

Template Lembar Kerja Individu

LATIHAN 1

Nama & NPM	Topik:	Tanggal:
Renni Ren Rofa.Pgb G1F021002	Operator Java	18 September 2024
[1.] Identifikasi Masalah:		
<p>1) Uraikan permasalahan : Dari soal latihan 1 memerintahkan ,</p> <p>1.1 Tambahkan baris <code>System.out.println("a + b = " + (a + b));</code> Ubahlah operator (+) dengan tanda (-, *, /, %)</p> <p>1.2 Analisa perhitungan matematika yang terjadi!</p> <p>Variable : Variable yang terlibat adalah a = 20 dan b = 3.</p> <p>2) Rincikan sumber informasi yang relevan (buku / webpage): Dokumentasi resmi Oracle tentang <u>Java Operators</u>.</p> <p>3) Parameter solusi (jika ada): Parameter solusi: Dua variabel integer (a dan b) dengan nilai tetap</p>		
[1.] Analisis dan Argumentasi		
<p>1) Uraikan rancangan solusi yang diusulkan. Rancangan solusi melibatkan deklarasi dua variabel (a dan b) dan kemudian menerapkan operator aritmatika untuk masing-masing operasi: penjumlahan, pengurangan, perkalian, pembagian, dan modulus. Setiap hasil akan dicetak ke layar.</p> <p>2) Analisis solusi, kaitkan dengan permasalahan. Solusi ini langsung menyelesaikan permasalahan yang dihadapi, yaitu menghitung hasil dari setiap operasi aritmatika dasar. Dengan menggunakan operator yang sesuai untuk setiap perhitungan, kita dapat melihat bagaimana berbagai operasi mempengaruhi dua bilangan yang diberikan.</p>		
[1.] Penyusunan Algoritma , Kode Program dan Analisa perhitungan Matematika		
<p>1) Rancang desain solusi atau algoritma :</p> <ul style="list-style-type: none">• Deklarasikan variabel a dan b dengan nilai masing-masing.• Hitung hasil dari setiap operasi aritmatika (+, -, *, /, %).• Cetak hasil operasi ke layar. <p>2) Tuliskan kode program dan luaran</p> <pre>public class OperatorAritmatika{ public static void main(String[] args) { // deklarasi nilai int a = 20, b = 3; // menampilkan nilai variabel a dan b System.out.println("a: " + a); System.out.println("b: " + b); // operasi penjumlahan System.out.println("a + b = " + (a + b)); // output: 23 // operasi pengurangan System.out.println("a - b = " + (a - b)); // output: 17</pre>		

```

// operasi perkalian
System.out.println("a * b = " + (a * b)); // output: 60

// operasi pembagian (bilangan bulat)
System.out.println("a / b = " + (a / b)); // output: 6

// operasi modulus (sisanya)
System.out.println("a % b = " + (a % b)); // output: 2
}
}

```

a) Uraikan luaran yang dihasilkan:

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

b) Screenshot/ Capture potongan kode dan hasil luaran

```

15 public class OperatorAritmatika{
16     public static void main(String[] args) {
17         // deklarasi nilai
18         int a = 20, b = 3;
19
20         // menampilkan nilai variabel a dan b
21         System.out.println("a: " + a);
22         System.out.println("b: " + b);
23
24         // operasi penjumlahan
25         System.out.println("a + b = " + (a + b)); // output: 23
26
27         // operasi pengurangan
28         System.out.println("a - b = " + (a - b)); // output: 17
29
30         // operasi perkalian
31         System.out.println("a * b = " + (a * b)); // output: 60
32
33         // operasi pembagian (bilangan bulat)
34         System.out.println("a / b = " + (a / b)); // output: 6
35
36         // operasi modulus (sisanya)
37         System.out.println("a % b = " + (a % b)); // output: 2
38     }
39 }

```

Gambar 1.1 Sourcecode Operator Aritmatika

Output	Generated Files
<pre> a: 20 b: 3 a + b = 23 a - b = 17 a * b = 60 a / b = 6 a % b = 2 </pre>	

Gambar 1.2.Luaran

3) Analisa Perhitungan Matematika Yang terjadi :

- **Penjumlahan** (a + b): 20 + 3 = 23
Hasil penjumlahan kedua bilangan menghasilkan 23.
- **Pengurangan** (a - b): 20 - 3 = 17
Pengurangan antara a dan b menghasilkan 17.
- **Perkalian** (a * b): 20 * 3 = 60
Perkalian kedua bilangan menghasilkan 60.
- **Pembagian** (a / b): 20 / 3 = 6
Pembagian bilangan bulat di Java memberikan hasil bilangan bulat, sehingga hasilnya adalah 6, karena 20 / 3 = 6 (sisa dibulatkan).
- **Modulus** (a % b): 20 % 3 = 2
Modulus memberikan sisa dari pembagian, sehingga hasilnya adalah 2 karena 20 dibagi 3 menyisakan 2.

[1.] Kesimpulan

1) Analisa

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

Berdasarkan permasalahan yang diajukan, program berhasil menghitung dan menampilkan hasil dari lima operasi aritmatika dasar antara dua bilangan. Algoritma ini sederhana dan mudah dipahami, mengaplikasikan operator aritmatika dengan benar untuk setiap operasi

b) Apakah dasar alasan pengambilan keputusan Anda untuk kasus ini?

Keputusan menggunakan operator aritmatika dasar diambil karena sesuai dengan permasalahan yang diajukan. Java menyediakan berbagai operator untuk mengerjakan operasi aritmatika, sehingga implementasi ini merupakan solusi yang logis dan praktis.

2) Evaluasi

a) Apa konsekuensi dari skenario pemrograman ini?

Skenario pemrograman ini menunjukkan bagaimana bilangan bulat ditangani dalam operasi aritmatika di Java. Khususnya pada operasi pembagian, hasil yang diperoleh adalah bilangan bulat tanpa desimal, dan modulus menunjukkan sisa pembagian

3) Kreasi

a) Apakah ada pengetahuan baru yang dikembangkan dan konsep baru sebagai usulan solusi?

Pengetahuan baru yang dapat dikembangkan adalah penanganan operasi aritmatika dengan tipe data lain (misalnya, bilangan desimal menggunakan double), serta bagaimana menangani pembagian dengan hasil decimal.

b) Konstruksikan hubungan antara variabel yang berbeda dengan konsep yang anda ketahui! (jika ada)

Dalam kasus ini, variabel a dan b mempengaruhi hasil dari setiap operasi aritmatika. Hubungan antara variabel ini mengikuti aturan dasar matematika, di mana hasil perhitungan tergantung pada nilai yang diberikan ke dalam variabel tersebut.

LATIHAN 2

[No. 2] Identifikasi Masalah:

1) Uraikan permasalahan :

2.1. Bandingkan hasil Contoh 1 dengan Contoh 2!

Masalah utama adalah bagaimana penggunaan operator aritmatika dan operator penugasan dalam bahasa pemrograman Java mempengaruhi hasil operasi aritmatika. Variabel yang digunakan adalah a dan b, di mana a = 20 dan b = 3.

2) Sumber informasi:

Webpage seperti dokumentasi resmi *Oracle Java* dan tutorial pemrograman Java online seperti *GeeksForGeeks*, *W3Schools*, dan *StackOverflow*.

[No.2] Analisis dan Argumentasi

1) Uraikan Rancangan Solusi dan argumentasi:

Buat dua program Java, satu menggunakan operator aritmatika dan satu menggunakan operator penugasan. Setiap program akan menggunakan variabel a dan b, dan melakukan operasi yang sama dengan hasil yang berbeda karena perbedaan operator

2) Analisis solusi :

Alasan solusi ini karena adanya perbedaan utama antara Contoh 1 dan Contoh 2 yang terletak pada jenis operator yang digunakan serta luaran yang dihasilkan

[No.2] Penyusunan Algoritma , Kode Program dan Perbandingan Hasil

1) Algoritma:

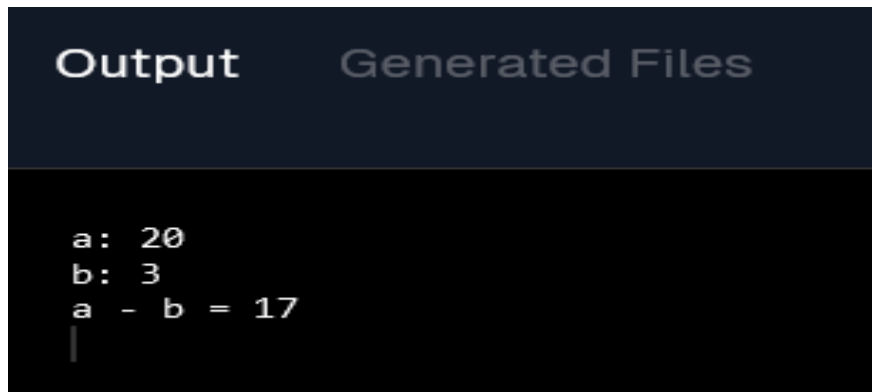
- Deklarasi variabel a dan b
- Lakukan operasi aritmatika pada variabel.
- Lakukan operasi penugasan dengan operator penugasan.
- Cetak hasil operasi ke layar.

2) Kode program dan luaran

a) Screenshot/ Capture potongan kode dan hasil luaran

```
14
15 public class OperatorAritmatika {
16     public static void main(String[] args) {
17         // Deklarasi nilai
18         int a = 20, b = 3;
19
20         // Operator aritmatika
21         System.out.println("a: " + a);
22         System.out.println("b: " + b);
23         System.out.println("a - b = " + (a - b)); // Pengurangan
24     }
25 }
26
```

Gambar 2.1 Kode program Operator Aritmatika



Gambar 2.2 Luaran operator Aritmatika

```
public class OperatorPenugasan {
    public static void main(String[] args) {
        // Deklarasi nilai
        int a = 20, b = 3;

        // Operator penugasan
        b += a; // Penambahan
        System.out.println("Penambahan : " + b);

        b -= a; // Pengurangan
        System.out.println("Pengurangan : " + b);

        b *= a; // Perkalian
        System.out.println("Perkalian : " + b);

        b /= a; // Pembagian
        System.out.println("Pembagian : " + b);

        b %= a; // Sisa bagi
        System.out.println("Sisa Bagi: " + b);
    }
}
```

Gambar 2.3 Kode program operator Penugasan



Gambar 2. 4 Luaran operator penugasan

b) Analisa luaran dan perbandingan hasil.

- **Program 1(aritmatika):** Menghasilkan operasi pengurangan saja, tanpa mempengaruhi nilai variabel a dan b.
- **Program 2(penugasan):** Menghasilkan beberapa operasi berturut-turut di mana setiap operasi mengubah nilai variabel b dan mencetak hasilnya.

[No.2] Kesimpulan

- 1) Kreasi, Pengetahuan baru yang dikembangkan:

Memahami kapan harus menggunakan operator aritmatika dan kapan operator penugasan lebih efisien dapat membantu dalam membuat program yang lebih efektif dan efisien.

[No. 3] Identifikasi Masalah:

1) Uraikan permasalahan dan variable:

- 3.1 Ubahlah nilai $A = 4$ dan $B = 4$. Analisa perubahan yang terjadi!
3.2 Bandingkan bagaimana perbedaan nilai A dan B mempengaruhi nilai luaran!

Variabel:

- Nilai A: Variabel integer yang menyimpan nilai pertama.
- Nilai B: Variabel integer yang menyimpan nilai kedua.
- Hasil : Variabel boolean yang menyimpan hasil perbandingan

2) Sumber informasi

Webpage seperti dokumentasi resmi *Oracle Java* dan tutorial pemrograman Java online seperti *GeeksForGeeks*, *W3Schools*, dan *StackOverflow*.

[No.3] Analisis dan Argumentasi

1) Uraikan Rancangan Solusi yang diusulkan :

- Program menggunakan operator relasional untuk membandingkan dua variabel integer (nilaiA dan nilaiB). Dua kasus akan diuji:
 - nilaiA = 12 dan nilaiB = 4.
 - nilaiA = 4 dan nilaiB = 4.
- Setiap operator ($>$, $<$, $>=$, $<=$, $==$, $!=$) akan dijalankan dan hasilnya dicetak ke layar.

2) Alasan solusi ini karena membantu memahami bagaimana operator relasional mempengaruhi hasil berdasarkan perbandingan antara dua nilai. Jika nilai berbeda ($A > B$), operator akan menghasilkan hasil yang berbeda dibandingkan ketika kedua nilai sama ($A == B$). Ini penting untuk logika keputusan dalam pemrograman.

[No.3] Penyusunan Algoritma, Kode Program dan perbedaan nilai A dan B mempengaruhi nilai Luaran.

1) Algoritma:

- Deklarasi variabel Nilai A dan nilai B
- Gunakan operator relasional untuk membandingkan kedua variabel.
- Cetak hasil perbandingan dari setiap operator ke layar.

2) Kode program dan luaran

a) Screenshot/ Capture potongan kode dan hasil luaran


```

15 public class OperatorRelasional {
16     public static void main(String[] args) {
17         int nilaiA = 4;
18         int nilaiB = 4;
19         boolean hasil;
20
21         System.out.println("A = " + nilaiA + "\nB = " + nilaiB);
22
23         // apakah A lebih besar dari B?
24         hasil = nilaiA > nilaiB;
25         System.out.println("Hasil A > B = " + hasil); // false
26
27         // apakah A lebih kecil dari B?
28         hasil = nilaiA < nilaiB;
29         System.out.println("Hasil A < B = " + hasil); // false
30
31         // apakah A lebih besar samadengan B?
32         hasil = nilaiA >= nilaiB;
33         System.out.println("Hasil A >= B = " + hasil); // true
34
35         // apakah A lebih kecil samadengan B?
36         hasil = nilaiA <= nilaiB;
37         System.out.println("Hasil A <= B = " + hasil); // true
38
39         // apakah nilai A sama dengan B?
40         hasil = nilaiA == nilaiB;
41         System.out.println("Hasil A == B = " + hasil); // true
42
43         // apakah nilai A tidak samadengan B?
44         hasil = nilaiA != nilaiB;
45         System.out.println("Hasil A != B = " + hasil); // false
46     }
47 }
48

```

Gambar 3.1 Kode program operator rasional

```

Output      Generated Files

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

Compiled and executed in 2.393 sec(s)

```

Gambar 3.2 Luaran program

Penjelasan :

Dari gambar 3.1 dan 3.2 diatas bisa di **analisa perubahan yang terjadi :**

1. Karena A dan B memiliki nilai yang sama (4), hasil dari operator relasional akan berbeda dibandingkan saat nilai A dan B berbeda.
2. Operator > (lebih besar) dan < (lebih kecil) keduanya mengembalikan false karena tidak ada variabel yang lebih besar atau lebih kecil.
3. Operator >= (lebih besar sama dengan) dan <= (lebih kecil sama dengan) keduanya mengembalikan true karena A sama dengan B.
4. Operator == mengembalikan true karena nilai A dan B sama.
5. Operator != mengembalikan false karena A dan B memiliki nilai yang sama, jadi mereka tidak berbeda.

b) Perbedaan nilai A dan B mempengaruhi nilai luaran!

Perbedaan nilai A dan B secara langsung mempengaruhi hasil dari operator relasional. Jika nilai A dan B berbeda, hasil operator yang membandingkan ukuran (seperti > dan <) akan berbeda.

akan lebih bervariasi. Sebaliknya, jika nilai A dan B sama, hasil operator relasional yang berkaitan dengan persamaan akan lebih dominan (seperti == dan != telah sesuai dengan kebutuhan dan permintaan data.

[No.3) Kesimpulan

1) Analisa

Operator relasional memudahkan pemrogram untuk membandingkan dua variabel. Ketika variabel memiliki nilai berbeda, hasil perbandingan akan lebih bervariasi. Jika variabel memiliki nilai yang sama, hasil dari operator == dan != akan menjadi lebih penting.

LATIHAN 4

[No. 4] Identifikasi Masalah:

1) Uraikan permasalahan dan variabel:

4.1 Berdasarkan luaran program Contoh 4, bandingkan hasil Post dan Pre untuk Increment dan Decrement!

Variabel :

Variabel yang digunakan meliputi:

- a dan b untuk increment.
- c dan d untuk decrement.

2) Sumber Informasi :

Website: Official Java Documentation, Oracle (<https://docs.oracle.com/javase/tutorial/java/nutsandbolts/operators.html>) untuk referensi lebih lanjut mengenai penggunaan operator increment dan decrement.

[No.4] Analisis dan Argumentasi

- 1) Saya mengusulkan permasalahan ini dapat diatasi dengan cara menulis program yang secara eksplisit menunjukkan perbedaan output ketika menggunakan Post dan Pre Increment/Decrement. Nilai variabel dicetak sebelum dan sesudah penggunaan operator.
- 2) Analisis Solusi, Permasalahan tentang bagaimana Post dan Pre Increment/Decrement bekerja dijelaskan melalui contoh program. Dengan melihat hasil output dari kedua jenis operator, dapat disimpulkan bahwa Post Increment/Decrement menggunakan nilai variabel sebelum operator diterapkan, sedangkan Pre Increment/Decrement menerapkan operator terlebih dahulu sebelum menggunakan nilai variabel.

[No.4] Penyusunan Algoritma dan Kode Program

1) Algoritma

2) Kode program dan luaran

```
public class Operator {
    public static void main(String[] args) {
        int a = 10;
        System.out.println("# Post Increment #");
        System.out.println("=====");
        // Mencetak nilai a sebelum dan sesudah Post Increment
        System.out.println("Isi variabel a: " + a);
        System.out.println("Isi variabel a: " + a++);
        System.out.println("Isi variabel a: " + a);

        System.out.println();

        int b = 10;
        System.out.println("# Pre Increment #");
        System.out.println("=====");
        // Mencetak nilai b sebelum dan sesudah Pre Increment
        System.out.println("Isi variabel b: " + b);
```

```

System.out.println("Isi variabel b: " + ++b);
System.out.println("Isi variabel b: " + b);

System.out.println();

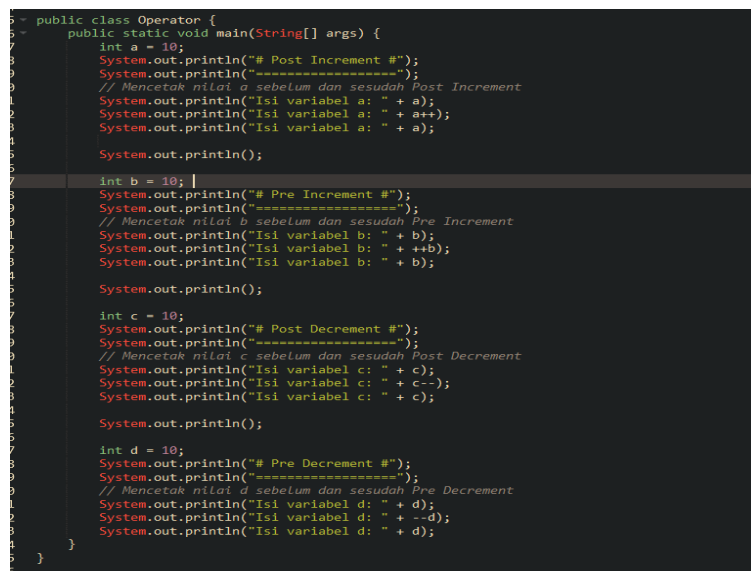
int c = 10;
System.out.println("# Post Decrement #");
System.out.println("=====");
// Mencetak nilai c sebelum dan sesudah Post Decrement
System.out.println("Isi variabel c: " + c);
System.out.println("Isi variabel c: " + c--);
System.out.println("Isi variabel c: " + c);

System.out.println();

int d = 10;
System.out.println("# Pre Decrement #");
System.out.println("=====");
// Mencetak nilai d sebelum dan sesudah Pre Decrement
System.out.println("Isi variabel d: " + d);
System.out.println("Isi variabel d: " + --d);
System.out.println("Isi variabel d: " + d);
}
}

```

3) Screenshot/ Capture potongan kode dan hasil luaran

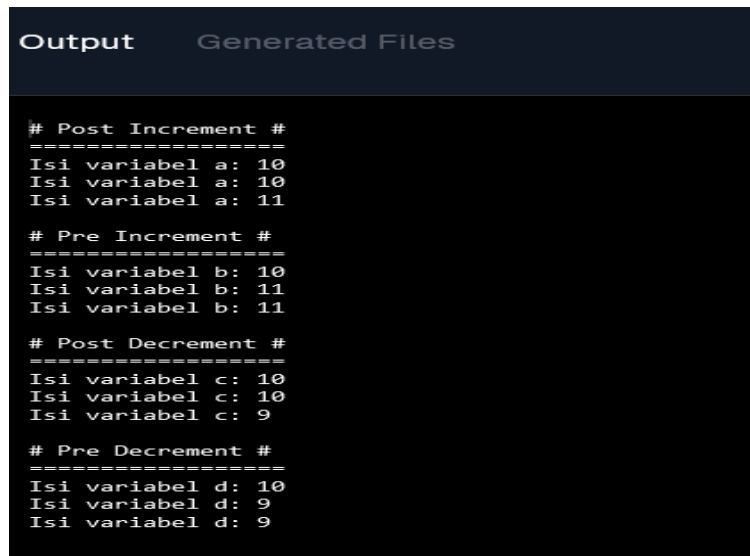


```

1 public class Operator {
2     public static void main(String[] args) {
3         int a = 10;
4         System.out.println("# Post Increment #");
5         System.out.println("=====");
6         // Mencetak nilai a sebelum dan sesudah Post Increment
7         System.out.println("Isi variabel a: " + a);
8         System.out.println("Isi variabel a: " + ++a);
9         System.out.println("Isi variabel a: " + a);
10        System.out.println();
11
12        int b = 10;
13        System.out.println("# Pre Increment #");
14        System.out.println("=====");
15        // Mencetak nilai b sebelum dan sesudah Pre Increment
16        System.out.println("Isi variabel b: " + b);
17        System.out.println("Isi variabel b: " + ++b);
18        System.out.println("Isi variabel b: " + b);
19        System.out.println();
20
21        int c = 10;
22        System.out.println("# Post Decrement #");
23        System.out.println("=====");
24        // Mencetak nilai c sebelum dan sesudah Post Decrement
25        System.out.println("Isi variabel c: " + c);
26        System.out.println("Isi variabel c: " + c--);
27        System.out.println("Isi variabel c: " + c);
28        System.out.println();
29
30        int d = 10;
31        System.out.println("# Pre Decrement #");
32        System.out.println("=====");
33        // Mencetak nilai d sebelum dan sesudah Pre Decrement
34        System.out.println("Isi variabel d: " + d);
35        System.out.println("Isi variabel d: " + --d);
36        System.out.println("Isi variabel d: " + d);
37    }
38 }

```

Gambar 1 Kode program post dan pre



```
# Post Increment #
=====
Isi variabel a: 10
Isi variabel a: 10
Isi variabel a: 11

# Pre Increment #
=====
Isi variabel b: 10
Isi variabel b: 11
Isi variabel b: 11

# Post Decrement #
=====
Isi variabel c: 10
Isi variabel c: 10
Isi variabel c: 9

# Pre Decrement #
=====
Isi variabel d: 10
Isi variabel d: 9
Isi variabel d: 9
```

Gambar 2. Luaran program

4) Analisa perbandingan hasil Post dan Pre untuk Increment dan Decrement!

○ Increment:

1. Post Increment (a++):

- Nilai variabel digunakan terlebih dahulu sebelum di-increment.
- Contoh: Pada `System.out.println("Isi variabel a: " + a++)`;; nilai a yang dicetak adalah **10** (nilai awal), kemudian baru variabel a bertambah menjadi **11**.
- Setelah itu, saat dipanggil lagi, `System.out.println("Isi variabel a: " + a)`;; nilainya sudah berubah menjadi **11**.

2. Pre Increment (++b):

- Nilai variabel di-increment terlebih dahulu sebelum digunakan.
- Contoh: Pada `System.out.println("Isi variabel b: " + ++b)`;; nilai b langsung bertambah menjadi **11** sebelum dicetak.
- Saat dipanggil lagi, nilai b tetap **11**.

○ Decrement:

1. Post Decrement (c--):

- Nilai variabel digunakan terlebih dahulu sebelum di-decrement.
- Contoh: Pada `System.out.println("Isi variabel c: " + c--)`;; nilai c yang dicetak adalah **10** (nilai awal), kemudian variabel c berkurang menjadi **9**.
- Saat dipanggil lagi, `System.out.println("Isi variabel c: " + c)`;; nilainya sudah menjadi **9**.

2. Pre Decrement (--d):

- Nilai variabel di-decrement terlebih dahulu sebelum digunakan.
- Contoh: Pada `System.out.println("Isi variabel d: " + --d)`;; nilai d langsung berkurang menjadi **9** sebelum dicetak.
- Saat dipanggil lagi, nilainya tetap **9**.

[No.4] Kesimpulan

1. Analisa :

Kesimpulannya adalah bahwa perbedaan utama antara Post dan Pre Increment/Decrement terletak pada urutan penggunaan nilai variabel. Post Increment/Decrement menggunakan nilai sebelum operator bekerja, sedangkan Pre Increment/Decrement langsung mengubah nilai sebelum digunakan.

LATIHAN 5

[No. 5] Identifikasi Masalah:

1) Uraikan permasalahan dan variable:

- 5.1 Tambahkan baris kode untuk memeriksa `a || b`.
- 5.2 Ubahlah nilai `a = false` dan `b = false`. Analisa perubahan dan perbedaan boolean yang terjadi!
- 5.3 Apabila diketahui pernyataan `a || b && a || !b`. Uraikan urutan logika yang akan dikerjakan! Analisa luaran `true` atau `false` dari pernyataan tersebut!

Permasalahan yang dibahas dalam latihan ini adalah memahami cara kerja operator logika `&&` (AND), `||` (OR), dan `!` (NOT) pada variabel boolean. Variabel yang digunakan dalam program adalah `a` dan `b` yang bertipe boolean.

[No.5] Analisis dan Argumentasi

- 1) Saya mengusulkan permasalahan ini dapat diatasi dengan cara menambahkan baris kode untuk memeriksa `a || b`, kemudian mengubah nilai `a` dan `b` menjadi `false` dan mengamati perubahan hasil evaluasi boolean. Setelah itu, dilakukan analisis untuk pernyataan boolean yang lebih kompleks seperti `a || b && a || !b`.
- 2) Alasan solusi ini karena solusi ini membantu untuk memahami bagaimana Java memprioritaskan operator logika. Operator `&&` memiliki prioritas lebih tinggi daripada `||`, sehingga akan dievaluasi terlebih dahulu. Dengan memeriksa berbagai kombinasi nilai `a` dan `b`, serta menyisipkan operasi `!`, kita dapat melihat bagaimana urutan evaluasi memengaruhi hasil logika.

[No.5] Penyusunan Algoritma dan Kode Program

1) Algoritma

- Definisikan variabel boolean `a` dan `b`.
- Lakukan evaluasi menggunakan operator `&&` dan `||`.
- Ubah nilai `a` dan `b` menjadi `false` dan ulangi evaluasi.
- Tambahkan pernyataan logika kompleks `a || b && a || !b`.

2) Kode program dan luaran :

```
public class OperatorLogika {
    public static void main (String [] args) {
        boolean a = true;
        boolean b = false;
        boolean c;

        // Evaluasi menggunakan && (AND)
        c = a && b;
        System.out.println("true && false = " + c); // Hasilnya adalah false

        // Evaluasi menggunakan || (OR)
        c = a || b;
        System.out.println("true || false = " + c); // Hasilnya adalah true

        // Mengubah nilai a dan b
        a = false;
        b = false;

        // Evaluasi ulang dengan nilai a dan b yang baru
        c = a || b;
```

```

        System.out.println("false || false = " + c); // Hasilnya adalah false

        // Evaluasi pernyataan kompleks a || b && a || !b
        c = a || b && a || !b;
        System.out.println("false || false && false || true = " + c); // Hasilnya adalah true
    }
}

```

3) Screenshot/ Capture potongan kode dan hasil luar

```

15 public class OperatorLogika {
16     public static void main (String [] args) {
17         boolean a = true;
18         boolean b = false;
19         boolean c;
20
21         // Evaluasi menggunakan && (AND)
22         c = a && b;
23         System.out.println("true && false = " + c); // Hasilnya adalah false
24
25         // Evaluasi menggunakan || (OR)
26         c = a || b;
27         System.out.println("true || false = " + c); // Hasilnya adalah true
28
29         // Mengubah nilai a dan b
30         a = false;
31         b = false;
32
33         // Evaluasi ulang dengan nilai a dan b yang baru
34         c = a || b;
35         System.out.println("false || false = " + c); // Hasilnya adalah false
36
37         // Evaluasi pernyataan kompleks a || b && a || !b
38         c = a || b && a || !b;
39         System.out.println("false || false && false || true = " + c); // Hasilnya adalah true
40     }
41 }
42

```

Gambar 5.1 Kode program operator logika

Output Generated Files

```

true && false = false
true || false = true
false || false = false
false || false && false || true = true

```

Gambar 5.2 Luaran

[No.5] Kesimpulan

1. Analisa:

Dari permasalahan, algoritma, dan kode program yang digunakan, dapat disimpulkan bahwa operator && (AND) hanya menghasilkan true jika kedua operand bernilai true, sementara || (OR) menghasilkan true jika salah satu operand bernilai true. Dalam evaluasi logika kompleks, operator && diproses terlebih dahulu karena memiliki prioritas lebih tinggi daripada ||.

LATIHAN 6

[No. 6] Identifikasi Masalah:

1) Uraikan permasalahan dan variable:

Permasalahan utama dalam Contoh 6 adalah penggunaan operator ternary (?) untuk mengevaluasi kondisi apakah nilai seorang siswa lebih besar dari 60. Variabel yang digunakan adalah nilai untuk menyimpan nilai siswa dan status untuk menampilkan hasil evaluasi, yaitu "Lulus" atau "Gagal".

[No.6] Analisis dan Argumentasi

- 1) Saya mengusulkan permasalahan ini dapat diatasi dengan cara mengganti nilai variabel nilai dari 80 menjadi 60 dan mengamati apakah kondisi dalam operator ternary berubah. Operator ternary akan mengevaluasi apakah nilai > 60 menghasilkan true atau false, dan kemudian mengeset status menjadi "Lulus" atau "Gagal."
- 2) Alasan solusi ini karena ketika nilai diubah menjadi 60, kondisi nilai > 60 akan menjadi false, sehingga variabel status akan di-set menjadi "Gagal." Operator ternary efektif untuk situasi di mana hanya ada dua hasil yang mungkin, dan logika ini mempermudah evaluasi singkat.

[No.6] Penyusunan Algoritma dan Kode Program

1) Algoritma :

1. Definisikan variabel nilai dengan nilai 60.
2. Gunakan operator ternary untuk mengevaluasi apakah nilai > 60.
3. Set variabel status berdasarkan hasil dari evaluasi.
4. Cetak hasil status.

2) Kode program dan luaran

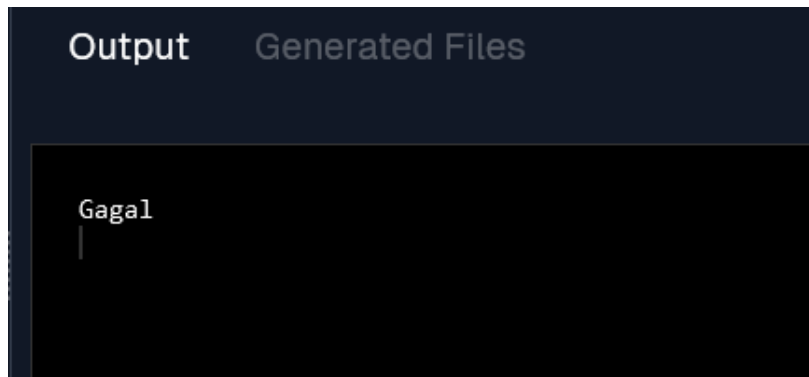
```
public class OperatorKondisi {  
    public static void main( String[] args ){  
        String status = "";  
        int nilai = 60; // Mengubah nilai menjadi 60  
        status = (nilai > 60) ? "Lulus" : "Gagal"; // Evaluasi dengan operator ternary  
        System.out.println(status); // Cetak status  
    }  
}
```

3) Screenshot/ Capture potongan kode dan hasil luaran

A screenshot of a code editor showing the Java code for the OperatorKondisi class. The code is as follows:

```
public class OperatorKondisi {  
    public static void main( String[] args ){  
        String status = "";  
        int nilai = 60; // Mengubah nilai menjadi 60  
        status = (nilai > 60) ? "Lulus" : "Gagal"; // Evaluasi dengan operator ternary  
        System.out.println(status); // Cetak status  
    }  
}
```

Gambar 6.1 Kode program operator kondisi



Gambar 6.2 Luaran program

4) Analisa luaran yang dihasilkan:

- Output: "Gagal"
- Hal ini karena kondisi nilai > 60 tidak terpenuhi (nilai 60 tidak lebih besar dari 60), sehingga operator ternary mengeset status menjadi "Gagal."

[No.6] Kesimpulan

1) Analisa:

Kesimpulan yang dapat diambil adalah bahwa operator ternary di Java bekerja dengan mengevaluasi kondisi logika, dan berdasarkan hasilnya, memberikan salah satu dari dua nilai. Dalam kasus ini, ketika nilai nilai adalah 60, kondisi nilai > 60 tidak terpenuhi, sehingga status diatur menjadi "Gagal."

LATIHAN 7

[No. 7] Identifikasi Masalah:

1) Uraikan permasalahan dan variable:

Permasalahan yang dihadapi adalah bagaimana melakukan operasi bitwise pada dua bilangan integer a dan b dan bagaimana perhitungan tersebut bekerja di tingkat biner.

Variabel yang digunakan:

a: variabel integer dengan nilai 10.

b: variabel integer dengan nilai 7.

hasil: variabel yang menyimpan hasil operasi bitwise

[No.7] Analisis dan Argumentasi

- 2) Saya mengusulkan permasalahan ini dapat diatasi dengan cara menggunakan operator bitwise untuk operasi logika biner langsung pada bilangan. Dalam hal ini, operasi dilakukan pada bilangan a dan b, masing-masing dioperasikan dengan AND, OR, XOR, dan operator shift.
- 3) Alasan solusi ini karena masalah ini berkaitan dengan bagaimana kita bisa memanfaatkan manipulasi bit untuk memahami bagaimana komputer melakukan operasi pada level biner. Dengan melakukan operasi bitwise, kita dapat langsung melihat hasil dari manipulasi bit pada angka, yang berguna untuk berbagai aplikasi seperti pemrograman sistem dan optimasi data.

[No.7] Penyusunan Algoritma dan Kode Program

1) Algoritma :

- Definisikan dua variabel a dan b dengan nilai tertentu.
- Gunakan operator bitwise (&, |, ^, ~, >>, <<) untuk melakukan perhitungan pada kedua variabel.
- hasil dari masing-masing operasi ke konsol.

2) Kode program dan luaran

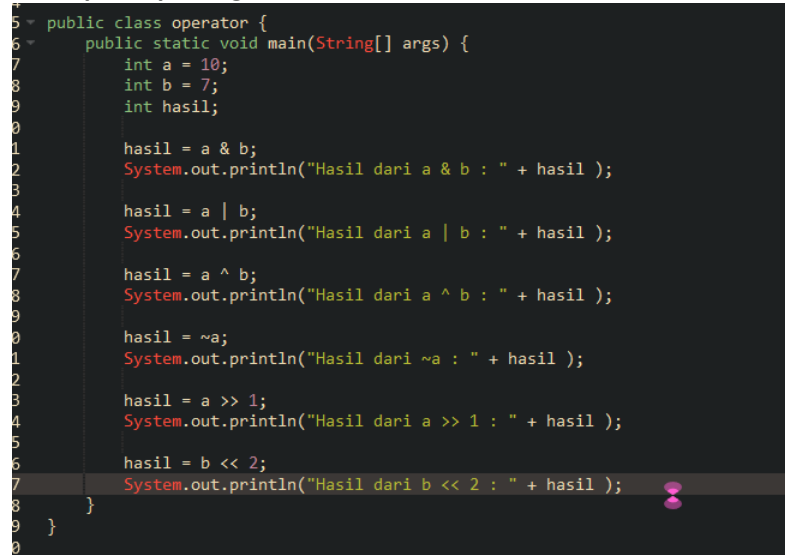
```
public class operator {  
    public static void main(String[] args) {  
        int a = 10;  
        int b = 7;  
        int hasil;  
  
        hasil = a & b;  
        System.out.println("Hasil dari a & b : " + hasil );  
  
        hasil = a | b;  
        System.out.println("Hasil dari a | b : " + hasil );  
  
        hasil = a ^ b;  
        System.out.println("Hasil dari a ^ b : " + hasil );  
  
        hasil = ~a;  
        System.out.println("Hasil dari ~a : " + hasil );  
  
        hasil = a >> 1;  
        System.out.println("Hasil dari a >> 1 : " + hasil );  
    }  
}
```

```

    hasil = b << 2;
    System.out.println("Hasil dari b << 2 : " + hasil);
}
}

```

3) Screenshot/ Capture potongan kode dan hasil luaran



```

5 public class operator {
6     public static void main(String[] args) {
7         int a = 10;
8         int b = 7;
9         int hasil;
10
11         hasil = a & b;
12         System.out.println("Hasil dari a & b : " + hasil );
13
14         hasil = a | b;
15         System.out.println("Hasil dari a | b : " + hasil );
16
17         hasil = a ^ b;
18         System.out.println("Hasil dari a ^ b : " + hasil );
19
20         hasil = ~a;
21         System.out.println("Hasil dari ~a : " + hasil );
22
23         hasil = a >> 1;
24         System.out.println("Hasil dari a >> 1 : " + hasil );
25
26         hasil = b << 2;
27         System.out.println("Hasil dari b << 2 : " + hasil );
28     }
29 }

```

Gambar 7.1 Kode program



Output	Generated Files
Hasil dari a & b : 2	
Hasil dari a b : 15	
Hasil dari a ^ b : 13	
Hasil dari ~a : -11	
Hasil dari a >> 1 : 5	
Hasil dari b << 2 : 28	

Compiled and executed in 1.244 sec(s)

Gambar 7.2 Luaran program operator *Bitwise*

4) Analisa luaran yang dihasilkan :

Perhitungan biner yang dipilih:

1. AND (&):
 - a = 10 (dalam biner: 1010)
 - b = 7 (dalam biner: 0111)
 - Hasil a & b adalah 0010 (2 dalam desimal)
2. OR (|):
 - a = 10 (dalam biner: 1010)
 - b = 7 (dalam biner: 0111)
 - Hasil a | b adalah 1111 (15 dalam desimal)
3. XOR (^):
 - a = 10 (dalam biner: 1010)

- $b = 7$ (dalam biner: 0111)
- Hasil $a \wedge b$ adalah 1101 (13 dalam desimal)

[No.7] Kesimpulan

1) Analisis:

Operator AND hanya menghasilkan 1 jika kedua bit sama-sama 1. OR menghasilkan 1 jika salah satu atau kedua bit bernilai 1. XOR menghasilkan 1 hanya jika salah satu dari kedua bit bernilai 1, tetapi tidak keduanya.