# Tipe Data, Variabel dan Operator pada Java

## Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

1. Mengenal konsep tipe data primitif.
2. Mengenal operator pada Java.
3. Dapat melakukan type casting.
4. Mengenal konsep variabel dan konstanta.

## Alat & Bahan

Alat & Bahan Yang digunakan adalah hardware perangkat PC beserta Kelengkapannya berjumlah 40 PC serta Software IntelliJ IDEA yang telah terinstall pada masing-masing PC

## Dasar Teori

### Tipe Data

Java merupakan bahasa yang sangat ketat mengatur tipe data. Dalam Java, setiap variabel dan ekspresi harus memiliki tipe data, dan tipe ini didefinisikan dalam aturan yang ketat. Selanjutnya, setiap *assignment*, baik yang berupa *assignment* langsung maupun yang berasal dari melewatkan parameter dalam method, akan di-cek kompatibilitas tipenya.

Dalam Java, variabel dapat memiliki tipe primitif maupun *reference*. Variabel bertipe primitif hanya dapat menyimpan satu nilai sesuai dengan tipe yang dideklarasikan dalam satu waktu. Variabel dengan tipe *reference* menyimpan lokasi objek pada memori, dengan demikian variabel tersebut merujuk pada suatu objek dalam program yang dijalankan. Variabel tipe reference akan dibahas pada pertemuan mengenai tipe data objek.

#### Tipe Data Primitif

Terdapat delapan tipe data primitif dalam Java, yaitu byte, short, int, long, char, float, double dan boolean. Kedelapan tipe ini dapat dikelompokkan dalam empat grup:

* Integer: Meliputi byte, short, int dan long. Merupakan angka bertanda (plus atau minus).
* Floating-point: Meliputi float dan double, menyatakan angka dalam bentuk pecahan.
* Character: Hanya terdiri atas char, merepresentasikan simbol dalam set karakter (huruf dan angka). Java menggunakan metode unicode untuk merepresentasikan tipe data ini.
* Boolean: Hanya terdiri atas boolean, merepresentasikan nilai true atau false.

Untuk lebih jelasnya, perhatikan tabel tipe data dasar berikut.

Tabel 2‑1 Tipe data primitif pada Java

|  |  |  |
| --- | --- | --- |
| **Tipe** | **Panjang** | **Jangkauan** |
| long | 64 | -9.223.372.036.854.775.808 sampai 9.223.372.036.854.775.807 |
| int | 32 | -2.147.483.648 sampai 2.147.483.647 |
| short | 16 | -32.768 sampai 32.767 |
| byte | 8 | -128 sampai 127 |
| double | 64 | 4.9e-324 sampai 1.8e+308 |
| float | 32 | 1.4e-045 sampai 3.4e+038 |

#### Literal

Nilai konstan pada Java disebut dengan literal. Terdapat literal integer, floating-point, boolean, character, dan string.

* 1. Literal integer

Semua nilai angka merupakan literal integer dalam Java. Selain mengenal angka desimal, Java juga menyediakan angka oktal dan heksadesimal. Setiap literal integer memiliki tipe integral (int), jadi, pada dasarnya literal ini harus di-assign pada variable bertipe sama (integer). Jika tipe variabel bukan integer, literal tetap dapat di-assign selama masih dalam range tipe variabel tersebut.

Tabel berikut menunjukkan jenis literal integer dan *assignment*-nya.

Tabel 2‑2 Jenis literal integer

|  |  |
| --- | --- |
| **Jenis** | **Keterangan** |
| Desimal | Menyatakan bilangan basis sepuluh ( 1 – 10)  Contoh: 1, 390, 42 |
| Oktal | Menyatakan bilangan basis delapan, pada Java ditandai dengan angka 0 di depan bilangan  Contoh: 01, 03, 06 |
| Heksadesimal | Menyatakan bilangan basis enam belas, pada Java ditandai dengan angka 0-kali (0x).  Contoh: 0x4, 0xA, 0x12 |
| Assignment | int: int data = 10  short/ byte: short data2 = 10 (masih tidak masalah karena dalam range short)  long: long data3 = 10L (menandakan tipe long)  char: char data4 = ‘10’ (masih dalam range char) |

* 1. Literal floating-point

Angka floating-point menyatakan nilai decimal dengan bagian pecahannya. Nilai ini dapat dinyatakan baik dalam bentuk standar maupun notasi ilmiah. Tipe data default bagi floating-point dalam Java adalah double, jika ingin menyatakan dalam float harus ditambahkan F/f di belakang bilangan.

Contoh:

* Notasi standar: 2.0; 5.8; 125.258
* Notasi ilmiah: 2.523651E12, 4.56323E-5
* Assignment float: float bil = 2.34F
  1. Literal boolean

Hanya memiliki nilai true atau false, tidak memiliki padanan 0 atau 1. Digunakan pada variabel boolean, atau sebagai ekspresi pada operator boolean.

* 1. Literal character

Merupakan set karakter Unicode. Dapat dikonversikan ke dalam bentuk integer dan dimanipulasi oleh operator integer (seperti ditambah atau dikurang). Literal ini ditulis diantara kutip tunggal. Semua karakter ASCII dapat langsung ditulis sebagai literal ini, untuk yang tidak memungkinkan, digunakan *escape sequence*.

128 karakter pertama dari Unicode dan ASCII adalah sama.

Contoh:

* Notasi literal: ‘a’, ‘12’, ‘@’
* Notasi dengan escape sequence: ‘\’’, ‘\n’
* Notasi dengan octal atau heksadesimal: ‘\141’ (‘a’), ‘\u0061’ (\u menandakan heksadesimal, ‘\u0061’ adalah notasi ISO-Latin-1 untuk ‘a’)
  1. Literal string

Literal string ditandai dengan tanda kutip dua ( “ )

Contoh:

“Hello World”, “two\nlines”, “\’Dalam tanda kutip\’”

#### Assignment

*Assignment* adalah perintah untuk memasukan sebuah nilai ke dalam variabel. Operator *assignment* adalah tanda “sama dengan ( = )”. Bentuk umum dari operator adalah:

*var = expression*

Tipe data dari *var* harus kompatibel dengan tipe *expression*.

|  |  |
| --- | --- |
| package praktikum2;  class Assigment {  public static void main (String args []){  int A = 5; // assignment  System.out.println(A);  }  } | Apakah Outputnya?  ………………………… |

Kode baris di atas akan melakukan pemberian nilai (assignment) pada variabel A dengan 5, dan menampilkan kembali isi dari variabel A.

#### Operator

Operator dalam bahasa Java pada dasarnya dapat dibagi ke dalam empat bagian: aritmatika, *bitwise*, relasional dan logika.

1. Operator Aritmatika

Penggunaan operator aritmatika pada bahasa Java sama seperti dalam aljabar. Operan dari operator ini harus dalam tipe numerik, tidak bisa dalam boolean. Namun, tipe char dapat digunakan sebagai operan pada operator ini karena pada Java, char merupakan subset dari int. Tabel berikut menampilkan daftar operator aritmatika pada Java.

Tabel 2‑3 Operator aritmatika

|  |  |
| --- | --- |
| **Jenis** | **Keterangan** |
| + | Penjumlahan |
| - | Pengurangan (juga untuk menyatakan minus, suatu operator uner) |
| \* | Perkalian |
| / | Pembagian |
| % | Modulus |
| ++ | Increment |
| += | Assignment penjumlahan |
| -= | Assignment pengurangan |
| \*= | Assignment perkalian |
| /= | Assignment pembagian |
| %= | Assignment modulus |
| -- | Decrement |

1. Operator *Bitwise*

Operator *bitwise* melakukan operasi pada bilangan integer (int, long, byte, short dan char) per bit. Jadi, operasi akan dilakukan pada bit-bit yang menyusun suatu bilangan, misal nilai 10 akan direpresentasikan oleh bit 1010. Yang perlu diingat, bilangan integer pada Java merupakan bilangan bertanda (*signed*), jadi bilangan negatif dan positif harus dibedakan. Untuk menyatakan bilangan negatif pada Java dilakukan operasi komplemen 2 dengan cara menginversikan bit suatu bilangan dan menambah satu setelahnya.

Tabel 2‑4 Operator bitwise

|  |  |
| --- | --- |
| **Jenis** | **Keterangan** |
| ~ | Bitwise unary NOT |
| & | Bitwise AND |
| | | Bitwise OR |
| ^ | Bitwise Exclusive OR |
| >> | Shift right |
| >>> | Shift right zero fill |
| << | Shift left |
| &= | Bitwise AND assignment |
| |= | Bitwise OR assignment |
| ^= | Bitwise Exclusive OR assignment |
| >>= | Shift right assignment |
| >>>= | Shift right zero fill assignment |
| <<= | Shift left assignment |

1. Operator Relasional

Operator relasional menyatakan hubungan antara satu operan dengan operan lainnya. Lebih jelasnya, operator ini menyatakan kesetaraan dan urutan (lebih besar, lebih kecil). Keluaran dari operasi ini adalah nilai boolean.

Tabel 2‑5 Operator relasional

|  |  |
| --- | --- |
| **Jenis** | **Keterangan** |
| == | Sebanding dengan (sama dengan) |
| != | Tidak sama dengan |
| > | Lebih besar dari |
| < | Lebih kecil dari |
| >= | Lebih besar atau sama dengan |
| <= | Lebih kecil atau sama dengan |

1. Operator Logika Boolean

Hanya bekerja dengan nilai boolean, dan menghasilkan nilai boolean juga.

Tabel 2‑6 Operator logika boolean

|  |  |
| --- | --- |
| **Jenis** | **Keterangan** |
| & | Logika AND |
| | | Logika OR |
| ^ | Logika XOR |
| || | Short-circuit OR |
| && | Short-circuit AND |
| ! | Logika NOT |
| &= | Assignment AND |
| |= | Assignment OR |
| ^= | Assignment XOR |
| == | Sama dengan |
| != | Tidak sama dengan |
| ?: | Operator ternary if-then-else |

1. Urutan operator

Tabel berikut menyatakan urutan operator, dari yang paling tinggi (jadi harus dilaksakan terlebih dahulu) sampai yang paling rendah. Jika operator dengan tingkatan yang sama ada dalam satu baris, maka urutan pengerjaan adalah dari kiri ke kanan.

Tabel 2‑7 Urutan operator

|  |  |  |  |
| --- | --- | --- | --- |
| **Paling tinggi** |  |  |  |
| ( ) | [ ] | . |  |
| ++ | -- | ~ | ! |
| \* | / | % |  |
| + | - |  |  |
| >> | >>> | << |  |
| > | >= | < | <= |
| == | != |  |  |
| & |  |  |  |
| ^ |  |  |  |
| | |  |  |  |
| && |  |  |  |
| || |  |  |  |
| ?: |  |  |  |
| = | op= |  |  |
| **Paling** **rendah** |  |  |  |

Berikut contoh kode untuk beberapa operator.

|  |  |
| --- | --- |
| package praktikum 3;  class DemonstrateOp {  public static void main(String args[]){  //Contoh operator aritmatika  System.out.println("Operator aritmatika pada integer");  int a = 1 + 1, b = a \* 3, c = b / 4;  System.out.println("a = " + a); System.out.println("b = " + b);  System.out.println("c = " + c);  System.out.println("Operator modulus");  int x = 8;  System.out.println("x = " + x);  System.out.println("x mod 10 = " + x%10);  //Contoh increment / decrement  System.out.println("Operator modulus");  int e = 3, f = ++e, g = f--;  System.out.println("e = " + e); System.out.println("f = " + f);  System.out.println("g = " + g);  //Shift right  int i, num = 0xFFFFFFE;  for(i=0; i<4; i++) {  num = num << 1;  System.out.println(num);  }  //Operator logika Boolean  boolean bool1 = true, bool2 = false;  boolean bool3 = bool1 | bool2;  boolean bool4 = bool1 ^ bool2;  boolean bool5 = (!bool1 & bool2) | (bool1 & !bool2);  System.out.println(" bool1 = " + bool1);  System.out.println(" bool2 = " + bool2);  System.out.println(" bool1|bool2 = " + bool3);  System.out.println(" bool1^bool2 = " + bool4);  System.out.println("!bool1 & bool2| bool1 & !bool2 = " + bool6);  }  } | Apakah Outputnya?  …………………………  …………………………  …………………………  …………………………  ………………………… |

#### Compound Assignment

Compound assignment merupakan kombinasi dari operator yang bertujuan untuk merperpendek suatu baris perintah.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Simbol** | **Sama Dengan** |  | **Simbol** | **Sama Dengan** |
| A += 5 | A = A +5 |  | A++ | A = A +1 |
| A -= 5 | A = A – 5 |  | ++A | A = A + 1 |
| A /= 5 | A = A / 5 |  | A-- | A = A - 1 |
| A \*= 5 | A = A \* 5 |  | --A | A = A - 1 |
| A %= 5 | A = A % 5 |  |  |  |

Perhatikan A++ dan ++A. Memang fungsinya sama, yaitu menambahkan satu ke variable A. Tetapi ada perbedaannya pada saat pemakaian, yaitu urutan eksekusinya dalam suatu perintah. Begitu juga untuk A—dan –A. Untuk lebih jelasnya perhatikan contoh kode berikut.

|  |  |
| --- | --- |
| package praktikum 4;  Class DemoIncr {  public static void main (String args[]){  int A = 5;  int B = A++;  System.out.println(“Nilai A ” + A);  System.out.println(“Nilai B ” + B);  A = 6;  B = ++A;  System.out.println(“Nilai A ” + A);  System.out.println(“Nilai B ” + B);  }  } | Apakah Outputnya?  …………………………  …………………………  …………………………  …………………………  ………………………… |

#### Inisialisasi Variabel

Sebuah variabel bisa diartikan sebagai sebuah penampung. Kita bisa menempatkan nilai apa saja ke variabel tersebut. Proses penempatan tersebut dinamakan assignment, sementara penempatan nilai pertama pada variabel dinamakan inisialisasi. Namun, sebelumnya variabel harus dideklarasikan terlebih dahulu. Bentuk dasar deklarasi variabel pada Java adalah sebagai berikut:

*type identifier [=value][, identifier[=value] …;*

*Type* dapat berupa tipe data primitif atau nama kelas atau interface. *Identifier* adalah nama variabel, juga digunakan pada method dan kelas. Aturan pemberian nama *identifier* pada Java adalah sebagai berikut:

1. *Identifier* dapat terdiri atas huruf, angka, garis bawah ( \_ ) dan tanda dolar ( $ ), namun tidak boleh diawali dengan angka.
2. Tidak boleh terpisah atau mengandung simbol (seperti ? atau % )
3. Tidak boleh merupakan reserved word

Sementara konvensi penulisan identifier dalam Java adalah sebagai berikut:

1. Variabel dan method diawali dengan huruf kecil. Jika identifier terdiri atas lebih dari satu suku kata, gunakan aturan “camel case” dimana huruf awal kata berikutnya dijadikan huruf besar.
2. Nama kelas, sebaliknya, diawali dengan huruf besar.
3. Sebaiknya tidak menggunakan tanda dolar karena biasanya merupakan nama yang dibuat otomatis oleh suatu tools (tapi tidak berarti dilarang)

Inisialisasi variabel dapat dengan cara memasukkan nilai literal langsung ke dalam variabel, dapat juga berupa nilai dari variabel lainnya, atau nilai kembalian dari suatu method (diterangkan pada bab lain). Untuk lebih jelasnya, perhatikan contoh berikut.

|  |  |
| --- | --- |
| package praktikum 5;  class DemoVar {  int i = 10;  int x = i;  public static void main (String args[]){  System.out.println(“Isi dari variabel i = ” + i + “ dan isi dari variabel x = ” + x);  x = 15;  System.out.println(“Isi dari variabel i = ” + i + “ dan isi dari variabel x = ” + x);  }  } | Apakah Outputnya?  …………………………  ………………………… |

Pada program diatas, variabel i diisi dengan nilai literal 10, sedangkan variabel x diisi dengan nilai dari variabel i. Walaupun awalnya kedua variabel memiliki nilai yang sama, namun perubahan nilai pada salah satu variabel tidak otomatis akan mempengaruhi variabel lainnya.

#### const (konstanta)

Selain variabel, dapat digunakan konstanta, yaitu nilai yang tidak akan berubah. Penulisan konstanta dalam bahasa Java adalah dengan menambahkan keyword final. Melakukan perubahan pada suatu konstanta akan memunculkan error. Untuk lebih jelasnya, perhatikan contoh berikut.

|  |  |
| --- | --- |
| package praktikum 6;  class DemoConst {  final int LUCKY\_NUMBER = 7;  final float PI = 3.14159;  final char NEW\_LINE = '\n';    public static void main (String args[]){  System.out.println(LUCKY\_NUMBER + “” + NEW\_LINE);  System.out.println(PI + “” + NEW\_LINE);  PI = 5;  }  } | Apakah Outputnya?  …………………………  ………………………… |

Konstanta ditulis dengan huruf kapital, jika terdapat lebih dari satu suku kata, maka dipisahkan dengan tanda garis bawah. Baris terakhir program akan menyebabkan program tidak bisa di-*compile*.

#### *Type conversion and Casting*

Seperti yang telah disebutkan pada sub-bab literal, kita dapat menempatkan nilai dengan tipe data yang berbeda dari variabel, selama nilai tersebut masih dalam range tipe data variabel (kompatibel). Pengubahan (konversi) tipe data yang masih berada dalam range yang kompatibel disebut konversi otomatis (*automatic conversion*). Selain itu, harus dilakukan casting terhadap data.

1. *Automatic* *Conversion*

*Automatic* *conversion* dari literal dengan tipe data yang berbeda dari variabel dapat dilakukan jika:

* Kedua tipe (bilangan dan variabel) kompatibel
* Tipe data tujuan (variabel) lebih besar dari tipe data asal (literal)

1. *Casting*

*Casting* dilakukan pada *assignment* dari dua tipe data yang tidak kompatibel, biasanya karena tipe data tujuan lebih kecil daripada tipe data asal (misal, memasukkan nilai int ke dalam variabel bertipe byte). Cara melakukan casting adalah dengan menuliskan tipe data yang di-cast di depan nilai:

*(target-type) value*

Untuk lebih jelasnya perhatikan contoh kode berikut.

|  |  |
| --- | --- |
| package praktikum6;  class Conversion{  public static void main (String args[]){  byte b; int i = 153; double d = 112.45;  System.out.println("\nKonversi int menjadi byte.");  b = (byte) i;  System.out.println("i dan b " + i + " " + b);  System.out.println("\nKonversi double menjadi int.");  i = (int) d;  System.out.println("i dan d " + i + " " + d);  System.out.println("\nKonversi double menjadi byte.");  b = (byte) d;  System.out.println("b dan d " + b + " " + d);  }  } | Apakah Outputnya?  …………………………  …………………………  ………………………… |

### Tips menulis Program

Berikut beberapa tips dalam menulis program:

1. Comment

Comment diperlukan dalam sebuah program untuk memudahkan si Programmer itu sendiri. Setidaknya juga masih berguna untuk programmer lain jika ingin melanjutkan programnya.

1. Penjelasan singkat (about)

Setiap program sebaiknya dibuat sebuah penjelasan singkat tentang program tersebut, misalnya nama program, nama pembuat, fungsi program, dan lain-lain.

1. Penulisan class

Nama class merupakan nama dari file Java yang dibuat. Jadi, untuk class HelloWorld misalnya, maka nama filenya pun harus HelloWorld.java

1. Indentansi

Dalam penulisan kode program ada baiknya jika memperhatikan pengindentasian/penjorokan ke dalam. Indentasi sangat berguna untuk membuat program yang mudah dibaca. Indentasi tidak akan mempengaruhi kerja compiler (dengan ataupun tanpa indentasi akan sama saja), namun program yang ditulis dengan indentasi yang baik akan lebih mudah ditelusuri.