



Dasar Pemrograman Java

Pertemuan 2

Pemrograman Berbasis Obyek



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 primitive dan reference variable.
- Mendeklarasikan variabel bertipe class.
- Mengetahui nilai inisialisasi default.
- Konversi dan casting tipe data primitif.

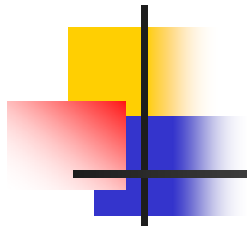


Java Keywords and Reserved Words

are considered as reserved keywords

- may not be used as identifiers.
- None of the reserved words have a capital letters
- 2 keyword that are reserved in Java but which are not used : const dan goto

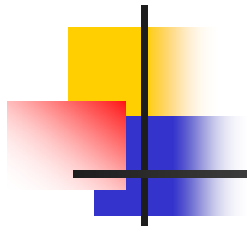
abstract	do	implements	private	this
boolean	double	import	protected	throw
break	else	instanceof	public	throws
byte	extends	int	return	transient
case	false	interface	short	true
catch	final	long	static	try
char	finally	native	strictfp	void
class	float	new	super	volatile
continue	for	null	switch	while
default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



Identifiers

- Nama yang digunakan oleh programmer untuk memberi nama pada variable, class, atau method.
- Can start with a Unicode letter, underscore (_), or dollar sign (\$)
- Are case-sensitive and have no maximum length
- Examples:

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 _



Primitive Types

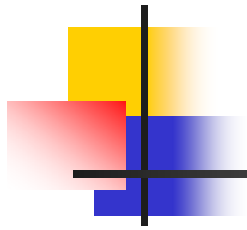
- The Java programming language defines eight primitive types:
 - Logical - `boolean`
 - Textual - `char`
 - Integral - `byte`, `short`, `int`, and `long`
 - Floating - `double` and `float`



Primitive Types

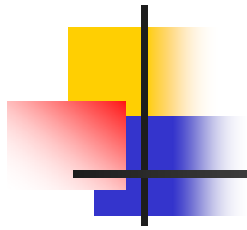
Primitive Data Types and Their Effective Sizes

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



Literals

- is a value
- cannot appear on the left side of assignments.

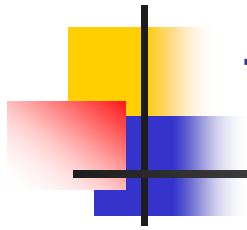


Logical literals

- 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



Textual Literals

- The range: $0 \sim 2^{16} - 1$.
- Java characters are in Unicode character (16-bit encoding).

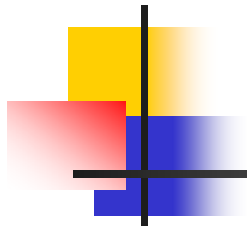


char literals

- Expressed by enclosing the desired character in single quotes (' ').
- Example:

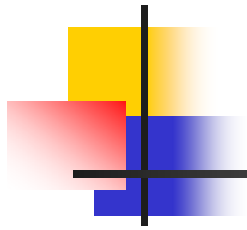
```
char c = 'w' ;
```
- Express as a Unicode value specified using four hexadecimal digits, preceded by \u
- Example:

```
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



String literals

- Is not a primitive data type; it is a class
- Represent sequences of characters
- Has its literal enclosed in double quotes (" ")
- Example:

```
String s = "Characters in strings are 16-bit  
Unicode.";
```

```
String s = "Good Morning !! \n";
```



Integral literals → byte, short, int and long

- Expressed in decimal, octal, or hexadecimal.
 - 2 The decimal value is 2
 - 077 The leading 0 indicates an octal value
 - 0xBAAC The leading 0x indicates a hexadecimal value
- Specify a long integer by putting an 'L' or 'l' after the number.
 - 'L' is preferred as it cannot be confused with the digit '1'.
 - Example:
`long x = 25L;`
- Has a default type of `int`



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

- 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

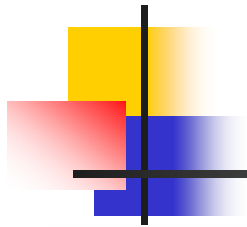
2.718F → a simple float size value

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:

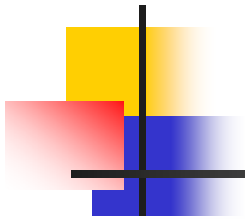
Semua tipe data primitif yang numerik (selain char dan boolean) adalah **signed**.



Member Variables Initialization

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		



Java Reference Types

- Beyond primitive types all others are reference types
- A *reference variable* contains a “handle” to an object.
- Example:

```
1 public class MyDate {  
2     private int day = 1;  
3     private int month = 1;  
4     private int year = 2000;  
5     public MyDate(int day, int month, int year) { ... }  
6     public void print() { ... }  
7 }
```

```
1 public class TestMyDate {  
2     public static void main(String[] args) {  
3         MyDate today = new MyDate(22, 7, 1964);  
4     }  
5 }
```

Reference variable



Conversion of primitives

- Terjadi pada saat kompilasi.
- 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.

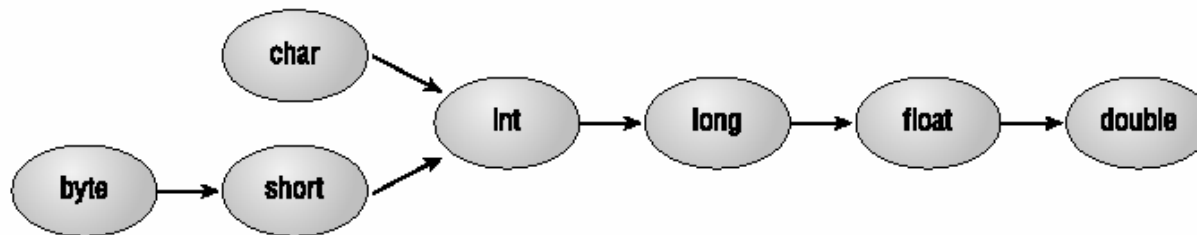


Aturan untuk primitive assignment conversion

- 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.

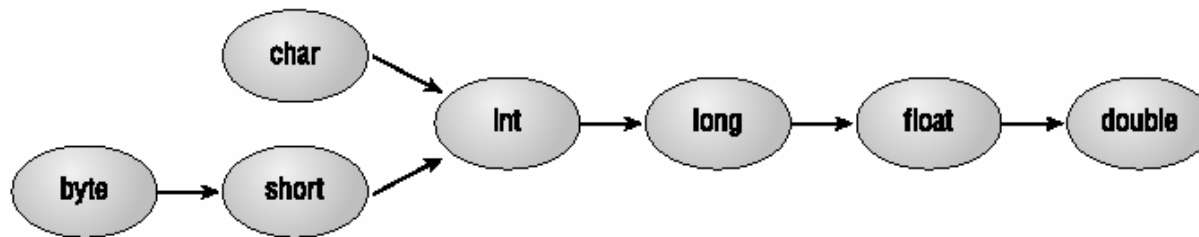
Java's widening conversions

- 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 assignment terhadap nilai.
int i = 12;
byte b = i; → i adalah bukan nilai



Primitive Conversion: Method Call

- Terjadi ketika kita berusaha melewati 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: Method Call

- Narrowing conversions tidak diperbolehkan
- Contoh ilegal:

```
double d = 12.0;
```

```
Object ob = myVector.elementAt(d);
```

- Hint : `myVector.elementAt(int i);`
- Hasil kompilasi: Incompatible type for method.
- Harus dilakukan casting secara eksplisit untuk mengkonversi dari double ke int



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 negatifkan.
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 → 259 = 1 0000 0011
- The cast tells the compiler **"Yes, I really want to do it"**