Pemrograman Berorientasi Objek dengan Java

Oleh : Yohanes Nugroho

Review Oleh: Hananto W

Overview

- Overview bahasa Java
- Hello World
- Konstruksi dasar bahasa Java
- OOP di 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

Yang akan dibahas dalam kuliah

- 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*/
public 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 /* */
- Komentar satu baris diawali dengan //

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

Penjelasan Hello World

- Semua program Java merupakan kumpulan kelas
  - Program "hello world" juga merupakan sebuah kelas
- Pada 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 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
  - int (32 bit)
  - long (64 bit)
  - byte (signed 8 bit)
  - char (16 bit UNICODE, tidak seperti 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:
  - 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

- 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
- Operator + pada string (dibahas dalam bagian mengenai String)

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 menjadi
  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 \ (bacslash) untuk mengetikkan karakter khusus, escape berikut sama dengan 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

```
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 String)

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

- Jangan membandingkan string dengan ==

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
- Di Java integer tidak sama dengan boolean
  - Di C integer dianggap sama dengan boolean
- Perhatikan bahwa hal berikut tidak boleh

```
int a = 1;
if (a) return; //integer tidak
  dikonversi ke boolean
```

- Di Java seharusnya:

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

Loop

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

OOP dengan Java

Implementasi Kelas, Method, dll

Property dan Method

- Di OOP:
  - Data menjadi property
  - Prosedur untuk data menjadi method
  - Data + Prosedur menjadi kelas

Deklarasi Kelas

- Deklarasi kelas memiliki sintaks seperti berikut:
- class NamaKelas {
  - /*0 atau lebih property*/
  - /*0 atau lebih method*/
- }

Property

- Deklarasi/definisi property:
  - Tipedata namaproperty;
- Tipe data adalah tipe data primitif atau kelas
  - Int x;
  - Point p;

Method

- Deklarasi method sama seperti prosedur di C:

```
public class ClassXXX{
    Tipekembalian method(tipe parameter) {
        /*isi method*/
        Return hasil; /*jika kembalian tidak void*/
    }
}
```

Passing Parameter

- Parameter untuk sebuah method boleh tipe dasar atau kelas (tepatnya reference ke objek)

```
public class Point{
   int x,y;
   void geser(int dx,int dy) {
     x+=dx;y+=dy;}/*boleh tidak pakai this*/
   void drawLine(Point p1, Point p2){/**/}
}
```

- Tidak ada pointer, tidak bisa membuat method yang mengubah tipe dasar yang dijadikan parameter
  - Tidak bisa membuat method untuk menukar dua int atau tipe dasar lain

Passing Reference(1)

- Karena reference yang dipassing, method seperti ini bisa berjalan sebagaimana yang dimaksudkan (p berubah)

```
public class Point{
    /*mencerminkan terhadap sumbu X*/
    /*contoh method yg salah dari konsep OO tapi
    jalan saat dieksekusi*/
    void mirrorSumbuX(Point p){
       p.setX(-p.getX());
    }
    /*pemanggilan*/
    public static void main(String args[]){
      Point p = new Point(10,20);
      p.mirrorSumbuX(p); /*p menjadi -10,20*/
    }
}
```

Passing Reference(2)

- Penjelasan kesalahan dari method mirrorSumbuX class Point
  - Dalam konsep OO parameter method hanya sebagai parameter input, bukan paramater output.
  - Method suatu objek hanya bertanggung jawab mengubah state dari objek itu sendiri, tidak boleh mengubah state dari objek lain.

Assign parameter(1)

- Jangan mengassign parameter dengan reference lain (efeknya: parameter tidak berubah)

```
public class Point{
   /*mencerminkan terhadap sumbu X*/
   /*contoh method yg salah dari konsep OO tapi
   jalan dari segi eksekusi*/
   void mirrorSumbuX(Point p){
      p = new Point(-p.getX(), p.getY);
   }
   /*pemanggilan*/
   public static void main(String args[]){
       Point p = new Point(10,20);
       p.mirrorSumbuX(p); /*p tetap 10,20*/
   }
}
```

Assign Parameter(2)

- Penjelasan kesalahan dari method mirrorSumbuX class Point
  - Dalam konsep OO parameter method hanya sebagai parameter input, bukan paramater output.
  - Kalau di dalam suatu method diinstansiasi sebuah objek, harus di passing sebagai return value dari method, bukan di passing sebagai parameter output dari method.

Overloading

- Boleh ada lebih dari satu methode dengan nama yang sama
- Parameter method harus berbeda
  - Jika tipe parameter sama urutannya tidak boleh sama

```
public class Mahasiswa {
   boolean find(String nim, int nilai) {      return true;}
   boolean find(String nim) {/*boleh*/     return true; }
   boolean find(int nilai, String nim){ /*boleh*/
     return true;     }
   boolean find (String nama, int nilai) {
     /*tidak boleh, urutan tipe sama*/
     return true;     }
}
```

Konstruktor

- Method khusus yang dijalankan ketika suatu objek diinstansiasi
- Nama method sama dengan nama kelas, tapi tanpa tipe kembalian sama sekali (tanpa void)
- Boleh ada lebih dari satu konstruktor

```
public class Mahasiswa {
   String nim,nama;
   Mahasiswa() {}
   Mahasiswa(String nim,String nama)
{
         this.nim = nim;
this.nama=nama;
   }
}
```

Access Modifier

- Ada 4 acess modifier yang bisa diberikan di depan deklarasi method/property
- Private: hanya bisa diakses kelas ini
- Protected: bisa diakses kelas ini dan turunannya
- Public: bisa diakses semua kelas
- "default": bisa diakses kelas dalam package yang sama

final

- Keyword final bisa diletakkan setelah deklarasi apapun yang nilainya tidak akan diubah lagi (konstan):
  - final double PI=3.14;
- Jika diletakkan di depan sebuah method, method tersebut tidak bisa diubah diturunannya (akan dibahas lebih lanjut dalam konsep turunan)

Menyalin Objek

- Point b = new Point(5,5);
- Point a = b;
- Jika a diubah (misal a.setX(6)) maka b ikut berubah
- Point a bukan salinan b, hanya menunjuk ke b, jika ingin membuat salinan b harus menggunakan method clone

Deep vs Shallow clone

- Dalam kelas Point, menyalin nilai x dan y ke titik yang baru sudah cukup untuk membuat objek baru
  - Dapat dicopy dengan shallow clone (nilai property cukup disalin)
- Dalam kelas List, menyalin head tidak cukup untuk membuat List baru (jika hanya head yang disalin, elemen list tetap dipakai bersama)
  - Harus dicopy dengan deep clone

clone()

- Shallow clone:
  - Agar dapat diclone kelas (Point) harus mengimplementasikan interface Cloneable
  - Hanya penanda bahwa bisa diclone
  - class Point implements Cloneable { }
  - Sekarang bisa:
  - Point b = (Point)a.clone();
- Deep clone:
  - Harus didefinisikan sendiri

Contoh Shallow Clone

```
class Point
{
   int x,y;
   Point clone(){
        Point p = new Point(x,y);
        return p;
   }
}
```

Contoh Deep Clone

```
class List {
   List clone(){
    List p = new List(); /*buat list baru*/
    Object o = this.First();
    while (o!=NULL){
     p.addLast(o.clone()); /*tambahkan clone ke p*/
     o = o.getNext();
    }
    return p;
   }
```

Penurunan (Inheritance)

- Sebuah kelas bisa diturunkan dari kelas lain (property dan method bisa digunakan lagi)
  - Contoh: Point3D dari Point2D, LingkaranBerwarna dari Lingkaran
- Turunan bisa menambah method dan atau field
  - Menambah field z pada Point2D (menambah accessor juga)
  - Menambah field warna pada Lingkaran

Hubungan penurunan

- Jika C diturunkan dari B dan B diturunkan dari A:
  - B adalah superclass/parent C (C adalah subclass/child/anak B)
  - A adalah superclass/parent B (B adalah subclass/child/anak A)
  - A adalah ancestor C
- Java memakai keyword extends:
  - Class LingkaranBerwarna extend Lingkaran
  - Class Point3D extends Point2D

Overriding

- Jika ada method di kelas parent yang sudah didefinisikan, dan didefinisikan ulang, maka method pada kelas anak akan menutup method parent
  - Kecuali jika di parent dinyatakan final, method tidak bisa diganti

Keyword super

- Digunakan untuk Memanggil konstruktur/method superclass
  - Pemanggilan Dengan parameter yang sesuai tentunya:
    - Pemanggilan konstruktor: super(x, y);
    - Pemanggilan method: super.method(x, y);
  - Parameter boleh kosong, seperti ini: super()
- Super pada konstruktor harus merupakan statement pertama dalam konstruktor anak

Contoh pemakaian super(1)

```
class Lingkaran {
   protected int r;
   Lingkaran(int r){
      this.r = r;
   }
   void draw() {} /*menggambar
     lingkaran*/
}
```

Contoh pemakaian super(2)

```
class LingkaranBerwarna extend Lingkaran {
   int warna;
   LingkaranBerwarna(int r, int warna){
      super(r);
      this.warna = warna;
   }
   void draw() {
   /*menggambar lingkaran*/
      super.draw();
      /*instruksi untuk menambah warna*/
   }
}
```

Kelas Abstrak

- Kelas abstrak adalah kelas yang belum lengkap
- Ada method yang belum ada isinya
- Turunan kelas abstrak akan mengisikan method yang masih "kosong"
- Ada keyword abstract untuk menandai bahwa kelas adalah abstrak
  - Keyword juga dipakai untuk menandai method yang masih "kosong"

Contoh kelas Abstrak dan Turunannya

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

Interface

- Interface adalah semacam kelas, tapi bukan kelas ☺ ☺ ☺
- Interface adalah kelas yang semua methodnya belum didefinisikan, hanya spesifikasi
- Untuk memberikan "kontrak" kepada suatu kelas
- Interface dimaksudkan agar diimplementasikan ("*defined*", didefinisikan bodynya) oleh suatu kelas lain

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

Polymorphism

- Sebuah kelas "tahu" konteksnya berdasarkan reference yang ditunjuknya
- Contoh:
  - Lingkaran dan Segitiga diturunkan dari bangun, Bangun memiliki method luas yang mengembalikan integer yang masih abstrak (diimplementasikan oleh Lingkaran dan Segitiga)
  - Jika sebuah variabel (reference) bertipe bangun menunjuk ke Lingkaran, maka "sifatnya" akan seperti lingkaran, jika menunjuk ke segitiga, sifatnya seperti segitiga

Contoh

```
Lingkaran l = new Lingkaran(10);
Segitiga s = new Segitiga(1,2,3);
Bangun b = l;
/*mencetak luas lingkaran*/
System.out.println(b.getLuas());
/*mencetak luas segitiga*/
b = s;
System.out.println(b.getLuas());
```

Method/algoritma generik dengan polymorphism

- Dari contoh sebelumnya bisa dibuat method printLuas

```
void printLuas(Bangun b){
   Sytem.out.println(b.getLuas());
}
printLuas(s);
printLuas(l);
```

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

Penutup

- Bahasa Java merupakan bahasa yang cukup kompleks meskipun lebih sederhana dibanding C++
- Materi di slide ini hanya membahas dasar-dasar bahasa Java
  - Banyak API Java yang tidak mungkin selesai dibahas
  - Masih ada beberapa fitur Java tingkat lanjut