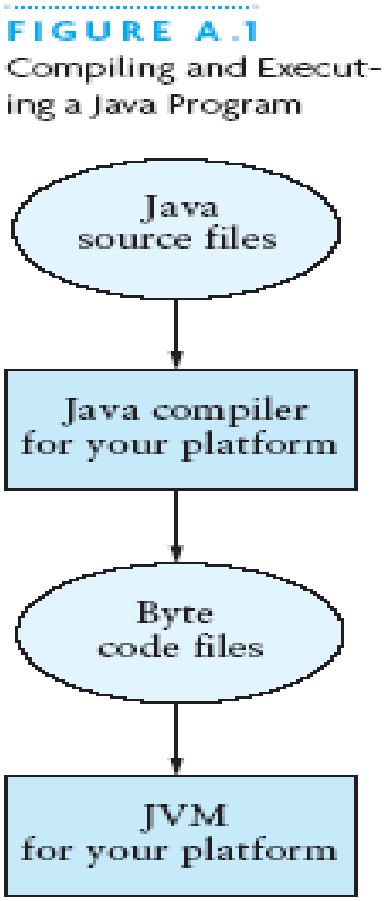


FIGURE A.1
Compiling and Executing a Java Program

Penjelasan Hello World

- Semua program Java merupakan kumpulan kelas
  - Tidak boleh lagi ada fungsi di luar kelas, seperti di C++
  - 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/C++)
  - Memiliki parameter (`String arg[]`) yang merupakan parameter dari user
    - Di C/C++: `int main(int argc, char *argv[]);`
    - Array di Java punya informasi panjang: arg.length seperti argc di C/C++
    - Elemen dari arg sama seperti char *argv[ ] di C/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 Operator terhadap tipe
- Tipe String
- Kondisional
- Loop
- Reference

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

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`	$+/-10^{-38}$ to $+/-10^{+38}$ and 0, about 6 digits precision
`double`	$+/-10^{-308}$ to $+/-10^{+308}$ and 0, about 15 digits precision
`char`	Unicode characters (generally 16 bits per char)
`boolean`	True or false

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*)
  - Asignment: `>>>=`
- 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
- Asignment: `=, += -= *= /= <<= >>= &= |=`

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

String

- String merupakan kelas khusus di Java
  - Java memperlakukan String tidak seperti objek lain
  - Bukan tipe dasar, tapi bagian dari bahasa Java
- String dibahas karena akan banyak dipakai (misalnya untuk menuliskan output)

```
String s = "Hello World";
```

- String merupakan sebuah kelas, dan di dalamnya ada beberapa method, misalnya:
  - `s.length()` --> panjang string
  - `s.substr(0, 2)` --> mengembalikan "He" (karakter ke 0 sampai ke-2 [*bukan* sampai dengan])

Operator String

- Selain memiliki method, string juga memiliki operator "+"
- Operator + akan mengkonversi float/integer secara otomatis jika digabung dengan string

```
/*konversi otomatis 2 ke string*/
String s = "Banyaknya "+ 2;
int z = 4;
String s = "Ada " + z + " buah ";
String s = 5; /*tidak boleh */
```

Literal String

- Literal string merupakan instans dari objek string
- Method boleh dipanggil langsung dari Literal:
  - `"Hello".length()` menghasilkan 5

Sequence Escape String

- Serangkaian karakter diawali \ (backslash) untuk mengetikkan karakter khusus, escape berikut sama dengan C/C++
  - `\n` : newline
  - `\t` : karakter tab
  - `\\` : backslash
  - `\”` : double quote
  - `\'` : apostrophe
- Escape khusus Java: `\udddd`: karakter dalam unicode dddd adalah digit heksadesimal (0-9, A-F)

Sifat Immutable String

- String sebenarnya *immutable* (tidak bisa diubah)
- Dalam instruksi sbb:

```
String a = "hello";
a = a + " world";
```

- Sebuah objek baru diciptakan, objek lama dibuang (untuk dipungut oleh garbage collector). a menunjuk ke objek yang baru

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 ==

Perbandingan String

- Untuk membandingkan string tanpa membedakan case, gunakan equalsIgnoreCase
- Untuk membandingkan urutan String menurut kamus (*lexicographicaly*), gunakan method compareTo,
- Contoh:

```
str1.compareTo(str2)
```

  - Nilai kembalian:
    - 0 jika string sama
    - Suatu nilai negatif jika str1 < str2
    - Suatu nilai positif jika str1 > str2

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

OOP di Java

Kelas, Objek, Penurunan

OOP Dengan Java

- Kelas
- Instansiasi Objek
- Penurunan

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)

Kelas Point

- Perhatikan kelas berikut

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

- Kelas di atas dapat dicompile sebagai C++ ataupun Java
  - Namun sebenarnya tidak sama (access modifiernya berbeda)

Instansiasi Objek

- Deklarasi

  ```
  Point p;
  ```

  Perhatikan, ini sama dengan: `Point *p` di C++
  (konstruktor belum dipanggil)

- Alokasi

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

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

  ```
  p.x = 5;
  int x = p.getX();
  ```

Access Modifer

- Default Access Modifier di Java adalah "default" (di C++ adalah private)
  - Ini akan dijelaskan lebih lanjut pada konsep "Package"
  - Secara sederhana: kelas pada direktori yang sama bisa mengakses field default
- Java juga mengenal akses modifier private, public, dan protected, dengan sifat sama seperti di C++

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;
}
```

Garbage Collector

- Java memiliki garbage collector
  - Thread yang otomatis mendealokasi memori yang tidak diperlukan
- Meskipun ada "new", tapi tidak ada "delete"
- Java secara otomatis melacak penggunaan variabel dan objek
  - Memori otomatis dibereskan jika sudah tidak dipakai
  - Pemberesan dilakukan jika sudah dirasa perlu oleh JVM (ketika load sedang rendah dan memori mulai penuh)
  - Pemberesan paksa bisa dilakukan

Menjalankan Paksa Garbage Collector

- Dalam keadaan tertentu sampah perlu dibereskan dengan segera (tidak menunggu sistem yang melakukannya)
  - Operasi akan lebih cepat jika memori dalam keadaan "kosong", jadi kadang memori perlu dibereskan sebelum operasi kompleks dilakukan
- Dilakukan dengan instruksi:

```
System.gc();
```

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
```

Konstruktor

- Di Java hanya ada konstruktor
  - Tidak ada Copy Constructor
  - Tidak ada destruktor
    - Otomatis ada garbage collection (pemberesan memori secara otomatis)
    - Ada finalizer
- 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;
}
```

Copy Constructor

- Tidak ada copy constructor yang otomatis dijalankan di Java
- Untuk membuat sesuatu yang mirip dengan copy constructor C++:
  - buat konstruktor yang parameternya adalah objek yang merupakan instans kelas tersebut

```
class A { A(A x) { } }
```

  - Isi method adalah menyalin x ke objek saat ini

Perbedaan dengan C++

- Copy constructor yang dibuat di Java harus dipanggil manual:

```
A x = new A();
A y = new A(x); /* memanggil copy
constructor x disalin menjadi y*/
```

- Copy constructor di C++ otomatis dijalankan pada:
  - Saat deklarasi sambil inisialisasi
  - Pemanggilan fungsi tanpa reference

Operator assignment

- Di Java, tidak ada keyword "operator" seperti di C++
  - Tidak bisa membuat operator =
- Untuk membuat salinan objek (deep clone), override method clone() di kelas objek:

```
Object clone() { }
```

- Buat salinan objek, lalu return-kan salinan tersebut
- Jika ada copy constructor, clone bisa ditulis:

```
Object clone() { return new A(this);
  }
```

Inheritance

- Di Java hanya ada single inheritance
- Penurunan selalu bersifat "public" (tidak ada penurunan private dan protected seperti di C++)
- Kata kunci yang dipakai adalah **extends**

```
class Line3D extends Line2D {


}
```

Kelas Abstrak

- Di C++ boleh ada method virtual murni dalam sebuah kelas, di Java juga boleh ada method semacam itu
  - method virtual murni: method yang belum ada isinya
- Method virtual murni dalam Java ditandai dengan kata kunci `abstract`
- Kelas yang mengandung method virtual murni harus dideklarasikan dengan kata kunci `abstract`

Contoh kelas Abstrak dan Turunannya

```
abstract class Bangun {
   void test() {
     System.out.println("Luas"+getLu
     as()); }
   abstract int getLuas();
}
class Lingkaran extends Bangun {
   int getLuas() { return pi*r*r;}
}
```

Keyword super

- Memanggil konstruktor superclass
  - Dengan parameter yang sesuai tentunya
  - Parameter boleh kosong, seperti ini: `super()`
- Harus merupakan statement pertama dalam konstruktor anak
  - Harus mengaktifkan `super` sebelum melakukan operasi yang lain

Memanggil konstruktor parent

```
class Ortu {
   Ortu(String namaKeluarga) {
     family=namaKeluarga; }
}
class Anak  {
   Anak(String nama, String
     namaKeluarga) {
      super(namaKeluarga);
   }
}
```

Pemanggilan Method pada Parent

- Jika suatu method meng-override method parent dan ingin memanggil implementasi parent, gunakan syntax:

```
super.namamethod(parameter)
```

- Sifat super pada konstruktor
  - konstruktor default (nullary constructor) parent akan selalu dipanggil jika super() tidak dipanggil

Interface

- Java memiliki konsep interface yang tidak dimiliki C++
  - Konsep ini memungkinkan sebagian fitur multiple inheritance diimplementasikan
- Interface adalah kelas yang semua methodnya belum didefinisikan
  - Semua method kosong

Contoh interface

```
interface Draw {
   void draw();
   void draw3D();
}
```

- Interface tidak punya konstruktor, destruktor (finalizer), dan apapun, hanya punya member variabel dan deklarasi method

Implementasi interface

- Isi interface diimplementasikan oleh kelas dengan keyword **implements**
- Sebuah kelas boleh mengimplementasikan banyak interface
- Contoh:

```
class Lingkaran implements Draw {
   void draw() { /*implementasi draw*/ }
   void draw3D() { /*implementasi
     draw3D*/ }
}
```

Implementasi banyak interface

```
interface Color {
   void setColor(int color);
   int getColor();
}
class Lingkaran implements Draw, Color
  {
   void draw() { /*implementasi draw*/
     }
   void draw3D() { /*implementasi
     draw3D*/ }
   void setColor(int color);
   int getColor();
}
```

Beda kelas abstrak dengan interface

- Kelas abstrak boleh memiliki method yang sudah diimplementasikan
  - Interface harus "kosong" (tidak ada method yang terdefinisi pada interface)
- Kelas hanya boleh meng-extend (diturunkan dari) satu kelas
  - Kelas boleh mengimplementasikan banyak interface

Kapan memakai kelas abstrak dan interface

- Kelas abstrak
  - Jika sudah ada algoritma yang bisa diimplementasikan di kelas tersebut
- Interface
  - Hanya memberi kontrak, misalnya Interface Measureable untuk menyatakan objek yang bisa diukur keliling dan luasnya

Inner Class

- Kelas di dalam kelas:

```
class A {
   class B {

   }
}
```

- Kelas B hanya boleh dipakai di A
  - Tidak bisa: `A.B x = new A.B()`
  - Salah satu tujuan kelas internal agar B tidak terlihat dari luar

static inner class

- Kelas internal bisa didefinisikan agar dapat diakses dari luar

```
class A {
   static class B { }
}
A.B = new A.B(); /*bisa/boleh*/
```

Package

Konsep Package dalam Java

Package

- Sekumpulan kelas Java bisa dikelompokkan dalam package
  - Seperti namespace di C++
  - Umumnya pengelompokkan dilakukan berdasarkan fungsionalitas
- Kelas yang berada di direktori yang sama otomatis berada dalam package yang sama
  - Package berhubungan langsung dengan filesystem

Contoh Penggunaan Package

- Kita ingin mengelompokkan kelas yang berhubungan dengan image, misalnya:
  - `ImageGIF, ImageJPEG`
  - kedua file kelas tersebut ada di direktori `c:\source\image`
- nama package diberikan di file ImageGIF.java (dan ImageJPEG.java) dengan keyword package, seperti ini:

```
package image;
class ImageGIF {
}
```

Mengkompilasi kelas dalam Package

- Dari direktori di c:\source

```
javac image\ImageGIF.java
javac image\ImageJPEG.java
```

- Lalu bagaimana memakai kelas dalam package tertentu?
  - Kita perlu menyebutkan dengan lengkap nama package dan kelas atau
  - Kita perlu mengimpor package atau kelas tersebut

Memakai ImageGIF

- Gunakan nama lengkap (nama package + nama Kelas) dengan titik untuk memakai kelas dalam Package
  - Cara ini merepotkan (harus selalu mengetik nama lengkap)
- Misalkan File `TestImage.java` yang memakai ImageGIF ada di `c:\source`

```
class TestImage {
   void hello() {
      image.ImageGIF a = new
       image.ImageGIF();
   }
}
```

Memakai ImageGIF dengan Import

- Instruksi import digunakan agar nama kelas pada suatu package dikenali tanpa nama lengkapnya (cukup nama kelasnya), contoh:

```
import image.ImageGIF;
class TestImage {
   void hello() {
      ImageGIF a = new ImageGIF();
   }
}
```

Instruksi import

- import namapackage.NamaKelas;
  - harus satu per satu nama kelas disebutkan
- atau import namapackage.*;
  - semua Kelas dalam package tersebut diimport
  - lebih singkat menuliskannya
  - kompilasi lebih lama (semua nama kelas dicek)

Mengkompilasi TestImage

- Pindah ke drive C
  - ketik `c:`
- Pindah ke direktori source
  - ketik: `cd \source`
- Kompilasi seperti biasa
  - ketik: `javac TestImage.java`

Hierarki Package

- Package bisa bertingkat
  - misalnya kita ingin membuat package `SMSserver`
  - di dalam package smsserver ada package gsm dan cdma, masing-masing memiliki kelas `SMS`
  - boleh ada 2 kelas bernama sama di package berbeda
- cara membuat direktori:

```
mkdir c:\SMSserver
mkdir c:\SMSserver\gsm
mkdir c:\SMSserver\cdma
```

Menempatkan file

- File `SMS.java` untuk package cdma diletakkan di direktori `c:\SMSserver\cdma`
  - header file `SMS.java` berisi:

```
package SMSserver.cdma;
```

- File `SMS.java` untuk package gsm diletakkan di direktori `c:\SMSserver\gsm`
  - header file SMS.java berisi:

```
package SMSserver.gsm;
```

Mengkompilasi Isi Package

- Kompilasi dilakukan seperti biasa (dari c:\)

```
javac c:\smsserver\cdma\*.java
javac c:\smsserver\gsm\*.java
```

Mengimpor package dalam hierarki

- Import dengan nama package (yang hierarkinya nama packagenya dipisah dengan titik)

```
import smsserver.gsm.SMS;
```

- Hirerarki bisa bertingkat sebanyak mungkin
- Jika dalam package smsserver ada file `MainServer.java`, maka
  - import smsserver.*;
  - hanya akan mengimpor semua kelas dalam package smsserver tapi tidak mengimpor kelas dalam subpackage gsm dan cdma

Penamaan Package

- Semua karakternya memakai huruf kecil
- Sesuai penamaan domain, tapi terbalik, misalnya package XML milik lab programming ITB:

```
id.ac.itb.informatika.programming.
  xml
```

- nama id.ac.itb.if.programming tidak bisa dipakai, karena if adalah keyword di Java

Pemaketan package dalam file JAR

- Selain diletakkan dalam direktori, file kelas bisa dimasukkan dalam file JAR
- Contoh pembuatan file JAR:

```
jar -cf smsserver.jar c:\
```

- File JAR harus dimasukkan ke classpath agar dapat dipakai

Class path

- Classpath adalah lokasi (*path*) di mana Java akan mencari file class
- Default classpath java adalah . (titik) yang berarti direktori saat ini, dan file JAR milik sistem (bawaan Java)
- Classpath bisa diubah

```
export CLASSPATH=/usr/test.jar:. (Linux)
set CLASSPATH=c:\test.jar;. (Windows)
```

Contoh Pemakaian Classpath

- Masukkan seluruh direktori smsserver atau direktori image dalam contoh sebelumnya ke c:\library
- set classpath menjadi c:\library dan direktori saat ini:

```
set CLASSPATH=c:\library;.
```

- Setelah classpath diset, maka file yang memakai kelas dalam package boleh berada di mana saja

Contoh: isi kelas dalam JAR

- Masukkan file jar ke c:\library
- set classpath menjadi:

```
set
  CLASSPATH=c:\library\smsserve
  r.jar;c:\library\image.jar;.
```

- File yang memakai kelas dalam package boleh berada di mana saja

Menjalankan Program dalam Package

- Set classpath, lalu:

```
java
  namapackage.subpackage.KelasX
```

- Atau set classpath untuk saat ini saja:

```
java -cp test.jar;.
  namapackage.subpackage.KelasX
```

Array

Objek Array

Array adalah objek

- Di Java, array adalah Objek
- Contoh array:

```
char ac[] = { 'n', 'o', 't', ' ',
  'a', ' ', 'S', 't', 'r', 'i', 'n',
  'g' };
```

- member yang ada: length dan semua method yang ada di kelas Objek
- dalam array di atas:

```
ac.length == 12
```

Array dinamis

- Array diciptakan dengan `new`
- Contoh array of Integer:

```
int a[] = new int[5];
```

- Array tidak bisa diresize
  - untuk mengubah ukuran array, buat array baru, salin isi array lama ke yang baru

Array of Objects

- `new` pada array hanya mengalokasikan array, kode:

```
Lingkaran ling = new Lingkaran[5];
```

  - Hanya menciptakan array of lingkaran (lingkarannya sendiri belum diciptakan, perhatikan bahwa ini berbeda dengan C++)

- Untuk membuat 5 Objek Lingkaran dalam array `ling`:

```
for(int i = 0; i<5; i++)
   ling[i] = new Lingkaran();
```

Array Multidimensi

- Array multidimensi diakses dengan operator [] sebanyak dimensinya
- Contoh:

```
int matriks[][] = new int[5][6];
```

- Untuk array yang ukuran tiap barisnya tidak sama:

```
int matrix[][] = new int[5][];
matrix[0] = new int[5];
matrix[1] = new int[3];
```

Exception

Exception di Java

Exception

- Sebagai bahasa yang mendukung OOP, Java mendukung Exception
  - Sintaks dan sifatnya mirip sekali dengan C++
- sintaks dasar:

```
try {

} catch (Exception e) {

}
```

Perbedaan dengan C++

- Dalam klausa catch tidak perlu & (reference), karena semua di Java adalah reference
- Kelas Exception harus diturunkan dari kelas Throwable
  - Di C++ kelas apa saja bisa menjadi kelas Exception
  - Kelas eksepsi yang paling standar di Java adalah: Exception

Klausa finally

- Klausa finally: dieksekusi apapun yang terjadi (baik eksepsi dilempar atau tidak)

```
try {

} catch (Exception e) {

} finally {
   //kode
}
```

Contoh Exception dengan Finally

- Dalam potongan kode berikut:

```
try {
   System.out.println("Hello");
   //kode lain
} catch (Exception e) {
   System.out.println("World");
} finally {
      System.out.println("Selesai");
}
```

- Apapun yang terjadi "Selesai" akan diprint

Perhatikan bahwa

- Pada potongan kode berikut:

```
public class Test {
      int testMethod() {
         try {
            return 5;
         } catch (Exception e) {
         } finally {
            return 10;
         }
      }
}
```

- testMethod akan mengembalikan 10 (dan bukan 5)

OOP Spesifik Java

Konsep OOP Spesifik Java

Kelas Akar

- Kelas akar (root class) adalah kelas yang menjadi nenek moyang (ancestor) semua objek
- Di Java, kelas akar adalah Object (`java.lang.Object`)
- Kelas yang tidak diturunkan dari apapun berarti diturunkan dari kelas yang bernama Object
  - Implikasinya: semua kelas adalah turunan dari Object
- Object memiliki beberapa method dasar seperti: `toString()`, `clone()`, dan method untuk sinkronisasi

Method String toString()

- Override method `toString()` untuk mencetak objek dengan lebih baik
- Misal, untuk kelas `Point`, isi methodnya:

```
String toString() {
   return "[" + x + ", " + y + "]";
}
```

- Dengan method di atas, Point bisa dicetak dengan mudah:

```
Point p = new Point(1,2);
System.out.println(p); /*mencetak point*/
```

- output potongan kode di atas: `[1, 2]`

final

- Sesuatu yang final tidak bisa diubah
  - Pada member variable berarti konstanta
  - Pada member function berarti tidak bisa diubah di turunannya
- Final member

```
final int pi=3.14;
```

- Final function

```
final void hello() {
}
```

  - Error jika di-redefinisi di turunannya

finalize

- Dalam kondisi tertentu ada resource yang tidak bisa direlease oleh Java
  - misal: kode yang memanggil kode native
  - resource ini harus direlease secara manual
- Kita bisa membuat Method `finalize` yang akan dieksekusi sebelum garbage collector menghancurkan Objek tersebut
  - biasanya yang dilakukan adalah membebaskan resource