Pemrograman Berbasis Obyek

Dasar Pemrograman Java

Oleh Politeknik Elektronika Negeri Surabaya

2017

Politeknik Elektronika Negeri Surabaya
Departemen Teknik Informatika dan Komputer

Konten

- 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.
- Tugas

Identifiers

- Nama yang digunakan oleh programer untuk memberi nama pada variable, class, atau method.
- Tidak boleh mengandung spasi
- 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 _
```

Variable

- Instance Variables (Non-Static Fields)
- Class Variables (Static Fields)
- Local Variables
- Parameters

```
class Bicycle {
    int cadence = 0;
    int speed = 0;
    int gear = 1;

    void changeCadence(int newValue) {
                cadence = newValue;
    }

    void changeGear(int newValue) {
                gear = newValue;
    }

    void speedUp(int increment) {
                speed = speed + increment;
    }

    void applyBrakes(int decrement) {
                speed = speed - decrement;
    }

    void printStates() {
                System.out.println("cadence:" + cadence + " speed:" + speed + " gear:" + gear);
    }
}
```

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 keywords 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	assert***
enum****				

- ** added in 1.2
- *** added in 1.4
- **** added in 5.0

Primitive Types

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

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		

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.
- Contoh
  - boolean result = true;
  - char capitalC = 'C';
  - byte b = 100;
  - short s = 10000;
  - int i = 100000;

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 ~ 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 in package java.lang.String;
- 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
```

- Has a default type of `int`

- 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;
```

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

- 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**.

Using Underscore Characters in Numeric Literals

- In Java SE 7 and later, any number of underscore characters (_) can appear anywhere between digits in a numerical literal. This feature enables you, for example. to separate groups of digits in numeric literals, which can improve the readability of your code.
- For instance, if your code contains numbers with many digits, you can use an underscore character to separate digits in groups of three, similar to how you would use a punctuation mark like a comma, or a space, as a separator.

The following example shows other ways you can use the underscore in numeric literals:

- long creditCardNumber = 1234_5678_9012_3456L;
- long socialSecurityNumber = 999_99_9999L;
- float pi = 3.14_15F;
- long hexBytes = 0xFF_EC_DE_5E;
- long hexWords = 0xCAFE_BABE;
- long maxLong = 0x7fff_ffff_ffff_ffffL;
- byte nybbles = 0b0010_0101;
- long bytes = 0b11010010_01101001_10010100_10010010;

You can place underscores only between digits; you cannot place underscores in the following places:

- At the beginning or end of a number
- Adjacent to a decimal point in a floating point literal
- Prior to an F or L suffix
- In positions where a string of digits is expected

Contoh

- // **Invalid: cannot put underscores**
- // **adjacent to a decimal point**
- float pi1 = 3_.1415F;

- // **Invalid: cannot put underscores**
- // **adjacent to a decimal point**
- float pi2 = 3._1415F;

- // **Invalid: cannot put underscores**
- // **prior to an L suffix**
- long socialSecurityNumber1 = 999_99_9999_L;

- // This is an identifier, not a numeric literal
- int x1 = _52;

- // OK (decimal literal)
- int x2 = 5_2;

- // **Invalid: cannot put underscores**
- // **At the end of a literal**
- int x3 = 52_;

- // OK (decimal literal)
- int x4 = 5_____2;

- // **Invalid: cannot put underscores**
- // **in the 0x radix prefix**
- int x5 = 0_x52;

- // **Invalid: cannot put underscores**
- // **at the beginning of a number**
- int x6 = 0x_52;

- // OK (hexadecimal literal)
- int x7 = 0x5_2;

- // **Invalid: cannot put underscores**
- // **at the end of a number**
- int x8 = 0x52_;

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   }
```

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

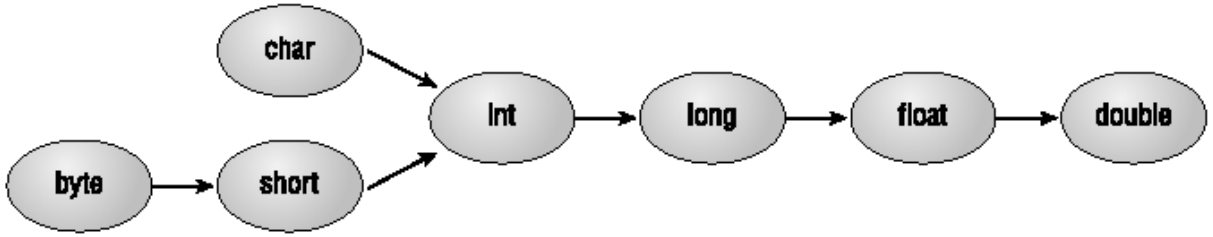
- Boolean tidak bisa di konversi ke tipe data lain
- Non-boolean dapat di konversi ke tipe data lain selain boolean, konversi yang dilakukan adalah *widening conversion*

- Note: *widening conversion* adalah merubah tipe data suatu variabel ke tipe data yang ukuran bit nya lebih besar dari aslinya.

Java's widening conversions (conversion)

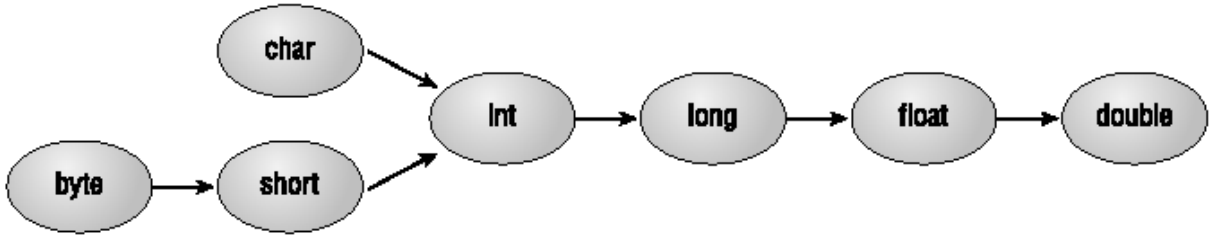
- From a byte to a short, an int, a long, a float, or a double
- From a short to an int, a long, a float, or a double
- From a char to an int, a long, a float, or a double
- From an int to a long, a float, or a double
- From a long to a float or a double
- From a float to a double

- **Note:** Konversi antar primitive types yang tidak mengikuti arah panah disebut dengan *narrowing conversion*.



Java's narrowing conversions (casting)

- 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

- Ilegal : 1.234 adalah literal untuk double sehingga tidak bisa di berikan pada float.

  float f = 1.234;

- **Ada yang istimewa tentang integral literal assignment**
- 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 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 pembandingan adalah boolean.

Aturan: Arithmatic Promotion

- Unary operators: +, -, ++, --, ~

- Jika operan bertipe byte, short, atau char, maka dikonversikan ke int

Aturan: Arithmatic 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;
```

- 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**"

Primitives and Casting

- Contoh ilegal:

```
double d = 12.0;
 Object ob = myVector.elementAt(d);
```

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

Tugas

1. Buatlah uraian yang berisi tentang spesifikasi 8 tipe data dasar !
2. Apakah yang dimaksud dengan casting (narrowing conversion) ?
3. Apakah yang dimaksud dengan konversi (widening conversion) ?

1. Oracle Java Documentation, The Java™ Tutorials, https://docs.oracle.com/javase/tutorial/, Copyright © 1995, Oracle 2015.
2. Tita Karlita, Yuliana Setrowati, Rizky Yuniar Hakkun, Pemrograman Berorientasi Obyek, PENS-2012
3. Sun Java Programming, Sun Educational Services, Student Guide, Sun Microsystems, 2001.
4. John R. Hubbard, Programming With Java, McGraw-Hill, ISBN: 0-07-142040-1, 2004.
5. Patrick Niemeyer, Jonathan Knudsen, Learning Java, O'reilly, CA, ISBN: 1565927184, 2000.
6. Philip Heller, Simon Roberts, Complete Java 2 Certification Study Guide, Third Edition, Sybex, San Francisco, London, ISBN: 0-7821-4419-5, 2002.
7. Herbert Schildt, The Complete Reference, Java™ Seventh Edition, Mc Graw Hill, Osborne, ISBN: 978-0-07-163177-8, 2007