



Dasar Pemrograman Java

Entin Martiana



Topik:

- Membedakan antara valid dan invalid identifiers.
- Mengetahui Java technology keywords.
- Mengetahui 8 tipe data primitif.
- Mendefinisikan literal value untuk tipe data numerik dan tekstual.
- Mendefinisikan tipe data primitive
- Mengetahui nilai inisialisasi default.
- Konversi dan casting tipe data primitif.



Class Fundamentals: main method

- The *main()* Method

```
public static void main(String[] args)
```

- **public** : method `main()` dapat diakses oleh apa saja, termasuk java technology interpreter.
- **static** : keyword ini berfungsi untuk memberi tahu kompiler bahwa method `main` bisa langsung digunakan dalam context class yang bersangkutan. Untuk mengeksekusi/menjalankan method yang bertipe `static`, tidak diperlukan instance nya.
- **void** : menunjukkan bahwa method `main()` tidak mengembalikan nilai
- **main** : merupakan nama method utama dari program java
- **String [] args** : Menyatakan bahwa method `main()` menerima single parameter yaitu `args` yang bertipe array. Digunakan pada saat memasukkan parameter pada saat menjalankan program.

Contoh: `java TestGreeting args[0] args[1] ...`



Menganalisa program Java Pertama

- Sekarang, kita akan mencoba untuk menganalisa program Java pertama :

```
public class Hello
{
    /**
     * My first java program
     */
    public static void main(String[] args) {
        //menampilkan string "Hello world" pada layar
        System.out.println("Hello world!");
    }
}
```




- Baris pertama kode :

public class Hello

- menandakan nama class yaitu Hello. Dalam Java, semua kode seharusnya ditempatkan di dalam deklarasi class. Kita melakukannya dengan menggunakan kata kunci class. Sebagai tambahan, class menggunakan access specifier public, yang mengindikasikan bahwa class kita mempunyai akses bebas ke class yang lain dari package yang lain pula .



- Baris berikutnya yaitu yang terdiri atas kurung kurawal { menandakan awal blok.
- Pada kode ini, kita menempatkan kurung kurawal pada baris selanjutnya setelah deklarasi class, bagaimanapun, kita dapat juga meletakkan kurung kurawal ini setelah baris pertama dari kode yang kita tulis.

- 
- Tiga baris selanjutnya menandakan adanya komentar Java. Komentar adalah sesuatu yang digunakan untuk mendokumentasikan setiap bagian dari kode yang ditulis.

```
/**
```

```
* My first java program
```

```
*/
```

- Komentar dinyatakan dengan tanda “/*” dan “*/”. Segala sesuatu yang ada diantara tersebut diabaikan oleh compiler Java, dan mereka hanya dianggap sebagai koment



- Baris selanjutnya,
public static void main(String[] args) {
atau dapat juga ditulis sebagai berikut,
public static void main(String[] args)
{
- mengindikasikan nama suatu method dalam class Hello yang bertindak sebagai method utama. Method utama adalah titik awal dari suatu program Java
- Baris berikutnya :
System.out.println("Hello world!");
- menampilkan teks “Hello World!” pada layar. Perintah **System.out.println()**, menampilkan teks yang diapit oleh tanda double quote (“ ”) pada layar.



Komentar pada Java

- Komentar adalah catatan yang ditulis pada kode dengan tujuan sebagai bahan dokumentasi. Teks tersebut bukan bagian dari program dan tidak mempengaruhi jalannya program.
- Java mendukung tiga jenis komentar :
 - C++ style komentar satu baris
 - C style beberapa baris
 - komentar javadoc khusus



- ***Penulisan Komentar C++ Style***

- Komentar C++ style diawali dengan //. Semua teks setelah // dianggap sebagai komentar. Sebagai contoh,

```
// This is a C++ style or single line
```

comments

- ***Penulisan Komentar C Style***

- Komentar C-style atau juga disebut komentar beberapa baris diawali dengan /* dan diakhiri dengan */. Semua teks yang ada diantara dua tanda tersebut dianggap sebagai komentar.

```
/* this is an example of a
```

```
C style or multiline comments */
```

- ***Komentar Khusus javadoc***

- Komentar javadoc khusus digunakan untuk men-generate dokumentasi HTML untuk program Java memulai baris dengan /** dan mengakhirinya dengan */.

```
/**
```

```
This is an example of special java doc comments used for \n
generating an html documentation. It uses tags like:
```

```
@author Florenza Polihita Elektronika Negeri Surabaya
```

```
@version 1.2
```



Identifiers

- Nama yang digunakan oleh programmer untuk memberi nama pada variable, class, atau method. Identifier ini akan dicek oleh compiler, sehingga nama yang digunakan harus memenuhi aturan sbb :
 - Dimulai dengan a Unicode letter, underscore (`_`), or dollar sign (`$`). Tidak boleh dimulai dengan angka , !
 - Setelah karakter pertama, selanjutnya identifier dapat berupa huruf, \$, angka.
 - Dalam prakteknya, tidak ada batasan berapa jumlah karakter yang menyusun identifier
 - Case sensitive (huruf besar dan huruf kecil dibedakan) Tidak bisa menggunakan keyword Java sebagai identifier.

- Contoh

```
1. foobar // legal
2. BIGinterface // legal: embedded keywords
3. // are OK.
4. $incomeAfterExpenses // legal
5. 3_node5 // illegal: starts with a digit
6. !theCase // illegal: must start with
7. // letter, $, or _
```



Java Keywords and Reserved Words

- Java Keywords sering disebut juga sebagai reserved keywords
- Tidak dapat digunakan sebagai identifier.
- Tidak ada reserved words yang mempunyai sebuah huruf besar
- Assert ditambahkan di 1.4 dan enum ditambahkan di 1.5
- 2 keywords that are reserved in Java but which are not used : const dan goto

<code>abstract</code>	<code>boolean</code>	<code>break</code>	<code>byte</code>	<code>case</code>	<code>catch</code>
<code>char</code>	<code>class</code>	<code>const</code>	<code>continue</code>	<code>default</code>	<code>do</code>
<code>double</code>	<code>else</code>	<code>extends</code>	<code>final</code>	<code>finally</code>	<code>float</code>
<code>for</code>	<code>goto</code>	<code>if</code>	<code>implements</code>	<code>import</code>	<code>instanceof</code>
<code>int</code>	<code>interface</code>	<code>long</code>	<code>native</code>	<code>new</code>	<code>package</code>
<code>private</code>	<code>protected</code>	<code>public</code>	<code>return</code>	<code>short</code>	<code>static</code>
<code>strictfp</code>	<code>super</code>	<code>switch</code>	<code>synchronized</code>	<code>this</code>	<code>throw</code>
<code>throws</code>	<code>transient</code>	<code>try</code>	<code>void</code>	<code>volatile</code>	<code>while</code>
<code>assert</code>	<code>enum</code>				



Tipe data

- Tipe data mendefinisikan jenis data yang dinyatakan oleh variabel.
- Contohnya adalah sebuah data bertipe integer, merepresentasikan bahwa data tersebut bilangan bulat.
- Terdapat dua tipe data :
 - Tipe data primitif
 - Tipe data objek



Tipe data primitif

- Terdapat 8 tipe data primitif :
 - Logical - `boolean`
 - Textual - `char`
 - Integral - `byte`, `short`, `int`, **and** `long`
 - Floating - `double` **and** `float`



Tipe data primitif

Primitive Data Types and Their Effective Sizes

Type	Effective Representation Size (bits)	Type	Effective Representation Size (bits)
boolean	1	char	16
byte	8	short	16
int	32	long	64
float	32	double	64



Deklarasi variabel

- Sintak umum untuk mendeklarasikan dan menginisialisasi variabel
 - `<modifier> <Tipe data> <Nama variabel> = <nilai awal>`
- Contoh : mendeklarasikan private variabel id dengan tipe data int dan variabel id diberikan nilai awal 10
`private int id = 10;`



Scope variabel

- Variabel lokal
 - variabel yang dideklarasikan dalam sebuah method.
 - Variabel ini hanya bisa diakses dalam method tersebut dan variabel tersebut dihapus (destroyed) setelah method selesai dijalankan.
 - Variabel lokal disebut juga stack variabel karena disimpan dalam stack.
- Variabel instance
 - Variabel yang dideklarasikan dalam sebuah class tapi diluar method.
 - Merupakan variabel instance dari setiap objek yang dicreate dari class tsb dan hanya berlaku untuk 1 objek tsb.
 - Variabel instance tersimpan dalam heap.



Scope variabel

- Variabel static
 - Variabel instance dideklarasikan dengan modifier static dalam sebuah class (diluar method). Variabel ini dapat dibaca/dishare oleh semua objek dari class tersebut.



Literals

- Suatu nilai
- Contoh angka desimal, angka pecahan, karakter.
- Tidak bisa diletakkan disebelah kiri pada proses assignment
- Contoh
 - 'b' // char literal
 - 42 // int literal
 - false // boolean literal
 - 2546789.343 // double literal



Logical literals

- The `boolean` data type has two literals, `true` and `false`.
- For example, the statement:
 1. `boolean isBig = true;`
 2. `boolean isLittle = false;`

Note: boolean literal tidak boleh berharga 0 atau 1



char literals

- The range: $0 \sim 2^{16} - 1$.
- Java characters are in Unicode character (16-bit encoding).
- Expressed by enclosing the desired character in single quotes (‘ ’).
- Example:

```
char c = 'w' ;
```
- Express as a Unicode value specified using four hexadecimal digits, preceded by `\u`
- Example:

```
char c1 = '\u4567' ;
```



char literals

- Special Characters
 - ‘\n’ for new line
 - ‘\r’ for return
 - ‘\t’ for tab
 - ‘\b’ for backspace
 - ‘\f’ for formfeed
 - ‘\’ for single quote
 - ‘\”’ for double quote
 - ‘\\’ for backslash



Integral literals → byte, short, int and long

- Tipe default adalah int
- Untuk menentukan tipe long dengan cara meletakkan 'L' or 'l' setelah angka.
 - Contoh:
`long x = 25L;`



Integral literals → byte, short, int and long

- Dinyatakan dengan desimal, octal atau hexadecimal

```
2          The decimal value is 2
077       The leading 0 indicates an octal value
0xBAAC   The leading 0x indicates a
          hexadecimal value
```

```
class Octal {
    public static void main(String [] args) {
        int six = 06;      // Equal to decimal 6
        int seven = 07;   // Equal to decimal 7
        int eight = 010;  // Equal to decimal 8
        int nine = 011;   // Equal to decimal 9
        System.out.println("Octal 010 = " + eight);
    }
}
```

Output
Octal 010 = 8

```
class HexTest {
    public static void main (String [] args) {
        int x = 0X0001;
        int y = 0x7fffffff;
        int z = 0xDeadCafe;
        System.out.println("x = " + x + " y = " + y + " z = " + z);
    }
}
```

X = 1 y = 2147483647 z = -559035650



Integral

Ranges of the Integral Primitive Types

Type	Size	Minimum	Maximum
byte	8 bits	-2^7	$2^7 - 1$
short	16 bits	-2^{15}	$2^{15} - 1$
int	32 bits	-2^{31}	$2^{31} - 1$
long	64 bits	-2^{63}	$2^{63} - 1$



Floating-Point literals

- Range untuk tipe Float dan Double

Type	Size	Range
Float	32 bits	-3.4×10^{38} to 3.4×10^{38}
Double	64 bits	-1.7×10^{308} to 1.7×10^{308}



Floating-Point literals

- Floating point literal includes either a decimal point or one of the following:
 - E or e (add exponential value)
 - F or f (float)
 - D or d (double)
 - 3.14 → a simple floating point value (a double)
 - 6.02E23 → a large floating point value
 - 2.718F → a simple float size value
 - 123.4E306D → a large double value
- Default is double
 - Specify a float by putting an 'F' or 'f' after the number.
 - Example:
`float x = 2.5F;`



Note:

Semua tipe data primitif yang numerik (selain char dan boolean) adalah **signed**.



Nilai default

Initialization Values for Member Variables

Element Type	Initial Value	Element Type	Initial Value
byte	0	short	0
int	0	long	0L
float	0.0f	double	0.0d
char	'\u0000'	boolean	false
object reference	null		



Conversion of primitives

- Terjadi pada saat kompile.
- Conversion of a primitives bisa terjadi pada:
 - Assignment
 - Method call
 - Arithmetic promotion



Primitive Conversion: Assignment

- Terjadi ketika suatu nilai kita berikan pada suatu variabel yang tipe datanya berbeda dari data aslinya.
- Tipe data yang baru harus mempunyai ukuran lebih besar dari tipe data yang lama.

```
1. int i;  
2. double d;  
3. i = 10;  
4. d = i; // Assign an int value to a double variable
```

- Nilai d = 10.0



Primitive Conversion: Assignment

- Contoh konversi yang illegal

```
1. double d;  
2. short s;  
3. d = 1.2345;  
4. s = d; // Assign a double to a short variable
```

- Muncul error: possible loss of precision
- Karena tipe data short lebih kecil dari double.

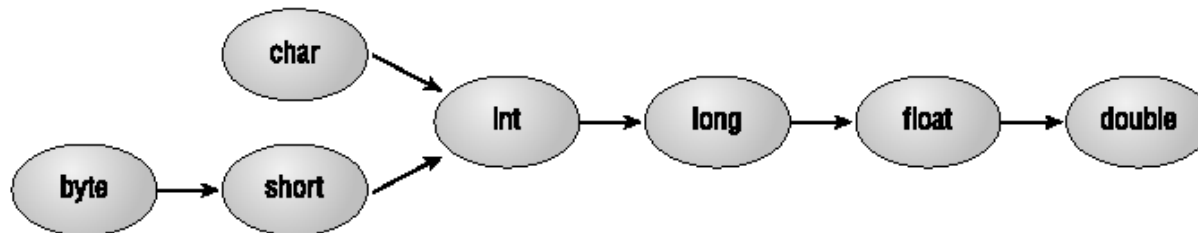


Aturan untuk primitive assignment conversion

- Boolean tidak bisa di konversi ke tipe data lain
- Non-boolean dapat di konversi ke tipe data lain selain boolean, konversi yang dilakukan adalah *widening conversion*
- Note: *widening conversion* adalah merubah tipe data suatu variabel ke tipe data yang ukuran bit nya lebih besar dari aslinya.

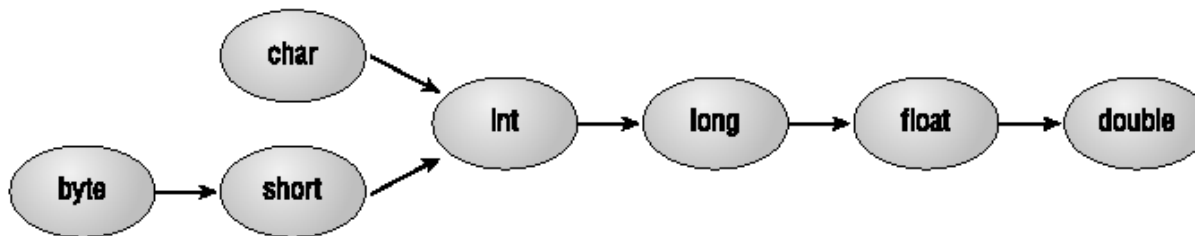
Java's widening conversions

- From a byte to a short, an int, a long, a float, or a double
 - From a short to an int, a long, a float, or a double
 - From a char to an int, a long, a float, or a double
 - From an int to a long, a float, or a double
 - From a long to a float or a double
 - From a float to a double
- **Note:** Konversi antar primitive types yang tidak mengikuti arah panah disebut dengan *narrowing conversion*.



Java's narrowing conversions

- From a byte to a char
- From a short to a byte or a char
- From a char to a byte or a short
- From an int to a byte, a short, or a char
- From a long to a byte, a short, a char, or an int
- From a float to a byte, a short, a char, an int, or a long
- From a double to a byte, a short, a char an int, a long, or a float



Note: Ubah arah panah!!



Primitive Conversion: Assignment

- **Ada yang istimewa tentang integral literal assignment**
- **Illegal** : 1.234 adalah literal untuk double sehingga tidak bisa di berikan pada float.
float f = 1.234;
- **Legal**: khusus untuk **integral literal** aturan assignment conversion dibebaskan.
byte b = 1;
short s = 2;
char c = 3;
- **Illegal**: Pembebasan assignment conversion untuk integral literal hanya untuk assigment terhadap nilai.
int i = 12;
byte b = i; → i adalah bukan nilai



Primitive Conversion: Method Call

- Terjadi ketika kita berusaha melewatkan suatu nilai variabel sebagai argumen suatu method, dimana tipe data variabel method tersebut berbeda dengan yang diterima.

```
1. float frads;  
2. double d;  
3. frads = 2.34567f;  
4. d = Math.cos(frads); // Pass float to method  
                        // that expects double
```

- **Hint:** `Math.cos(double d);`
- Pada contoh diatas frands yang bertipe float akan secara otomatis di konversi menjadi double.
- Pada contoh diatas terjadi widening conversions.



Primitive Conversion: Arithmetic Promotion

- Terjadi pada operasi matematika.
- Kompiler berusaha mencari tipe data yang sesuai dengan tipe data operan yang berbeda-beda.

```
1. short s = 9;  
2. int i = 10;  
3. float f = 11.1f;  
4. double d = 12.2;  
5. if ((-s * i) >= (f/d))  
6.     System.out.println(">>>>");  
7. else  
8.     System.out.println("<<<<");
```

- Penyelesaian:
 1. Short s dipromosikan ke int, selanjutnya di negatifykan.
 2. Hasil step 1 (int) dikalikan dengan int i.
Karena kedua operan bertipe int maka hasilnya adalah int.
 3. Float f di promosikan menjadi double, selanjutnya dibagi dengan double d.
Menghasilkan double.
 4. Hasil langkah 2 (int) dibandingkan dengan hasil langkah 3 (double). Int dipromosikan menjadi double.
 5. Hasil perbandingan adalah boolean.



Aturan: Arithmetic Promotion

- Unary operators: `+`, `-`, `++`, `--`, `~`
- Jika operan bertipe `byte`, `short`, atau `char`, maka dikonversikan ke `int`



Aturan: Arithmetic Promotion

- Binary operators: +, -, *, /, %, >>, >>>, <<, &, ^, |
- Jika salah satu operan adalah double, operan lain dikonversikan ke double.
- Jika salah satu operan adalah float, operan lain dikonversikan ke float.
- Jika salah satu operan adalah long, operan lain dikonversikan ke long.
- Selain tipe data diatas maka dikonversikan ke int.



Primitives and Casting

- *Casting* means explicitly telling Java to make a conversion.
- Cara: tambahkan tipe data yang diinginkan dalam tanda kurung sebelum nilai.

1. `int i = 5;`
2. `double d = (double)i;`

- Sama dengan:

1. `int i = 5;`
2. `double d = i;`



Primitives and Casting

- Are required when you want to perform a **narrowing** conversion.

```
1. short s = 259;
2. byte b = s; // Compile error
3. System.out.println("s = " + s + " , b = " + b);
```

- Pesan error = Explicit cast needed to convert short to byte.

- **Solusi:** dengan menambahkan casting

```
1. short s = 259;
2. byte b = (byte)s; // Explicit cast
3. System.out.println("b = " + b);
```

- Hasil : $b = 3$
- Kenapa $\rightarrow 259 = 1\ 0000\ 0011$
- The cast tells the compiler **“Yes, I really want to do it”**



Tipe Data Objek

- Objek dibuat dari class. Class adalah blueprint dari objek.
- Class sendiri ada dua macam:
 - Class yang sudah disediakan oleh Java

```
String s = new String("abc");  
Vector v = new Vector();
```
 - Class yang kita buat sendiri (dijelaskan pada pertemuan selanjutnya)