Laporan Praktikum

Algoritma Pemograman



Disusun Oleh :

**Ahda Rindang Al-Amin (2311531003)**

Dosen Pengampu : Dr. Wahyudi, S.T, M.T

Asisten Labor : Rahmad Dwirizki Olders

Departemen Informatika

Fakultas Teknologi Informasi

Universitas Andalas

Tahun 2023

**Tipe Data, Operator Aritmatika, Operator Logika, dan Operator Relasional Bahasa Pemograman Java**

1. **Tujuan Praktikum**
2. Memahami berbagai macam tipe data pada bahasa pemograman Java.
3. Memahami operator aritmatika yang bisa diterapkan dan cara menerapkannya pada bahasa pemograman Java.
4. Memahami operator logika yang bisa diterapkan dan cara menerapkannya pada bahasa pemograman Java.
5. Memahami operator relasional yang bisa diterapkan dan cara menerapkannya pada bahasa pemograman Java.
6. **Pendahuluan**

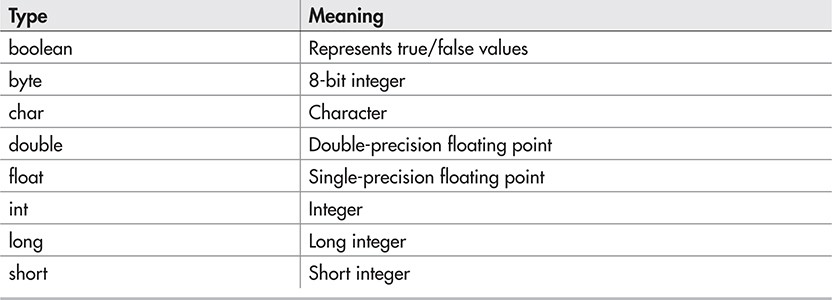
### Tipe Data

Tipe data sangat penting di Java karena merupakan bahasa pengetikan yang kuat. Ini berarti bahwa semua operasi diperiksa tipenya oleh kompiler untuk kompatibilitas tipe. Operasi ilegal tidak akan dikompilasi. Oleh karena itu, pemeriksaan tipe yang kuat membantu mencegah kesalahan dan meningkatkan keandalan. Untuk mengaktifkan pemeriksaan tipe yang kuat, semua variabel, ekspresi, dan nilai memiliki tipe.

Misalnya, tidak ada konsep variabel “type-less”. Selain itu, jenis suatu nilai menentukan operasi apa yang diperbolehkan pada nilai tersebut. Operasi yang diperbolehkan pada satu jenis mungkin tidak diperbolehkan pada jenis lainnya.

### Java’s Primitive Types

Java berisi dua kategori umum tipe data bawaan: object-oriented and non-object-oriented.. Tipe object-oriented pada Java ditentukan oleh kelas, dan diskusi tentang kelas ditunda hingga nanti. Namun, inti dari Java terdapat delapan tipe data primitif (juga disebut elemental atau simple). Istilah primitif digunakan di sini untuk menunjukkan bahwa tipe ini bukanlah objek dalam pengertian object-oriented, melainkan nilai biner normal. Tipe primitif ini bukan objek karena alasan efisiensi. Semua tipe data Java lainnya dibuat dari tipe primitif ini.

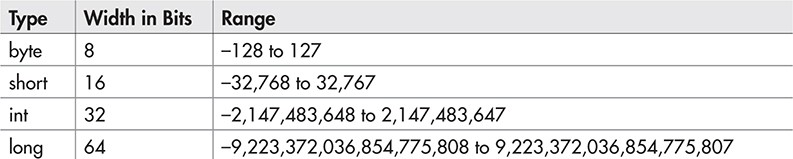


Java secara ketat menentukan rentang dan perilaku untuk setiap tipe primitif, yang harus didukung oleh semua implementasi Mesin Virtual Java. Karena persyaratan portabilitas Java, Java tidak berkompromi dalam hal ini. Misalnya, **int** sama di semua lingkungan eksekusi.

Hal ini memungkinkan program menjadi portabel sepenuhnya. Tidak perlu menulis ulang kode agar sesuai dengan platform tertentu. Meskipun menentukan secara ketat rentang tipe primitif dapat menyebabkan sedikit penurunan kinerja di beberapa lingkungan, hal ini diperlukan untuk mencapai portabilitas.

1. **Integers**

Java mendefinisikan empat tipe integer: **byte**, **short**, **int**, dan **long**, yang ditampilkan di sini:



Seperti yang diperlihatkan tabel, semua tipe bilangan bulat diberi tanda nilai positif dan negatif. Java tidak mendukung *unsigned integers* (hanya positif). Banyak bahasa komputer lain yang mendukung bilangan bulat bertanda dan tidak bertanda. Namun, desainer Java merasa bahwa *unsigned integers* tidak diperlukan.

Tipe integer yang paling umum digunakan adalah **int**. Variabel bertipe **int** sering digunakan untuk mengontrol loop, untuk mengindeks array, dan untuk melakukan matematika bilangan bulat tujuan umum.

Saat Anda membutuhkan bilangan bulat yang memiliki rentang lebih besar dari **int**, gunakan **long**.

1. **Floating-Point Types**

Tipe floating-point dapat mewakili bilangan desimal. Ada dua jenis tipe floating-point, **float** dan **double**, yang masing-masing mewakili bilangan presisi tunggal dan ganda. Tipe **float** lebarnya 32 bit dan tipe **double** lebarnya 64 bit.

Dari keduanya, **double** adalah yang paling umum digunakan, dan banyak fungsi matematika di kelas Java menggunakan nilai **double**.

1. **Character**

Di Java, karakter bukanlah kuantitas 8-bit seperti di banyak bahasa komputer lainnya. Sebaliknya, Java menggunakan Unicode. Unicode mendefinisikan kumpulan karakter yang dapat mewakili semua karakter yang ditemukan dalam semua bahasa manusia. Di Java, **char** adalah tipe 16-bit unsigned yang memiliki rentang 0 hingga 65.535. Kumpulan karakter ASCII 8-bit standar adalah subset Unicode dan berkisar antara 0 hingga 127. Dengan demikian, karakter ASCII masih merupakan karakter Java yang valid.

Character dapat diberi nilai dengan mengapit karakter dalam tanda kutip tunggal. Misalnya, ini memberikan variabel **ch** huruf X:



Anda dapat mengeluarkan nilai **char** menggunakan pernyataan **println()**. Misalnya, baris ini menampilkan nilai dalam **ch**:



Karena **char** adalah tipe unsigned 16-bit, berbagai manipulasi aritmatika dapat dilakukan pada variabel **char**.

1. **String**

Java mendukung satu tipe literal lainnya: **string**. **String** adalah sekumpulan karakter yang diapit oleh tanda kutip ganda. Misalnya,

adalah sebuah **string**.

Selain karakter normal, **string** juga dapat berisi satu atau lebih rangkaian escape yang baru saja dijelaskan.

### Boolean

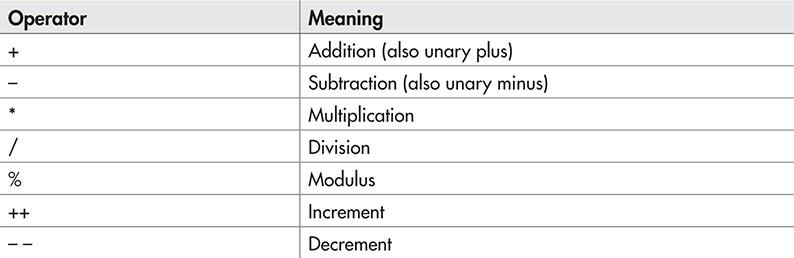
Tipe **boolean** mewakili nilai benar/salah. Java mendefinisikan nilai benar dan salah menggunakan kata-kata yang dicadangkan benar dan salah. Jadi, variabel atau ekspresi bertipe **boolean** akan menjadi salah satu dari dua nilai ini.

### Operator

Java kaya akan lingkungan operator. Operator adalah simbol yang memberitahu kompiler untuk melakukan manipulasi matematika atau logika tertentu. Java memiliki empat kelas operator umum: aritmatika, bitwise, relasional, dan logika. Java juga mendefinisikan beberapa operator tambahan yang menangani situasi khusus tertentu.

### Operator Aritmatika

Java mendefinisikan operator aritmatika berikut:



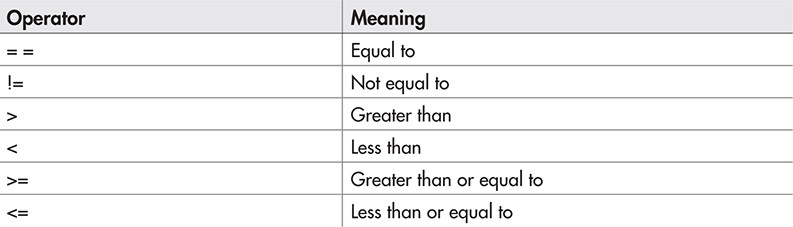
Operator +, –, \*, dan / semuanya bekerja dengan cara yang sama di Java seperti halnya dalam bahasa komputer lainnya (atau aljabar, dalam hal ini). Ini dapat diterapkan pada tipe data numerik bawaan apa pun. Mereka juga dapat digunakan pada objek bertipe **char**.

Meskipun tindakan operator aritmatika sudah diketahui oleh semua pembaca, beberapa situasi khusus memerlukan penjelasan. Pertama, ingat bahwa ketika / diterapkan pada integer, sisanya akan terpotong; misalnya, 10/3 sama dengan 3 dalam pembagian integer. Anda dapat memperoleh sisa pembagian ini dengan menggunakan operator modulus %. Ini menghasilkan sisa pembagian integer. Misalnya, 10 % 3 adalah 1. Di Java, % dapat diterapkan pada tipe integer dan floating-point. Jadi, 10.0 % 3.0 juga 1.

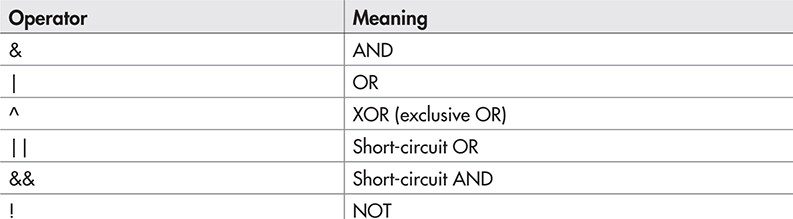
### Operator Relasional dan Logika

Dalam istilah operator relasional dan operator logika, relasional mengacu pada hubungan yang dimiliki suatu nilai satu sama lain, dan logika mengacu pada cara di mana nilai benar dan salah dapat dihubungkan bersama. Karena operator relasional memberikan hasil benar atau salah, operator tersebut sering kali bekerja dengan operator logika.

Operator relasional :

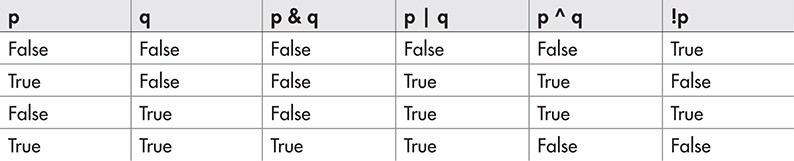


Operator logika :



Hasil dari operator relasional dan logika adalah nilai **boolean**. Di Java, semua objek dapat dibandingkan persamaan atau pertidaksamaannya menggunakan = = dan !=. Namun, operator perbandingan, <, >, <=, atau >=, hanya dapat diterapkan pada tipe yang mendukung hubungan pengurutan. Oleh karena itu, semua operator relasional dapat diterapkan ke semua tipe numerik dan tipe **char**. Namun, nilai bertipe **boolean** hanya dapat dibandingkan untuk kesetaraan atau ketidaksetaraan, karena nilai benar dan salah tidak diurutkan. Misalnya, **true > false** tidak ada artinya di Java.

Untuk operator logika, operannya harus bertipe **boolean**, dan hasil operasi logikanya harus bertipe **boolean**. Operator logika, &, |, ^, dan !, mendukung operasi logika dasar AND, OR, XOR, dan NOT, berdasarkan tabel kebenaran berikut:



1. **Metode Praktikum**
2. Operator Aritmatika
3. Buat Class pada IDE untuk menulis program Java.
4. Tulis program berikut:

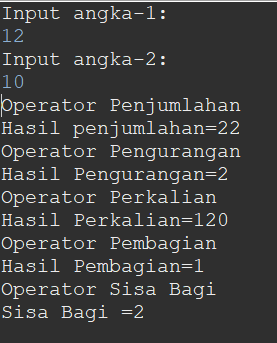


Pada program tersebut, karena angka-angka yang akan dilakukan operasi aritmatika diinput setelah program dijalankan, maka digunakan pemanggilan input dengan impor Scanner. Pada baris ke-9, Scanner dipanggil sehingga program bisa menerima input setelah dijalankan. Bilangan A1 dan A2 diinput dengan menyatakan **nextInt()** (perhatikan baris ke-11 dan ke-13). Untuk mengakhiri proses input, nyatakan **close()** (baris ke-14).

Pada program di atas, telah dituliskan berbagai operasi aritmatika yang ada di bahasa pemograman Java.

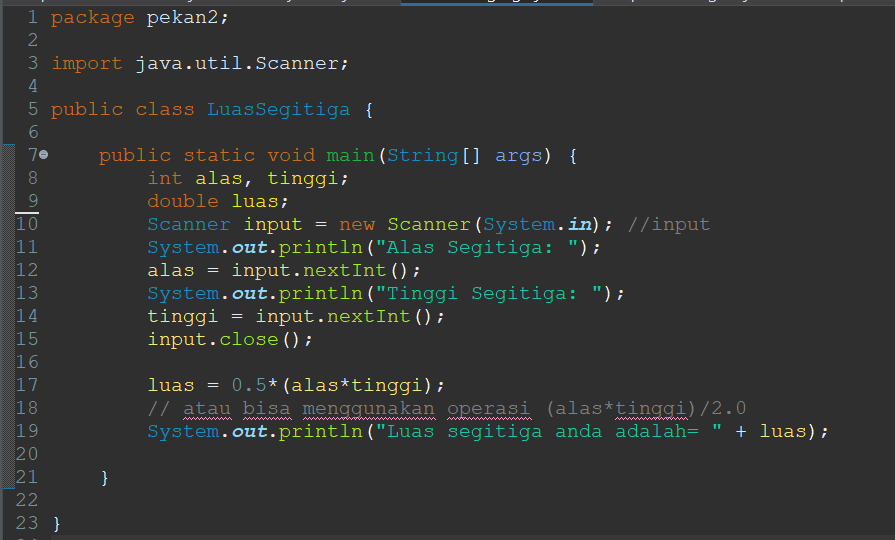
* Pada baris ke-18, **hasil** dinyatakan dengan menjumlahkan **A1** dan **A2**. Untuk melakukan operasi penjumlahan di Java, gunakan simbol plus (+).
* Pada baris ke-22, **hasil** dinyatakan dengan mengurangkan **A1** dengan **A2**. Untuk melakukan operasi pengurangan di Java, gunakan simbol minus (-).
* Pada baris ke-26, **hasil** dinyatakan dengan mengalikan **A1** dengan **A2**. Untuk melakukan operasi perkalian di Java, gunakan simbol bintang (\*).
* Pada baris ke-30, **hasil** dinyatakan dengan membagi **A1** dengan **A2**. Untuk melakukan operasi pembagian di Java, gunakan simbol garis miring (/).
* Pada baris ke-34, **hasil** dinyatakan sebagai hasil bagi **A1** terhadap **A2**, atau disebut operasi modulus. Untuk melakukan operasi modulus di Java, gunakan simbol persen (%).

1. Jalankan program tersebut. Masukkan input angka A1 dan A2, maka output program akan muncul seperti berikut ini.

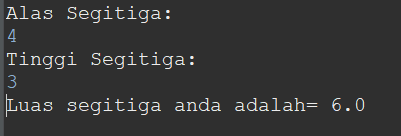


Dapat dilihat, input A1 adalah 12 dan A2 adalah 10. Dan tiap operator aritmatika berjalan sesuai fungsinya. Karena input menggunakan tipe data integer, maka output pun menampilkan bilangan bulat (integer).

1. Contoh program operator aritmatika lainnya adalah dengan mencari luas dari bangun segitiga. Tulis program berikut :



1. Jalankan program tersebut. Masukkan input angka alas dan tinggi, maka output program akan muncul seperti berikut ini.



Karena tipe data **hasil** adalah double, maka output hasil tidak bilangan bulat 6, namun 6.0 yang merupakan float-point.

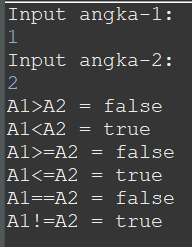
1. Operator Relasional
2. Buat Class pada IDE untuk menulis program Java.
3. Tulis program berikut:



Untuk melakukan operator relasional, tipe data dari hasil operator haruslah boolean. Pada program ini, nilai A1 dan A2 juga diinputkan setelah program dijalankan, jadi diperlukan impor Scanner.

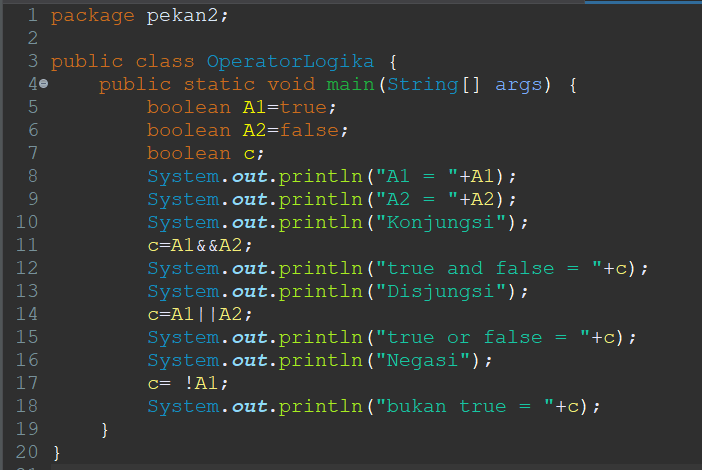
* Pada baris ke-16, **hasil** menentukan apakah *true or false* A1 besar dari A2. Untuk operasi relasional ‘besar dari’ pada bahasa pemograman Java menggunakan simbol >.
* Pada baris ke-18, **hasil** menentukan apakah *true or false* A1 kecil dari A2. Untuk operasi relasional ‘kecil dari’ pada bahasa pemograman Java menggunakan simbol <.
* Pada baris ke-20, **hasil** menentukan apakah *true or false* A1 besar dari atau sama dengan A2. Untuk operasi relasional ‘besar dari atau sama dengan’ pada bahasa pemograman Java menggunakan simbol >=.
* Pada baris ke-22, **hasil** menentukan apakah *true or false* A1 kecil dari atau sama dengan A2. Untuk operasi relasional ‘kecil dari atau sama dengan’ pada bahasa pemograman Java menggunakan simbol <=.
* Pada baris ke-24, **hasil** menentukan apakah *true or false* A1 sama dengan A2. Untuk operasi relasional ‘sama dengan’ pada bahasa pemograman Java menggunakan simbol = = .
* Pada baris ke-26, **hasil** menentukan apakah *true or false* A1 tidak sama dengan A2. Untuk operasi relasional ‘tidak sama dengan’ pada bahasa pemograman Java menggunakan simbol !=.

1. Jalankan program tersebut, masukkan input angka A1 dan A2, maka output program akan muncul seperti berikut ini.



Dengan input A1 adalah 1 dan A2 adalah 2, maka dihasilkan output **hasil** *true* atau *false* untuk setiap operator relasional.

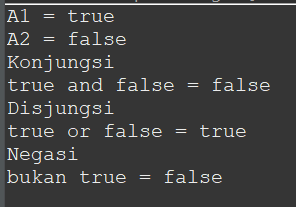
1. Operator Logika
2. Buat Class pada IDE untuk menulis program Java.
3. Tulis program berikut:



Operator logika dijalankan berdasarkan tabel kebenaran. Oleh karena itu, tipe data yang digunakan tentu saja **boolean**. Pada program di atas, A1 dinyatakan *true* dan A2 dinyatakan *false*.

* Pada baris ke-11, **c** dinyatakan sebagai logika AND. Operator logika AND pada bahasa pemograman Java menggunakan simbol &&.
* Pada baris ke-11, **c** dinyatakan sebagai logika OR. Operator logika OR pada bahasa pemograman Java menggunakan simbol | | .
* Pada baris ke-11, **c** dinyatakan sebagai logika NOT. Operator logika NOT pada bahasa pemograman Java menggunakan simbol !.

1. Jalankan program tersebut, maka output yang dihasilkan akan seperti ini:



1. **Kesimpulan Praktikum**

Pada bahasa pemograman Java, terdapat berbagai tipe data. Yaitu **int**, **double**, **boolean**, **string**, dsb. Berbagai tipe data diperlukan untuk fungsi yang berbeda-beda pula. Dengan penggunaan tipe data yang benar, programer bisa menjalankan operator aritmatika, operator relasional, dan operator logika pada bahasa pemograman, termasuk Java.