

Nama & NPM	Topik:	Tanggal:
Hamzah Rizqullah Rahmad G1F024067	Tipe Data	11 September 2024

[No. 1] Identifikasi Masalah:

- 1.1. Rekomendasikan perbaikan kode agar program Contoh 1 dapat berjalan!
- 1.2. Tambahkan baris untuk menampilkan perhitungan dengan operator (-, *, /, %) pada Contoh 1!

[No.1] Analisis dan Argumentasi

- 1) Saya mengusulkan permasalahan ini dapat diatasi dengan cara menambahkan tanda plus (+) pada baris `System.out.println("a + b = " (a + b));` menjadi `System.out.println("a + b = " + (a + b));` , lalu memberikan perhitungan operator (-, *, /, %)

[No.1] Penyusunan Algoritma dan Kode Program

- 1) Algoritma
Algoritma adalah langkah-langkah penyelesaian masalah.

a. Deklarasi variabel:

- Buat dua variabel integer a dan b untuk menyimpan nilai-nilai yang akan digunakan dalam operasi aritmatika.

b. Inisialisasi variabel:

- Tetapkan nilai awal untuk variabel a dan b. Dalam contoh ini, a diinisialisasi dengan 20 dan b dengan 3.

c. Penampilkan nilai variabel:

- Cetak nilai variabel a dan b ke layar untuk informasi.

d. Perhitungan dan tampilan hasil:

- Lakukan operasi aritmatika berikut dan cetak hasilnya:
 - Penjumlahan: $a + b$
 - Pengurangan: $a - b$
 - Perkalian: $a * b$
 - Pembagian: a / b (hasil bilangan bulat)
 - Modulus: $a \% b$ (sisanya)


Kode program dan luaran

```
public class OperatorAritmatika {  
    public static void main(String[] args) {  
        // Deklarasi nilai  
        int a = 20, b = 3;  
  
        // Operator aritmatika  
        System.out.println("a: " + a);  
        System.out.println("b: " + b);  
        System.out.println("a + b = " + (a + b)); // Penjumlahan  
        System.out.println("a - b = " + (a - b)); // Pengurangan  
        System.out.println("a * b = " + (a * b)); // Perkalian  
        System.out.println("a / b = " + (a / b)); // Pembagian (hasil bilangan bulat)  
        System.out.println("a % b = " + (a % b)); // Modulus (sisanya bagi)  
    }  
}
```

Gambar 1 (input)

Output Generated Files

```
a: 20  
b: 3  
a + b = 23  
a - b = 17  
a * b = 60  
a / b = 6  
a % b = 2
```

 CPU Time: 0.05 sec(s) | Memory: 38772 kilobyte(s) | Compiled and executed in 1.477 sec(s)

Gambar 2 (luaran/output)

a) Screenshot/ Capture potongan kode dan hasil luaran

Analisa luaran yang dihasilkan

a: 20 - Menampilkan nilai awal dari variabel a.

b: 3 - Menampilkan nilai awal dari variabel b.

a + b = 23 - Menampilkan hasil penjumlahan antara a dan b.

a - b = 17 - Menampilkan hasil pengurangan antara a dan b.

a * b = 60 - Menampilkan hasil perkalian antara a dan b.

a / b = 6 - Menampilkan hasil pembagian antara a dan b. Karena kedua variabel bertipe int, hasil pembagiannya juga akan dibulatkan menjadi bilangan bulat terdekat.

a % b = 2 - Menampilkan sisa pembagian antara a dan b.

[No.1] Kesimpulan

Analisa

a) Susunlah kesimpulan berdasarkan permasalahan, algoritma, dan kode program! Kode program di atas bertujuan untuk mendemonstrasikan penggunaan operator aritmatika dasar dalam bahasa pemrograman Java. Operator-operator ini digunakan untuk melakukan operasi matematika sederhana seperti penjumlahan, pengurangan, perkalian, pembagian, dan modulus pada dua buah bilangan bulat.

[No. 2] Identifikasi Masalah:

- 2.1. Tambahkan baris Contoh 2 untuk menampilkan perhitungan dengan operator (-=, *=, /=, %=)!
- 2.2. Berikan argumentasi tentang perbedaan luaran dan waktu eksekusi Contoh 1 dan Contoh 2!

[No.2] Analisis dan Argumentasi

- 2) Saya mengusulkan permasalahan ini dapat diatasi dengan cara menambahkan 2 baris contoh perhitungan menggunakan operator
- 2.2 argumentasi untuk soal 2.2:
- contoh 1: Menampilkan nilai awal a dan b, kemudian hasil dari setiap operasi penugasan yang dilakukan pada b menggunakan operator +=, -=, *= dan sebagainya.
 - contoh 2: Menampilkan nilai awal a dan b, kemudian hasil dari operasi aritmatika dasar antara a dan b menggunakan operator +, -, *, /, dan %.

Meskipun kedua kode menghasilkan output yang terkait dengan operasi aritmatika, luarannya berbeda dalam cara penyajian. Kode 1 fokus pada perubahan nilai b setelah setiap operasi penugasan, sedangkan Kode 2 menampilkan hasil langsung dari setiap operasi aritmatika.

Waktu Eksekusi:

Secara umum, waktu eksekusi kedua kode ini dapat dianggap hampir sama. Kedua kode melibatkan operasi aritmatika dasar yang sederhana, dan perbedaan dalam cara penyajian hasil tidak akan berdampak signifikan pada waktu eksekusi.

[No.2] Penyusunan Algoritma dan Kode Program

2) Algoritma

Algoritma adalah langkah-langkah penyelesaian masalah.

A.Deklarasi Variabel:

Deklarasikan dua variabel integer a dan b untuk menyimpan nilai-nilai yang akan digunakan dalam operasi penugasan.

Inisialisasi variabel a dengan nilai 20 dan variabel b dengan nilai 3.

B. Output Nilai Awal:

Cetak nilai awal dari variabel a dan b ke konsol.

C. Operasi Penugasan:

Penjumlahan:

Tambahkan nilai a ke variabel b menggunakan operator +=.

Cetak nilai baru b setelah penjumlahan.

Pengurangan:

Kurangi nilai a dari variabel b menggunakan operator -=.

Cetak nilai baru b setelah pengurangan.

Perkalian:

Kalikan nilai a dengan variabel b menggunakan operator *=.

Cetak nilai baru b setelah perkalian.

Pembagian:

Bagi nilai b dengan nilai a menggunakan operator /= (hasilnya akan dibulatkan ke bawah).

Cetak nilai baru b setelah pembagian.

Modulus:

Hitung sisa bagi dari pembagian b dengan a menggunakan operator %=.

Cetak nilai baru b setelah modulus.

D. Selesai:


Program selesai setelah semua operasi penugasan dilakukan dan hasilnya ditampilkan.

Kode program dan luaran

```
1 public class OperatorPenugasan {
2     public static void main(String[] args) {
3         // | nilai
4         int a = 20, b = 3;
5
6         // Operator
7         System.out.println("Nilai awal a: " + a);
8         System.out.println("Nilai awal b: " + b);
9
10        // Penjumlahan
11        b += a;
12        System.out.println("Penambahan: b = b + a, b = " + b);
13
14        // Pengurangan
15        b -= a;
16        System.out.println("Pengurangan: b = b - a, b = " + b);
17
18        // Perkalian
19        b *= a;
20        System.out.println("Perkalian: b = b * a, b = " + b);
21
22        // Pembagian
23        b /= a;
24        System.out.println("Pembagian: b = b / a, b = " + b);
25
26        // Modulus
27        b %= a;
28        System.out.println("Modulus: b = b % a, b = " + b);
29    }
30 }
```

Gambar 2 (input)

```
Nilai awal a: 20
Nilai awal b: 3
Penambahan: b = b + a, b = 23
Pengurangan: b = b - a, b = 3
Perkalian: b = b * a, b = 60
Pembagian: b = b / a, b = 3
Modulus: b = b % a, b = 3
```

 CPU Time: 0.09 sec(s) | Memory: 38664 kilobyte(s) | Compiled and executed in 1.366 sec(s)

Gambar 2 (Output)

a) Screenshot/ Capture potongan kode dan hasil luaran

b) Analisa luaran yang dihasilkan

Nilai Awal:

Kode ini mencetak nilai awal dari variabel a dan b yang telah dideklarasikan dan diinisialisasi.

Operasi Penugasan:

- Penjumlahan: Nilai a (20) ditambahkan ke nilai b (3), menghasilkan 23. Nilai b kemudian diperbarui menjadi 23.
- Pengurangan: Nilai a (20) dikurangi dari nilai b (23), menghasilkan 3. Nilai b diperbarui menjadi 3.
- Perkalian: Nilai a (20) dikalikan dengan nilai b (3), menghasilkan 60. Nilai b diperbarui menjadi 60.
- Pembagian: Nilai b (60) dibagi dengan nilai a (20), menghasilkan 3 (hasil bagi bulat). Nilai b diperbarui menjadi 3.
- Modulus: Sisa bagi dari pembagian b (3) dengan a (20) adalah 0. Nilai b diperbarui menjadi 0.

c) [No.2] Kesimpulan

Analisa

- b) Susunlah kesimpulan berdasarkan permasalahan, algoritma, dan kode program!

Kode ini memberikan contoh sederhana namun efektif tentang bagaimana operator penugasan digunakan dalam Java. Operator-operator ini sangat berguna untuk memperpendek penulisan kode dan membuat program lebih efisien. Dengan memahami konsep operator penugasan, kita dapat melakukan berbagai macam manipulasi data dalam program Java.

[No. 3] Identifikasi Masalah:

Ubahlah nilai A = 4 dan B = 4 pada Contoh 3. Simpulkan perubahan yang terjadi!

[No.3] Analisis dan Argumentasi

- 3) Saya mengusulkan permasalahan ini dapat diatasi dengan cara mengubah nilai A dan B menjadi 4 dan menyimpulkan perubahan nya

3.1 Kesimpulan perubahan :

Luaran:

- Kode 1: Menampilkan nilai awal a dan b, kemudian hasil dari setiap operasi penugasan yang dilakukan pada b menggunakan operator +=, -=, *= dan sebagainya.
- Kode 2: Menampilkan nilai awal a dan b, kemudian hasil dari operasi aritmatika dasar antara a dan b menggunakan operator +, -, *, /, dan %.

Meskipun kedua kode menghasilkan output yang terkait dengan operasi aritmatika, luarannya berbeda dalam cara penyajian. Kode 1 fokus pada perubahan nilai b setelah setiap operasi penugasan, sedangkan Kode 2 menampilkan hasil langsung dari setiap operasi aritmatika.

Waktu Eksekusi:

Secara umum, waktu eksekusi kedua kode ini dapat dianggap hampir sama. Kedua kode melibatkan operasi aritmatika dasar yang sederhana, dan perbedaan dalam cara penyajian hasil tidak akan berdampak signifikan pada waktu eksekusi.

[No.3] Penyusunan Algoritma dan Kode Program

A. Algoritma

Algoritma adalah langkah-langkah penyelesaian masalah.

▣ Inisialisasi Variabel:

- Deklarasikan tiga variabel:
 - nilaiA bertipe integer untuk menyimpan nilai pertama.
 - nilaiB bertipe integer untuk menyimpan nilai kedua.
 - hasil bertipe boolean untuk menyimpan hasil perbandingan.
- Beri nilai awal kepada nilaiA dan nilaiB. Dalam contoh ini, keduanya diberikan nilai 4.

Tampilkan Nilai Awal:

- Cetak nilai nilaiA dan nilaiB ke layar untuk menunjukkan nilai awal yang digunakan dalam perbandingan.

Perbandingan Lebih Besar:

- Bandingkan nilaiA dengan nilaiB menggunakan operator >.
- Simpan hasil perbandingan (true atau false) ke dalam variabel hasil.
- Cetak hasil perbandingan ke layar.

Perbandingan Lebih Kecil:

- Bandingkan nilaiA dengan nilaiB menggunakan operator <.
- Simpan hasil perbandingan ke dalam variabel hasil.
- Cetak hasil perbandingan ke layar.

Perbandingan Lebih Besar Sama Dengan:

- Bandingkan nilaiA dengan nilaiB menggunakan operator >=.
- Simpan hasil perbandingan ke dalam variabel hasil.
- Cetak hasil perbandingan ke layar.

Perbandingan Lebih Kecil Sama Dengan:

- Bandingkan nilaiA dengan nilaiB menggunakan operator <=.
- Simpan hasil perbandingan ke dalam variabel hasil.
- Cetak hasil perbandingan ke layar.

Perbandingan Sama Dengan:

- Bandingkan nilaiA dengan nilaiB menggunakan operator ==.
- Simpan hasil perbandingan ke dalam variabel hasil.
- Cetak hasil perbandingan ke layar.

Perbandingan Tidak Sama Dengan:

- Bandingkan nilaiA dengan nilaiB menggunakan operator !=.
- Simpan hasil perbandingan ke dalam variabel hasil.
- Cetak hasil perbandingan ke layar.

Kode program dan luaran

```
public class OperatorKelasional {
    public static void main(String[] args) {
        int nilaiA = 4;
        int nilaiB = 4;
        boolean hasil;

        System.out.println(" A = " + nilaiA + "\n B = " + nilaiB);
        // apakah A lebih besar dari B?
        hasil = nilaiA > nilaiB;
        System.out.println("\n Hasil A > B = " + hasil);

        // apakah A lebih kecil dari B?
        hasil = nilaiA < nilaiB;
        System.out.println("\n Hasil A < B = " + hasil);

        // apakah A lebih besar samadengan B?
        hasil = nilaiA >= nilaiB;
        System.out.println("\n Hasil A >= B = " + hasil);

        // apakah A lebih kecil samadengan B?
        hasil = nilaiA <= nilaiB;
        System.out.println("\n Hasil A <= B = " + hasil);

        // apakah nilai A sama dengan B?
        hasil = nilaiA == nilaiB;
        System.out.println("\n Hasil A == B = " + hasil);

        // apakah nilai A tidak samadengan B?
        hasil = nilaiA != nilaiB;
        System.out.println("\n Hasil A != B = " + hasil);
    }
}
```

Gambar 3 (input)

```
A = 4
B = 4

Hasil A > B = false

Hasil A < B = false

Hasil A >= B = true

Hasil A <= B = true

Hasil A == B = true

Hasil A != B = false
```

Gambar 3 (output)

d) Screenshot/ Capture potongan kode dan hasil luaran

Analisa luaran yang dihasilkan

$A > B$ dan $A < B$: Karena nilai A dan B sama, maka A tidak lebih besar dan tidak lebih kecil dari B. Oleh karena itu, hasil dari kedua perbandingan ini adalah false.

$A \geq B$ dan $A \leq B$: Karena A sama dengan B, maka A baik lebih besar sama dengan maupun lebih kecil sama dengan B. Jadi, hasil dari kedua perbandingan ini adalah true.

$A == B$: Karena nilai A dan B sama persis, maka perbandingan kesamaan ini menghasilkan true.

$A != B$: Karena nilai A dan B sama, maka perbandingan tidak sama dengan ini menghasilkan false.

[No.3] Kesimpulan

Analisa

c) Susunlah kesimpulan berdasarkan permasalahan, algoritma, dan kode program!

Memahami operator relasional dan bagaimana mereka bekerja adalah dasar penting dalam pemrograman. Dengan memahami konsep ini, Anda dapat membuat program yang lebih kompleks dan cerdas, yang mampu mengambil keputusan berdasarkan kondisi tertentu.

[No. 4] Identifikasi Masalah:

- 4.1. Berikan saran operasi apa yang diperlukan (pre/post increment, pre/post decrement) agar Contoh 4 menghasilkan nilai a = 5 dan b = 6?
- 4.2. Simpulkan hasil eksperimen Anda!

[No.4] Analisis dan Argumentasi

- 4) Untuk mendapatkan nilai b sebesar 6, Saya mengusulkan permasalahan ini dapat diatasi dengan cara, kita perlu menggunakan *pre-increment* pada variabel a

4.2 kesimpulan eksperimen:

Perbedaan mendasar antara *post-increment* (a++) dan *pre-increment* (++a) terletak pada urutan operasi terhadap suatu variabel. Pada *post-increment*, nilai variabel akan digunakan terlebih dahulu dalam ekspresi, baru kemudian nilainya dinaikkan satu. Sebaliknya, *pre-increment* akan menaikkan nilai variabel satu tingkat terlebih dahulu sebelum digunakan dalam ekspresi. Hal ini secara langsung memengaruhi hasil akhir dari suatu operasi, terutama ketika nilai variabel tersebut digunakan dalam perhitungan atau perbandingan.

[No.4] Penyusunan Algoritma dan Kode Program

B. Algoritma

Algoritma adalah langkah-langkah penyelesaian masalah.

Deklarasi Variabel:

- Deklarasikan variabel integer a untuk menyimpan nilai yang akan digunakan.
- Inisialisasi variabel a dengan nilai 5.

Output Nilai Awal:

- Cetak nilai awal dari variabel a ke konsol.

Operasi Pre-Increment:

- Lakukan operasi pre-increment pada variabel a menggunakan ++a.
- Nilai a akan dinaikkan satu unit sebelum digunakan dalam ekspresi.

Output Nilai Setelah Pre-Increment:

- Cetak nilai a yang telah dinaikkan ke konsol.

Kode program dan luaran

```
1 public class operator {
2     public static void main(String[] args) {
3         int a = 5;
4
5         System.out.println("a: " + a);
6         System.out.println("b: " + (++a));
7     }
8 }
9
```

Gambar 4 (input)

```
a: 5
b: 6
```

Gambar 4 (output/luaran)

- Screenshot/ Capture potongan kode dan hasil luaran

- Analisa luaran yang dihasilkan

Deklarasi: Variabel a dideklarasikan dengan nilai awal 5.

Output Pertama: Nilai a dicetak, yaitu 5.

Output Kedua:

`++a` adalah *pre-increment*. Ini berarti nilai a akan dinaikkan terlebih dahulu sebelum digunakan dalam ekspresi.

Karena a sudah dinaikkan menjadi 6 sebelum digunakan dalam ekspresi, maka nilai yang dicetak untuk b adalah 6.

[No.4] Kesimpulan

Analisa

d) Susunlah kesimpulan berdasarkan permasalahan, algoritma, dan kode program! Pemahaman yang baik tentang *post-increment* dan *pre-increment* sangat penting dalam pemrograman. Dengan memahami urutan operasi, kita dapat menulis kode yang lebih efisien dan menghindari kesalahan logika. Penggunaan *post-increment* atau *pre-increment* tergantung pada konteks masalah dan hasil yang diinginkan. Dalam beberapa kasus, perbedaan antara keduanya mungkin terlihat sepele, namun dalam program yang lebih kompleks, pemilihan operator yang tepat dapat sangat berpengaruh pada kinerja dan hasil akhir.

[No. 5] Identifikasi Masalah:

5.1. Rekomendasikan berapa nilai a dan b apabila ingin menghasilkan luaran *true* dengan operator **&&** dan operator **| |** ?

5.2. Berikan kesimpulan dari latihan 5.1

[No.5] Analisis dan Argumentasi

5.1 Saya mengusulkan permasalahan ini dapat diatasi dengan cara membuat kode dengan operator **&&** dan operator **| |** yang bertujuan agar nilai a dan b bernilai true

5.2 Operator AND (**&&**) memerlukan kedua operand (a dan b) bernilai true untuk menghasilkan luaran true. Jadi, jika ingin hasil true dengan operator **&&**, nilai a dan b harus true.

Operator OR (**| |**) memerlukan setidaknya satu operand yang bernilai true untuk menghasilkan luaran true. Jadi, jika ingin hasil true dengan operator **| |**, cukup salah satu dari a atau b yang bernilai true. Namun, agar konsisten dalam skenario ini, kedua variabel a dan b juga dapat diatur ke true untuk memastikan hasilnya true pada kedua operator.

Dengan demikian, solusi paling efektif adalah menetapkan nilai a dan b sama-sama true agar kedua operasi logika (**&&** dan **| |**) menghasilkan output true.

[No.5] Penyusunan Algoritma dan Kode Program

C. Algoritma

Algoritma adalah langkah-langkah penyelesaian masalah.

- I. **Mulai** program.
- II. **Deklarasikan** dua variabel boolean, a dan b, dengan nilai true.
- III. **Hitung** hasil operasi logika AND (**&&**) antara variabel a dan b.
-Jika kedua nilai a dan b adalah true, hasil akan true, jika tidak maka false.
- IV. **Tampilkan** hasil operasi logika a **&&** b.
- V. **Deklarasikan** dua variabel boolean baru, A dengan nilai true dan B dengan nilai false.
- VI. **Hitung** hasil operasi logika OR (**| |**) antara variabel A dan B.
-Jika salah satu dari A atau B adalah true, hasil akan true, jika tidak maka false.
- VII. **Tampilkan** hasil operasi logika A **| |** B.
- VIII. **Selesai**.

Kode program dan luaran

```
1 public class Operator {
2     public static void main(String[] args) {
3         // Bagian pertama (logika a && b)
4         boolean a = true;
5         boolean b = true;
6         System.out.println("Hasil logika (a && b) : " + (a && b));
7
8         // Bagian kedua (logika a || b)
9         boolean A = true; // Variabel baru untuk menghindari konflik dengan a dan b sebelumnya
10        boolean B = false;
11        System.out.println("Hasil logika (a || b) : " + (A || B));
12    }
13 }
14 }
```

Gambar 5 (input)

```
Hasil logika (a && b) : true
Hasil logika (a || b) : true
```

Gambar 5 (output)

- a) Screenshot/ Capture potongan kode dan hasil luaran
Beri komentar pada kode yang di Screenshot
- b) Analisa luaran yang dihasilkan
Bagian Pertama: Operasi AND (&&)
- c) boolean a = true;
- d) boolean b = true;
- e) System.out.println("Hasil logika (a && b) : " + (a && b));
- f) Pada bagian ini, dua variabel boolean a dan b diinisialisasi dengan nilai true.
- g) Operasi logika && (AND) hanya akan menghasilkan true jika kedua operand adalah true. Karena a dan b keduanya true, hasilnya akan true.
- h) **Bagian Kedua: Operasi OR (||)**
- i) boolean A = true;
- j) boolean B = false;
- k) System.out.println("Hasil logika (A || B) : " + (A || B));
- l) Pada bagian ini, dua variabel boolean A dan B diinisialisasi dengan nilai true dan false.
- m) Operasi logika || (OR) akan menghasilkan true jika salah satu atau kedua operand adalah true. Karena A adalah true meskipun B adalah false, hasil dari operasi ini adalah true.

[No.5] Kesimpulan

Analisa

e) Susunlah kesimpulan berdasarkan permasalahan, algoritma, dan kode program! Untuk memastikan bahwa kedua operasi logika, baik **AND (&&)** maupun **OR (||)**, menghasilkan luaran **true**, solusi yang paling konsisten adalah menetapkan kedua variabel a dan b bernilai **true**. Operator **AND** membutuhkan kedua operand bernilai **true** agar hasilnya **true**, sedangkan operator **OR** hanya memerlukan salah satu operand bernilai **true** untuk mendapatkan hasil **true**. Oleh karena itu, menetapkan kedua variabel a dan b bernilai **true** adalah pendekatan yang paling efektif dan sederhana untuk memastikan bahwa kedua operasi tersebut menghasilkan luaran **true**.

[No. 6] Identifikasi Masalah:

6.1 Rekomendasikan apa bentuk tanda operator agar nilai = 60 memenuhi untuk Lulus !

[No.6] Analisis dan Argumentasi

- 5) Saya mengusulkan permasalahan ini dapat diatasi dengan cara menambah tanda sama dengan pada baris status = (nilai > 60)? "Lulus": "Gagal"; menjadi status = (nilai >= 60) ? "Lulus" : "Gagal";

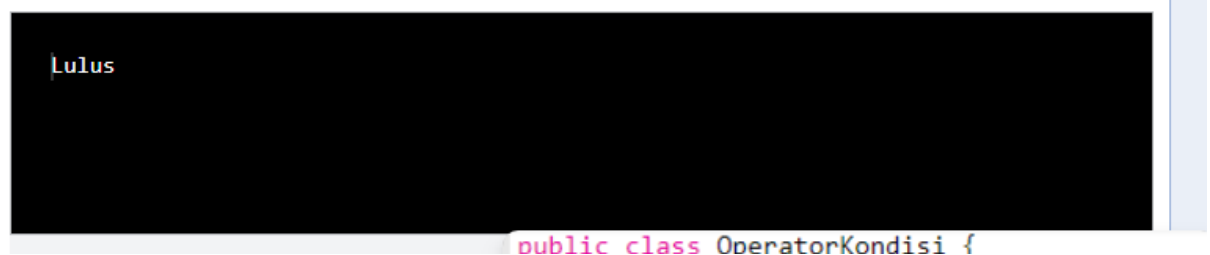
[No.6] Penyusunan Algoritma dan Kode Program

- D. Algoritma
- Algoritma adalah langkah-langkah penyelesaian masalah.
Mulai program.
 - Deklarasikan** variabel status sebagai string kosong.
 - Deklarasikan** variabel nilai dengan nilai awal 60.
 - Periksa kondisi** apakah nilai lebih besar atau sama dengan 60:
Jika benar (nilai >= 60), set status menjadi "Lulus".
Jika salah, set status menjadi "Gagal".
 - Tampilkan** hasil variabel status.
 - Selesai.**

Kode program dan luaran

```
public class OperatorKondisi {  
    public static void main( String[] args ){  
        String status = "";  
        int nilai = 60;  
        // Menggunakan >= agar nilai 60 memenuhi syarat untuk "Lulus"  
        status = (nilai >= 60) ? "Lulus" : "Gagal";  
        System.out.println( status );  
    }  
}
```

Gambar 6 (input)



```
Lulus  
  
public class OperatorKondisi {
```

Gambar 6 (output)

n) Screenshot/ Capture potongan kode dan hasil luaran

o) Analisa luaran yang dihasilkan

- Deklarasi dan Inisialisasi Variabel:**
 - String status = ""; : Variabel status dideklarasikan sebagai string kosong.
- int nilai = 60; : Variabel nilai dideklarasikan dan diinisialisasi dengan nilai 60.
- Operator Ternary:**
 - status = (nilai >= 60) ? "Lulus" : "Gagal";
 - Ini adalah operator ternary yang bekerja seperti pernyataan if-else.

- Kondisi yang diuji adalah (nilai ≥ 60). Jika kondisi ini benar (true), maka nilai status akan menjadi "Lulus". Jika kondisi ini salah (false), maka nilai status akan menjadi "Gagal".
- Dalam hal ini, karena nilai 60 sama dengan 60, kondisi nilai ≥ 60 adalah true. Oleh karena itu, status akan diatur menjadi "Lulus".
- **Output:**
- `System.out.println(status);` : Baris ini akan mencetak nilai dari variabel status ke layar.
- Mengingat hasil dari operator ternary adalah "Lulus", maka output yang dihasilkan adalah:
- Lulus

[No.6] Kesimpulan

Analisa

f) Susunlah kesimpulan berdasarkan permasalahan, algoritma, dan kode program! Dengan menggunakan operator ternary `status = (nilai ≥ 60) ? "Lulus" : "Gagal";`, kita memastikan bahwa nilai 60 memenuhi syarat untuk dinyatakan "Lulus". Algoritma dan kode program yang disusun dengan benar memberikan hasil yang diharapkan. **Operator \geq** adalah bentuk yang tepat untuk memastikan nilai 60 memenuhi syarat untuk mendapatkan hasil "Lulus". Output yang dihasilkan sesuai dengan ekspektasi dan menunjukkan bahwa nilai 60 berhasil memenuhi kriteria "Lulus".

[No. 7] Identifikasi Masalah:

7.1 Evaluasi penyebab hasil $\sim a = -11$? Buktikan jawaban Anda dalam perhitungan biner!

[No.7] Analisis dan Argumentasi

- 6) Saya mengusulkan permasalahan ini dapat diatasi dengan menganalisis kode java tersebut lalu menyimpulkan hasilnya, dan membuktikan dalam perhitungan biner

[No.7] Penyusunan Algoritma dan Kode Program

E. Algoritma

Mulai Program

Deklarasi Variabel

- Deklarasikan variabel a dengan nilai 10.
- Deklarasikan variabel b dengan nilai 7.
- Deklarasikan variabel hasil untuk menyimpan hasil operasi.

Operasi Bitwise AND (&)

- Hitung hasil dari a & b.
 - Representasi biner dari a adalah 0000 0000 0000 0000 0000 0000 0000 1010.
 - Representasi biner dari b adalah 0000 0000 0000 0000 0000 0000 0000 0111.
 - Operasi AND: 0000 0000 0000 0000 0000 0000 0000 1010 & 0000 0000 0000 0000 0000 0000 0000 0111 menghasilkan 0000 0000 0000 0000 0000 0000 0000 0010.
- Simpan hasil operasi ke dalam variabel hasil.
- Tampilkan hasil: "Hasil dari a & b : 2".

Operasi Bitwise OR (|)

- Hitung hasil dari a | b.
 - Operasi OR: 0000 0000 0000 0000 0000 0000 0000 1010 | 0000 0000 0000 0000 0000 0000 0000 0111 menghasilkan 0000 0000 0000 0000 0000 0000 0000 1111.
- Simpan hasil operasi ke dalam variabel hasil.
- Tampilkan hasil: "Hasil dari a | b : 15".

Operasi Bitwise XOR (^)

- Hitung hasil dari a ^ b.
 - Operasi XOR: 0000 0000 0000 0000 0000 0000 0000 1010 ^ 0000 0000 0000 0000 0000 0000 0000 0111 menghasilkan 0000 0000 0000 0000 0000 0000 0000 1101.
- Simpan hasil operasi ke dalam variabel hasil.
- Tampilkan hasil: "Hasil dari a ^ b : 13".

Operasi Bitwise NOT (~)

- Hitung hasil dari $\sim a$.
 - Representasi biner dari a adalah 0000 0000 0000 0000 0000 0000 0000 1010.
 - Operasi NOT: $\sim 0000 0000 0000 0000 0000 0000 0000 1010$ menghasilkan 1111 1111 1111 1111 1111 1111 1111 0101.
- Simpan hasil operasi ke dalam variabel hasil.
- Tampilkan hasil: "Hasil dari $\sim a$: -11".

Operasi Right Shift (>>)

- Hitung hasil dari a >> 1.
 - Representasi biner dari a adalah 0000 0000 0000 0000 0000 0000 0000 1010.
 - Right Shift: 0000 0000 0000 0000 0000 0000 0000 1010 >> 1 menghasilkan 0000 0000 0000 0000 0000 0000 0000 0101.
- Simpan hasil operasi ke dalam variabel hasil.
- Tampilkan hasil: "Hasil dari a >> 1 : 5".

Operasi Left Shift (<<)

- Hitung hasil dari b << 2.
 - Representasi biner dari b adalah 0000 0000 0000 0000 0000 0000 0000 0111.
 - Left Shift: 0000 0000 0000 0000 0000 0000 0000 0111 << 2 menghasilkan 0000 0000 0000 0000 0000 0000 0001 1100.
- Simpan hasil operasi ke dalam variabel hasil.

- Tampilkan hasil: "Hasil dari b << 2 : 28".
Selesai

Kode program dan luaran

```
public class OperatorBitwise {
    public static void main(String[] args) {
        int a = 10; // Representasi biner: 0000 0000 0000 0000 0000 0000 1010
        int b = 7;  // Representasi biner: 0000 0000 0000 0000 0000 0000 0111
        int hasil;

        // Operasi AND (&)
        hasil = a & b; // 0000 0000 0000 0000 0000 0000 0010
        // AND
        // 0000 0000 0000 0000 0000 0000 0111
        // -----
        // 0000 0000 0000 0000 0000 0000 0010
        System.out.println("Hasil dari a & b : " + hasil); // Output: 2

        // Operasi OR (|)
        hasil = a | b; // 0000 0000 0000 0000 0000 0000 1010
        // OR
        // 0000 0000 0000 0000 0000 0000 0111
        // -----
        // 0000 0000 0000 0000 0000 0000 1111
        System.out.println("Hasil dari a | b : " + hasil); // Output: 15

        // Operasi XOR (^)
        hasil = a ^ b; // 0000 0000 0000 0000 0000 0000 1010
        // XOR
        // 0000 0000 0000 0000 0000 0000 0111
        // -----
        // 0000 0000 0000 0000 0000 0000 1101
        System.out.println("Hasil dari a ^ b : " + hasil); // Output: 13

        // Operasi NOT (~)
        hasil = ~a; // 0000 0000 0000 0000 0000 0000 1010
        // NOT
        // 1111 1111 1111 1111 1111 1111 0101
        // Hasil ~a adalah -11 dalam desimal
        System.out.println("Hasil dari ~a : " + hasil); // Output: -11

        // Operasi Right Shift (>>)
        hasil = a >> 1; // 0000 0000 0000 0000 0000 0000 1010
        // Right Shift by 1
        // 0000 0000 0000 0000 0000 0000 0101
        System.out.println("Hasil dari a >> 1 : " + hasil); // Output: 5

        // Operasi Left Shift (<<)
        hasil = b << 2; // 0000 0000 0000 0000 0000 0000 0111
        // Left Shift by 2
        // 0000 0000 0000 0000 0000 0000 1100
        System.out.println("Hasil dari b << 2 : " + hasil); // Output: 28
    }
}
```

Gambar 7 (input)



Gambar 7 (output)

p) Screenshot/ Capture potongan kode dan hasil luaran

- Analisa luaran yang dihasilkan
Operasi AND (&) membandingkan bit demi bit dari a dan b. Hanya bit yang sama-sama 1 menghasilkan 1 pada posisi tersebut. Dengan a = 10 (0000 1010 dalam biner) dan b = 7 (0000 0111 dalam biner), hasil operasi AND adalah 2 (0000 0010 dalam biner).

- Operasi OR (|) membandingkan bit demi bit dari a dan b. Bit yang salah satu atau keduanya 1 menghasilkan 1 pada posisi tersebut. Dengan a = 10 dan b = 7, hasil operasi OR adalah 15 (0000 1111 dalam biner).
- Operasi XOR (^) membandingkan bit demi bit dari a dan b. Bit yang berbeda menghasilkan 1 pada posisi tersebut. Dengan a = 10 dan b = 7, hasil operasi XOR adalah 13 (0000 1101 dalam biner).
- Operasi NOT (~) membalikkan setiap bit dari a. Dengan a = 10 (0000 1010 dalam biner), hasil operasi NOT adalah -11, yang merupakan bentuk dua's complement dari hasil biner 1111 0101.
- Operasi Right Shift (>>) menggeser bit ke kanan, menghilangkan bit paling kanan dan menambahkan bit 0 di sebelah kiri. Dengan a = 10 (0000 1010 dalam biner), hasil operasi Right Shift satu bit adalah 5 (0000 0101 dalam biner).
- Operasi Left Shift (<<) menggeser bit ke kiri, menambahkan bit 0 di sebelah kanan, dan membuang bit paling kiri. Dengan b = 7 (0000 0111 dalam biner), hasil operasi Left Shift dua bit adalah 28 (0001 1100 dalam biner).

[No.7] Kesimpulan

Analisa

g) Susunlah kesimpulan berdasarkan permasalahan, algoritma, dan kode program!
 Dalam analisa operasi bitwise pada kode Java yang diberikan, setiap operasi memberikan hasil yang sesuai dengan ekspektasi berdasarkan perhitungan biner.

1. **Operasi AND (&)** pada variabel a (10) dan b (7) menghasilkan 2. Ini karena hanya bit-bit yang sama-sama 1 dari kedua variabel yang menghasilkan 1 pada posisi tersebut, yaitu 0000 0010 dalam biner.
2. **Operasi OR (|)** pada variabel a dan b menghasilkan 15. Operasi ini menggabungkan semua bit yang memiliki nilai 1 dari salah satu atau kedua variabel, menghasilkan 0000 1111 dalam biner.
3. **Operasi XOR (^)** pada variabel a dan b menghasilkan 13. XOR menghasilkan 1 hanya jika bit-bit pada posisi tersebut berbeda antara kedua variabel, menghasilkan 0000 1101 dalam biner.
4. **Operasi NOT (~)** membalikkan setiap bit dari variabel a. Dengan a yang memiliki nilai 10 (0000 1010 dalam biner), operasi NOT menghasilkan 1111 0101 dalam biner, yang dikonversi menjadi -11 dalam desimal. Ini adalah bentuk dua's complement dari -11.
5. **Operasi Right Shift (>>)** pada variabel a dengan nilai 10 menghasilkan 5. Penggeseran bit ke kanan oleh satu posisi membuang bit paling kanan dan menambahkan bit 0 di sebelah kiri, menghasilkan 0000 0101 dalam biner.
6. **Operasi Left Shift (<<)** pada variabel b dengan nilai 7 menghasilkan 28. Penggeseran bit ke kiri oleh dua posisi menambahkan dua bit 0 di sebelah kanan, menghasilkan 0001 1100 dalam biner.

Analisa ini membuktikan bahwa hasil dari setiap operasi bitwise sesuai dengan perhitungan biner yang dilakukan, menunjukkan bagaimana operasi bitwise memanipulasi bit pada level yang sangat mendetail.

Refleksi

Setelah mempelajari dan membuat laporan praktikum tentang operator bitwise dalam Java, saya merasa sangat terbantu untuk memahami konsep dasar pengolahan data pada tingkat bit. Operator bitwise seperti AND (&), OR (|), XOR (^), dan NOT (~) memberikan kontrol yang mendalam dan spesifik dalam manipulasi bit, yang sangat penting dalam berbagai aplikasi, mulai dari pengolahan citra hingga algoritma kriptografi.

Dari praktikum ini, saya menyadari betapa kuatnya operator bitwise dalam pengolahan data, khususnya dalam optimisasi performa. Misalnya, operasi bitwise sering digunakan dalam pengaturan bit flags dan konfigurasi sistem, di mana kecepatan dan efisiensi adalah kunci.

Beberapa poin penting yang saya pelajari termasuk:

1. **Kepentingan Representasi Biner:** Memahami bagaimana angka desimal diterjemahkan ke dalam representasi biner dan bagaimana operasi bitwise bekerja pada tingkat ini sangat penting. Ini membantu dalam memahami bagaimana komputer memproses dan menyimpan data.
2. **Operator Bitwise dan Aplikasinya:** Penggunaan operator bitwise dapat sangat efisien dalam banyak situasi, terutama dalam aplikasi yang memerlukan manipulasi bit yang presisi, seperti sistem embedded dan aplikasi performa tinggi.
3. **Kesalahan Umum dan Perhitungan Negatif:** Mengetahui bagaimana operasi NOT menghasilkan angka negatif dengan menggunakan representasi dua's complement memperjelas bagaimana sistem komputer menyimpan angka negatif.
4. **Pentingnya Praktikum:** Praktikum ini tidak hanya mengasah keterampilan teknis saya dalam menggunakan operator bitwise tetapi juga membantu saya memahami konsep-konsep abstrak dengan cara yang lebih konkrit melalui penerapan langsung dalam kode.

Secara keseluruhan, laporan praktikum ini sangat bermanfaat dalam memperdalam pemahaman saya tentang operasi bitwise. Saya merasa lebih siap untuk menerapkan pengetahuan ini dalam proyek-proyek pemrograman yang lebih kompleks dan real-world.