

### Template Lembar Kerja Individu dan Kelompok

Nama & NPM	Topik:	Tanggal:
Yeni kusherawati G1F024013	OPERATOR JAVA	7 september 2024
<b>[Nomor 1] Identifikasi Masalah:</b>		
<pre>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));     } }</pre> <p><b>Luaran:</b> a: 20 b: 3 a - b = 17</p> <p>System.out.println("a + b = " + (a - b)); Label yang dicetak adalah "a + b = ", namun operator yang digunakan adalah pengurangan (a - b) bukan penjumlahan (a + b).</p> <p>permasalahan nya yaitu: Label tidak sesuai dengan operasi aritmatika: Label System.out.println("a + b = " ...) menunjukkan bahwa hasil yang akan ditampilkan adalah penjumlahan, tetapi operasi yang dilakukan sebenarnya adalah pengurangan (a - b).</p> <p>Solusi: Gantilah operator aritmatika dari - menjadi +, agar label dan operasi yang dilakukan konsisten. Perbaiki kodenya adalah: System.out.println("a + b = " + (a + b)); Jika memang ingin menampilkan pengurangan, maka label harus disesuaikan agar konsisten dengan operasi pengurangan: System.out.println("a - b = " + (a - b));</p>		
<b>[Nomor 1] Analisis dan Argumentasi</b>		
<b>Latihan 1.</b>		
<p>1.1. Tambahkan baris <code>System.out.println("a + b = " + (a + b));</code> Ubahlah operator ( + ) dengan tanda ( -, *, /, % ) Jawab: Ditambahkan baris <code>System.out.println("a + b = " + (a + b));</code> untuk menampilkan hasil penjumlahan a dan b. Setiap operator pada baris tambahan menguji berbagai operasi aritmatika: penjumlahan (+), pengurangan (-), perkalian (*), pembagian (/), dan modulus (%).</p> <p>1.2. Analisa perhitungan matematika yang terjadi! Jawab: "a + b (Penjumlahan): Hasil dari penjumlahan 20 + 3 adalah 23. a - b (Pengurangan): Hasil dari pengurangan 20 - 3 adalah 17. a * b (Perkalian):</p>		

Hasil dari perkalian  $20 * 3$  adalah 60.

a / b (Pembagian):

Hasil dari pembagian  $20 / 3$  adalah 6. Karena kedua nilai bertipe integer, hasil yang ditampilkan juga dalam bentuk integer, sehingga bagian desimal diabaikan.

a % b (Modulus atau Sisa Bagi):

Hasil dari  $20 \% 3$  adalah 2. Ini adalah sisa dari pembagian  $20 / 3$  (karena  $20 = 6 * 3 + 2$ )."

#### [Nomor 1] Penyusunan Algoritma dan Kode Program

##### 1) Algoritma

- a. Mulai.
- b. Deklarasikan dua variabel a dan b dengan nilai 20 dan 3 masing-masing.
- c. Cetak nilai variabel a dan b.
- d. Lakukan operasi pengurangan a - b dan cetak hasilnya.
- e. Selesai.

##### 2) Kode program dan luaran

```
1 public class OperatorAritmatika {
2     public static void main(String[] args) {
3         // deklarasi nilai
4         int a = 20, b = 3;
5         // operator aritmatika
6         System.out.println("a: " + a);
7         System.out.println("b: " + b);
8         //menampilkan hasil dari penjumlahan
9         System.out.println("a - b = " + (a - b)); //pengurangan
10        //menampilkan hasil dari pembagian
11        System.out.println("a + b = " + (a + b)); //penjumlahan
12        //menampilkan hasil dari pengurangan
13        System.out.println("a * b = " + (a * b)); //perkalian
14        //menampilkan hasil dari pembagian
15        System.out.println("a / b = " + (a / b)); //pembagian
16        //menampilkan hasil dari sisa bagi
17        System.out.println("a % b = " + (a % b)); //sisa bagi
18    }
19 }
20
```

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

```
=== Code Execution Successful ===
```

Luaran sudah sesuai dengan program yang disusun.

#### **[Nomor Soal 1] Kesimpulan**

1) Analisa

Penjumlahan (+): Menambahkan dua bilangan untuk mendapatkan jumlah total. Hasilnya adalah bilangan yang lebih besar dari kedua operand.

Pengurangan (-): Mengurangi bilangan kedua dari bilangan pertama untuk mendapatkan selisih. Hasilnya adalah selisih antara operand.

Perkalian (\*): Mengalikan dua bilangan untuk mendapatkan hasil perkalian. Hasilnya adalah bilangan yang merupakan hasil penggandaan kedua operand.

Pembagian (/): Membagi bilangan pertama dengan bilangan kedua. Dalam tipe data integer, hasilnya adalah bilangan bulat, mengabaikan desimal.

<b>Nama &amp; NPM Template Lembar Kerja Individu dan Kelompok</b>	<b>Topik:</b>	<b>Tanggal:</b>
<b>Yeni kuserawati G1F024013</b>	<b>OPERATOR JAVA</b>	<b>7 september 2024</b>
<b>[Nomor 2] Identifikasi Masalah:</b>		
<pre> public class OperatorPenugasan {     public static void main(String[] args) {         // deklarasi nilai         int a = 20, b = 3;         //operator penugasan         b += a;         System.out.println("Penambahan : " + b);          // pengurangan         b -= a;         System.out.println("Pengurangan : " + b);          // perkalian         b *= a;         System.out.println("Perkalian : " + b);          // Pembagian         b /= a;         System.out.println("Pembagian : " + b);          // Sisa bagi         b %= a;         // sekarang b=0         System.out.println("Sisa Bagi: " + b);     } } </pre> <p><b>Luaran:</b></p> <pre> Penambahan : 23 Pengurangan : 3 Perkalian : 60 Pembagian : 3 Sisa Bagi: 3 </pre> <p>Berdasarkan kode dan hasil yang diharapkan, tidak ada masalah dalam program ini. Kode tersebut dengan benar menunjukkan penggunaan operator penugasan dan hasilnya sesuai dengan apa yang diharapkan. Jika mendapatkan output yang berbeda, pastikan semua langkah di atas sudah diperiksa dengan teliti.</p> <p><b>Jika Output Tidak Sesuai:</b></p> <ol style="list-style-type: none"> <li>1. Verifikasi Nilai Awal: Pastikan nilai awal variabel *a* dan *b* sesuai dengan yang harapkan (a = 20 dan b = 3)</li> <li>2. Periksa Penggunaan Operator: Pastikan menggunakan operator penugasan yang benar seperti +=, -=, *=, /=, dan %=.</li> <li>3. Periksa Output Program: Pastikan output yang dicetak sesuai dengan hasil dari operasi yang</li> </ol>		

dilakukan.

4. Cek Lingkungan Pengembangan: Kadang-kadang, masalah bisa disebabkan oleh lingkungan pengembangan atau kesalahan lain di luar kode. Pastikan menjalankan kode di lingkungan yang bersih dan tidak ada kesalahan kompilasi.

## [Nomor 2] Analisis dan Argumentasi

### Latihan 2.

2.1. Bandingkan hasil Contoh 1 dengan Contoh 2!

Jawab:

1. Penambahan:

- Contoh 1:  $(a + b = 20 + 3 = 23)$

- Contoh 2:  $(b += a)$  mengubah b dari 3 menjadi 23, hasilnya sama dengan Contoh 1.

2. Pengurangan:

- Contoh 1:  $(a - b = 20 - 3 = 17)$

- Contoh 2: Setelah penambahan, b adalah 23.  $(b -= a)$  mengubah b menjadi 3, hasilnya sama dengan nilai awal b setelah penambahan.

3. Perkalian:

- Contoh 1:  $(a * b = 20 * 3 = 60)$

- Contoh 2: Setelah pengurangan, b adalah 3.  $(b *= a)$  mengubah b menjadi 60, hasilnya sama dengan Contoh 1.

4. Pembagian:

- Contoh 1:  $(a / b = 20 / 3 = 6)$  (integer division)

- Contoh 2: Setelah perkalian, b adalah 60.  $(b /= a)$  mengubah b menjadi 3, hasilnya sama dengan nilai awal b setelah perkalian.

5. Modulus:

- Contoh 1:  $(a \% b = 20 \% 3 = 2)$

- Contoh 2: Setelah pembagian, b adalah 3.  $(b \% = a)$  mengubah b menjadi 3, karena  $(3 \% 20 = 3)$ .

Hasil ini berbeda dengan hasil pada Contoh 1.

Kesimpulan:

- Contoh 1 menggunakan operator aritmatika langsung, sedangkan Contoh 2 menggunakan operator penugasan yang mengubah nilai variabel secara bertahap.

## [Nomor 2] Penyusunan Algoritma dan Kode Program

### 1. Algoritma

1. Mulai
2. Inisialisasi:
  - $a = 20$
  - $b = 3$
3. Penambahan:
  - $b = b + a$  (hasil  $b = 23$ )
  - Cetak "Penambahan : 23"
4. Pengurangan:
  - $b = b - a$  (hasil  $b = 3$ )
  - Cetak "Pengurangan : 3"
5. Perkalian:
  - $b = b * a$  (hasil  $b = 60$ )
  - Cetak "Perkalian : 60"
6. Pembagian:
  - $b = b / a$  (hasil  $b = 3$ )
  - Cetak "Pembagian : 3"
7. Sisa Bagi:
  - $b = b \% a$  (hasil  $b = 3$ )
  - Cetak "Sisa Bagi: 3"
8. Selesai

### 2. Kode program dan luaran

```
1 public class OperatorPenugasan {
2     public static void main(String[] args) {
3         // deklarasi nilai
4         int a = 20, b = 3;
5         //operator penugasan
6         b += a;
7         System.out.println("Penambahan : " + b);
8         // pengurangan
9         b -= a;
10        System.out.println("Pengurangan : " + b);
11        // perkalian
12        b *= a;
13        System.out.println("Perkalian : " + b);
14        // Pembagian
15        b /= a;
16        System.out.println("Pembagian : " + b);
17        // Sisa bagi
18        b %= a;
19        // sekarang b=0
20        System.out.println("Sisa Bagi: " + b);
21    }
22 }
```

```
Penambahan : 23
Pengurangan : 3
Perkalian : 60
Pembagian : 3
Sisa Bagi: 3
```

```
=== Code Execution Successful ===
```

Luaran sudah sesuai dengan program yang disusun.

<b>[Nomor 21] Kesimpulan</b>
<p>Analisa</p> <p>Kesimpulan:</p> <p>Operator penugasan seperti +=, -=, *=, /=, dan %= digunakan untuk memperbarui nilai variabel dengan hasil dari operasi yang melibatkan variabel tersebut. Operasi ini dilakukan secara berurutan dan, yang berarti nilai variabel akan berubah setelah setiap operasi.</p> <p>Hasil Operasi: Dalam contoh ini, setiap operasi penugasan diperbarui secara langsung pada nilai variabel b dan hasil dari setiap operasi dicetak:</p> <p>Penambahan mengubah b dari 3 menjadi 23.</p> <p>Pengurangan mengubah b dari 23 kembali menjadi 3.</p> <p>Perkalian mengubah b dari 3 menjadi 60.</p> <p>Pembagian mengubah b dari 60 menjadi 3.</p> <p>Modulus mempertahankan b pada nilai 3, karena <math>3 \% 20 = 3</math>.</p> <p>Perubahan Nilai Variabel: Nilai b berubah secara bertahap setelah setiap operasi. Pada akhir program, nilai b adalah 3, hasil dari operasi modulus. Program ini secara efektif menunjukkan bagaimana operator penugasan memodifikasi nilai variabel dan bagaimana hasil akhir dipengaruhi oleh urutan operasi yang dilakukan</p>

<b>Nama &amp; NPM Template Lembar Kerja Individu dan Kelompok</b>	<b>Topik:</b>	<b>Tanggal:</b>
<b>Yeni kusherawati G1F024013</b>	<b>OPERATOR JAVA</b>	<b>7 september 2024</b>

**[Nomor 3] Identifikasi Masalah:**

```
public class OperatorRelasional {
    public static void main(String[] args) {
        int nilaiA = 12;
        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("Hasil A > B = "+ hasil);

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

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

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

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

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

**Luaran:**

A = 12  
B = 4

Hasil A > B = true  
Hasil A < B = false  
Hasil A >= B = true  
Hasil A <= B = false  
Hasil A == B = false  
Hasil A != B = true

Tidak ada masalah pada kode yang diberikan. Perubahan nilai pada nilaiA dan nilaiB mempengaruhi hasil perbandingan seperti yang diharapkan berdasarkan operator relasional. Hasil keluaran berubah sesuai dengan nilai-nilai yang diberikan, yang menunjukkan bahwa operator relasional berfungsi dengan benar dalam perbandingan nilai-nilai integer.



### [Nomor 3] Analisis dan Argumentasi

#### Latihan 3

3.1. Ubahlah nilai  $A = 4$  dan  $B = 4$ . Analisa perubahan yang terjadi!

Jawab: Analisis Perubahan

- $A > B$ : Sekarang false, karena 4 tidak lebih besar dari 4.
- $A < B$ : Sekarang false, karena 4 tidak lebih kecil dari 4.
- $A \geq B$ : Sekarang true, karena 4 lebih besar atau sama dengan 4.
- $A \leq B$ : Sekarang true, karena 4 lebih kecil atau sama dengan 4.
- $A == B$ : Sekarang true, karena 4 sama dengan 4.
- $A != B$ : Sekarang false, karena 4 tidak berbeda dari 4.

Kesimpulan

Perubahan nilai A dan B mempengaruhi hasil perbandingan relasional. Jika nilai A dan B sama, maka:

- Hasil dari perbandingan  $==$  adalah true, dan  $!=$  adalah false.
- Hasil dari perbandingan  $>$  dan  $<$  adalah false, sedangkan  $\geq$  dan  $\leq$  adalah true.

3.2. Bandingkan bagaimana perbedaan nilai A dan B mempengaruhi nilai luaran!

jawab: Dalam kode tersebut, perbedaan antara nilai A dan B mempengaruhi hasil perbandingan yang ditampilkan.

1. nilai  $A >$  nilai  $B$ :

- Jika nilai A lebih besar dari nilai B, hasilnya true.
- Misalnya, dengan nilai  $A = 12$  dan nilai  $B = 4$ , hasilnya adalah true karena 12 lebih besar dari 4.

2. nilai  $A <$  nilai  $B$ :

- Jika nilai A lebih kecil dari nilai B, hasilnya true.
- Dalam contoh ini, karena 12 tidak lebih kecil dari 4, hasilnya adalah false.

3. nilai  $A \geq$  nilai  $B$ :

- Jika nilai A lebih besar atau sama dengan nilai B, hasilnya true.
- Karena 12 lebih besar dari 4, hasilnya adalah true.

4. nilai  $A \leq$  nilai  $B$ :

- Jika nilai A lebih kecil atau sama dengan nilai B, hasilnya true.
- Dalam hal ini, karena 12 tidak lebih kecil dari 4, hasilnya adalah false.

5. nilai  $A ==$  nilai  $B$ :

- Jika nilai A sama dengan nilai B, hasilnya true.
- Dengan nilai yang berbeda (12 dan 4), hasilnya adalah false.

6. nilai  $A !=$  nilai  $B$ :

- Jika nilai A tidak sama dengan nilai B, hasilnya true.
- Karena 12 dan 4 berbeda, hasilnya adalah true.

Jadi, hasil dari perbandingan bergantung pada hubungan antara nilai A dan B dan kondisi perbandingan yang diuji.

### [Nomor 3] Penyusunan Algoritma dan Kode Program

#### 1. Algoritma

- Mulai.
- Tentukan nilai A = 12 dan B = 4.
- Bandingkan A dan B menggunakan operator relasional.
- Cetak hasil perbandingan.
- Selesai.

#### 2. Kode program dan luaran

```
3 int nilaiA = 4;
4 int nilaiB = 4;
5 boolean hasil;
6 System.out.println(" A = " + nilaiA + "\n B = " + nilaiB);
7 // apakah A lebih besar dari B?
8 hasil = nilaiA > nilaiB;
9 System.out.println("Hasil A > B = " + hasil);
10 // apakah A lebih kecil dari B?
11 hasil = nilaiA < nilaiB;
12 System.out.println("Hasil A < B = " + hasil);
13 // apakah A lebih besar samadengan B?
14 hasil = nilaiA >= nilaiB;
15 System.out.println("Hasil A >= B = " + hasil);
16 // apakah A lebih kecil samadengan B?
17 hasil = nilaiA <= nilaiB;
18 System.out.println("Hasil A <= B = " + hasil);
19 // apakah nilai A sama dengan B?
20 hasil = nilaiA == nilaiB;
21 System.out.println("Hasil A == B = " + hasil);
22 // apakah nilai A tidak samadengan B?
23 hasil = nilaiA != nilaiB;
24 System.out.println("Hasil A != B = " + hasil);
25
```

```
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

=== Code Execution Successful ===
```

Luaran sudah sesuai dengan program yang disusun.

### [Nomor 3] Kesimpulan

#### Analisa

##### Analisis Keseluruhan:

Hasil dari perbandingan ini mencerminkan penggunaan dasar dari operator relasional dalam pemrograman. Setiap operator memiliki tujuan spesifik dalam membandingkan dua nilai, dan hasilnya bergantung sepenuhnya pada hubungan antara nilai tersebut. Ketika nilaiA lebih besar dari nilaiB, semua operator yang melibatkan kondisi "lebih besar" akan menghasilkan true, sementara operator yang membutuhkan nilai lebih kecil akan menghasilkan false. Program ini sederhana, namun sangat mendasar dalam menjelaskan bagaimana operasi logika dapat dilakukan dalam pemrograman. Ini membantu dalam pengambilan keputusan berdasarkan kondisi tertentu, yang merupakan fondasi dari banyak algoritma kompleks. Misalnya, dalam kasus di mana nilai berubah-ubah, perbandingan semacam ini dapat digunakan

untuk menentukan alur program yang berbeda, seperti dalam struktur kontrol if-else. Kesimpulannya, perbedaan antara dua nilai mempengaruhi hasil setiap operator relasional, dan program ini menunjukkan bagaimana perbandingan logis dapat memvalidasi hubungan numerik antara dua angka. Pemahaman mendalam tentang operator relasional sangat penting untuk menyusun logika dalam pemrograman, terutama dalam situasi yang membutuhkan keputusan berdasarkan nilai variabel yang berubah-ubah.

<b>Nama &amp; NPM Template Lembar Kerja Individu dan Kelompok</b>	<b>Topik:</b>	<b>Tanggal:</b>
<b>Yeni kusherawati G1F024013</b>	<b>OPERATOR JAVA</b>	<b>7 september 2024</b>

#### **[Nomor 4] Identifikasi Masalah:**

```
public class operator {
    public static void main(String[] args) {
        int a = 10;
        System.out.println("# Post Increment #");
        System.out.println("=====");
        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("=====");
        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("=====");
        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("=====");
        System.out.println("Isi variabel d: " + d);
        System.out.println("Isi variabel d: " + --d);
        System.out.println("Isi variabel d: " + d);
    }
}
```

#### **Post-Increment (a++):**

Pada `System.out.println("Isi variabel a: " + a++)`;, nilai variabel a ditampilkan terlebih dahulu (yaitu 10), kemudian variabel tersebut ditingkatkan nilainya menjadi 11.

Inilah sebabnya ketika dicetak dua kali, nilai pertama adalah 10 dan pada cetakan kedua setelah increment, nilainya menjadi 11.

#### **Pre-Increment (++b):**

Pada `System.out.println("Isi variabel b: " + ++b)`;, variabel b terlebih dahulu ditambah nilainya menjadi 11 sebelum dicetak, sehingga hasil langsung menampilkan nilai 11.

#### Post-Decrement (c--):

Pada `System.out.println("Isi variabel c: " + c--);`, nilai variabel `c` dicetak terlebih dahulu (yaitu 10), kemudian variabel tersebut dikurangi nilainya menjadi 9. Inilah sebabnya hasil yang ditampilkan adalah 10, lalu pada output berikutnya hasilnya menjadi 9.

#### Pre-Decrement (--d):

Pada `System.out.println("Isi variabel d: " + --d);`, variabel `d` dikurangi nilainya terlebih dahulu menjadi 9, kemudian baru dicetak, sehingga hasilnya adalah 9.

Tidak ada masalah pada kode tersebut, semua berjalan sesuai dengan cara kerja operator increment dan decrement dalam Java.

### [Nomor 4] Analisis dan Argumentasi

#### Latihan 4.

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

Jawab: - Post-Increment (a++):

- Langkah 1: Nilai variabel ditampilkan terlebih dahulu, kemudian variabel tersebut ditingkatkan (increment) setelah proses selesai.

- Contoh:

- Nilai awal variabel `a`: 10.

- Ketika `a++` dieksekusi:

- Ditampilkan: 10.

- Setelah baris kode dieksekusi, nilai `a` menjadi 11.

- Jadi, ketika memanggil `System.out.println(a++)`, nilai yang ditampilkan adalah 10, kemudian `a` ditingkatkan menjadi 11.

- Pre-Increment (++b):

- Langkah 1: Nilai variabel ditingkatkan (increment) terlebih dahulu, kemudian ditampilkan setelah peningkatan.

- Contoh:

- Nilai awal variabel `b`: 10.

- Ketika `++b` dieksekusi:

- Nilai `b` langsung menjadi 11 sebelum ditampilkan.

- Jadi, ketika memanggil `System.out.println(++b)`, nilai yang ditampilkan adalah 11 (karena sudah dinaikkan sebelum ditampilkan).

- Post-Decrement (c--):

- Langkah 1: Nilai variabel ditampilkan terlebih dahulu, kemudian variabel tersebut diturunkan (decrement) setelah proses selesai.

- Contoh:

- Nilai awal variabel `c`: 10.

- Ketika `c--` dieksekusi:

- Ditampilkan: 10.

- Setelah baris kode dieksekusi, nilai `c` menjadi 9.

- Jadi, ketika memanggil `System.out.println(c--)`, nilai yang ditampilkan adalah 10, kemudian `c` diturunkan menjadi 9.

- Pre-Decrement (--d):
  - Langkah 1: Nilai variabel diturunkan (decrement) terlebih dahulu, kemudian ditampilkan setelah penurunan.
  - Contoh:
    - Nilai awal variabel d: 10.
    - Ketika --d dieksekusi:
      - Nilai d langsung menjadi 9 sebelum ditampilkan.
      - Jadi, ketika memanggil System.out.println(--d), nilai yang ditampilkan adalah 9 (karena sudah diturunkan sebelum ditampilkan).

Kesimpulan:

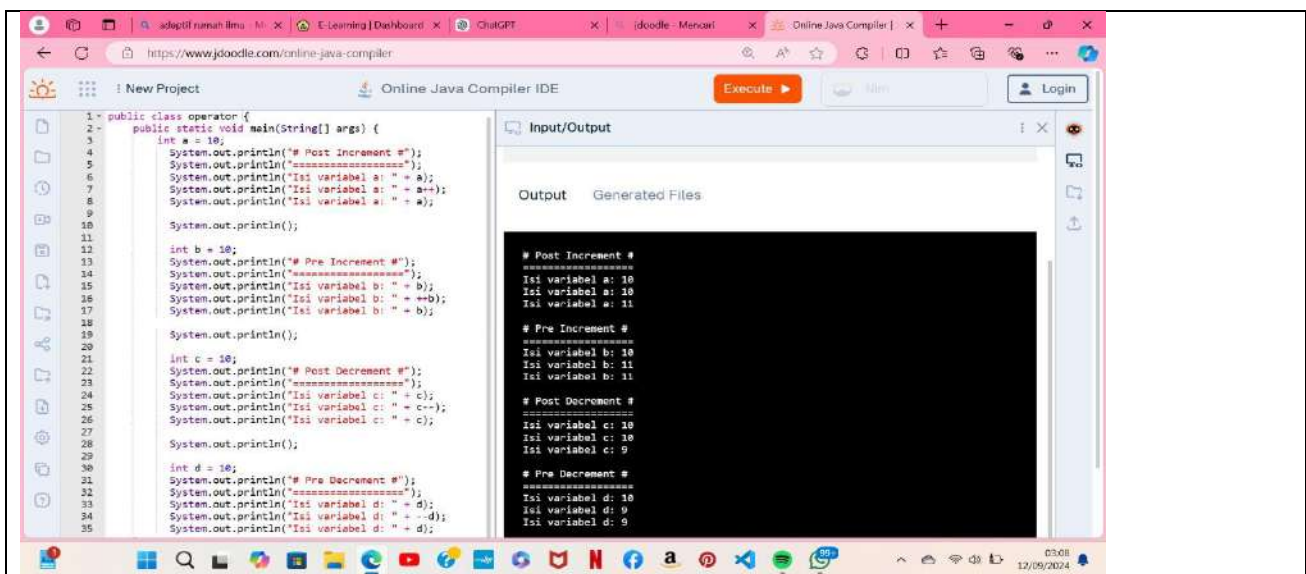
- Post-Increment/Decrement: Menampilkan nilai variabel terlebih dahulu, kemudian melakukan perubahan.
- Pre-Increment/Decrement: Mengubah nilai variabel terlebih dahulu, kemudian menampilkannya

#### **[Nomor 4] Penyusunan Algoritma dan Kode Program**

##### 1. Algoritma

- a. Mulai
- b. Inisialisasi: Setel variabel a, b, c, dan d ke 10.
- c. Post Increment:
  - Tampilkan nilai a.
  - Tampilkan hasil dari a++.
  - Tampilkan nilai a setelah a++.
- d. Pre Increment:
  - Tampilkan nilai b.
  - Tampilkan hasil dari ++b.
  - Tampilkan nilai b setelah ++b.
- e. Post Decrement:
  - Tampilkan nilai c.
  - Tampilkan hasil dari c--.
  - Tampilkan nilai c setelah c--.
- f. Pre Decrement:
  - Tampilkan nilai d.
  - Tampilkan hasil dari --d.
  - Tampilkan nilai d setelah --d.
- h. Selesai

##### 2. Kode program dan luaran



Luaran sudah sesuai dengan program yang disusun.

#### [Nomor 4] Kesimpulan

Analisa

Post Increment (a++) dan Post Decrement (c--): Mencetak nilai variabel sebelum melakukan perubahan (increment/decrement).

Pre Increment (++b) dan Pre Decrement (--d): Mencetak nilai variabel setelah melakukan perubahan (increment/decrement).

Dengan kata lain, Post mengubah nilai setelah cetak, sedangkan Pre mengubah nilai sebelum cetak.

#### Template Lembar Kerja Individu dan Kelompok

<b>Nama &amp; NPM</b>	<b>Topik:</b>	<b>Tanggal:</b>
<b>Yeni kusherawati G1F024013</b>	<b>OPERATOR JAVA</b>	<b>7 september 2024</b>
<b>[Nomor 5] Identifikasi Masalah:</b>		
<pre> public class OperatorLogika {     public static void main (String [] args) {         boolean a = true; </pre>		

```

        boolean b = false;
        boolean c;
        c = a && b;
        System.out.println("true && false = " + c);
    }
}

```

#### **Luaran:**

```
true && false = false
```

Program Java ini mendemonstrasikan penggunaan operator logika AND (&&). Program melakukan hal berikut:

Mendeklarasikan Variabel: Dua variabel boolean, a dengan nilai true dan b dengan nilai false, serta variabel c untuk menyimpan hasil.

Menghitung Hasil: Menggunakan a && b untuk menghitung hasil logika AND antara a dan b, yang disimpan di c. Hasilnya adalah false karena salah satu operand adalah false.

Menampilkan Hasil: Mencetak hasil dari operasi logika ke konsol, menampilkan true && false = false.

### **[Nomor 5] Analisis dan Argumentasi**

#### **Latihan 5**

5.1. Tambahkan baris kode untuk memeriksa a || b.

Jawab:

- a || b adalah operasi logika OR (atau), yang akan menghasilkan true jika salah satu operand (a atau b) bernilai true.
- Kode ini menambahkan operasi logika OR ke program untuk memeriksa hasil dari a || b.

Kode Tambahan:

```

java
boolean d;
d = a || b;
System.out.println("true || false = " + d);

```

Hasil Output:

- Untuk nilai a = true dan b = false:
 

```

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

```
- true || false menghasilkan true karena salah satu operand (a) bernilai true.

5.2. Ubahlah nilai a = false dan b = false. Analisa perubahan dan perbedaan boolean yang terjadi!

Jawab: Jika kita mengubah nilai a dan b menjadi false, kedua operator logika (AND dan OR) akan memberikan hasil yang berbeda.

Analisis:

- a && b (false && false): Operator logika AND menghasilkan false karena kedua operand harus true untuk menghasilkan true. Jika kedua operand bernilai false, hasilnya tetap false.
- a || b (false || false): Operator logika OR menghasilkan false jika kedua operand bernilai false. Untuk OR, hasilnya akan true hanya jika salah satu operand bernilai true.

5.2. Apabila diketahui pernyataan a || b && a || !b. Uraikan urutan logika yang akan dikerjakan! Analisa luaran true atau false dari pernyataan tersebut!

jawab: Uraian ini akan membahas urutan logika yang akan dieksekusi dan hasil dari pernyataan tersebut.

Urutan Evaluasi:

1. Negasi b:

- !b artinya "tidak b", sehingga !false menjadi true.

2. Evaluasi a || !b:

- a bernilai false, dan !b bernilai true. Sehingga, false || true akan menghasilkan true.



### 3. Evaluasi `a && (a || !b)`:

- `a` bernilai `false`, dan hasil dari `(a || !b)` adalah `true`. Sehingga, `false && true` akan menghasilkan `false`.

Hasil Output:

`false && (false || !false) = false`

Analisa:

- Urutan Eksekusi: Pertama, operator NOT (!) dievaluasi, kemudian operator OR (||), dan akhirnya operator AND (&&). Ini mengikuti urutan prioritas operator logika di Java.

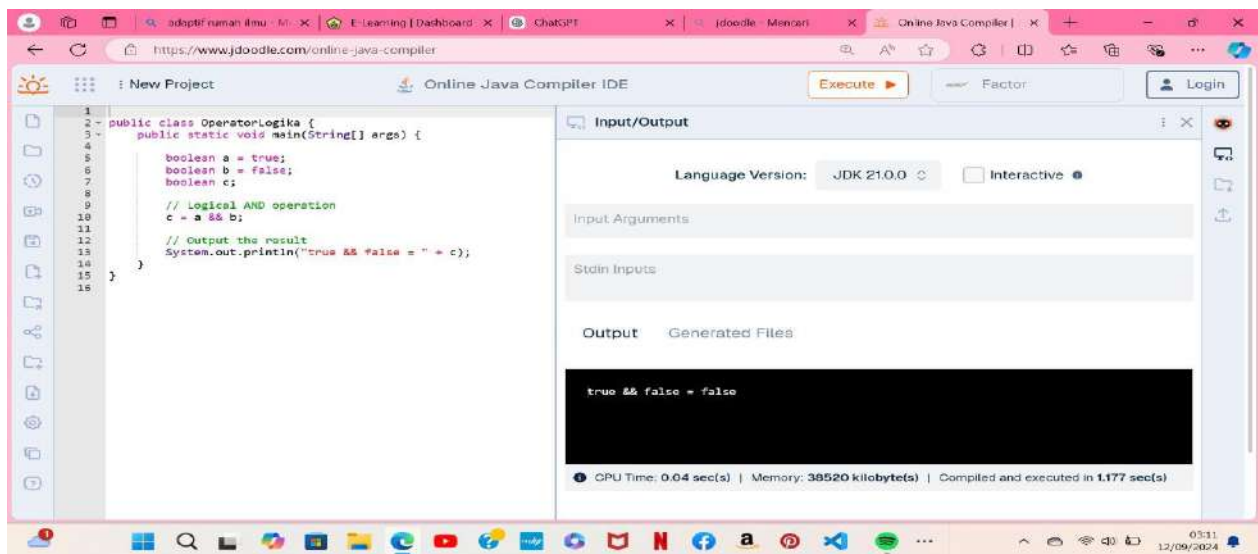
- Hasil: Meskipun `a || !b` menghasilkan `true`, hasil akhir `false && true` tetap `false` karena untuk operator AND, kedua operand harus `true` agar hasilnya `true`.

## [Nomor 5] Penyusunan Algoritma dan Kode Program

Algoritma

- a) Mulai
- b) Inisialisasi variabel
- c) Negasi `b`
- d) Evaluasi ekspresi dalam kurung `(a || !b)`:
- e) Evaluasi ekspresi akhir `(a && (a || !b)b)`:
- f) Cetak hasil
- g) Selesai

Kode program dan luaran



The screenshot shows a web browser window with the URL `https://www.jdoodle.com/online-java-compiler`. The IDE displays a Java program in a file named `New Project`. The code is as follows:

```
1 public class OperatorLogika {
2     public static void main(String[] args) {
3
4         boolean a = true;
5         boolean b = false;
6         boolean c;
7
8         // Logical AND operation
9         c = a && b;
10
11         // Output the result
12         System.out.println("true && false = " + c);
13     }
14 }
15
16
```

The right-hand side of the IDE shows the 'Input/Output' panel. It indicates the 'Language Version' is 'JDK 21.0.0' and 'Interactive' is checked. Under the 'Output' tab, the result of the program execution is displayed:

```
true && false = false
```

At the bottom of the output panel, it shows performance metrics: 'CPU Time: 0.04 sec(s) | Memory: 38520 kilobyte(s) | Compiled and executed in 1.177 sec(s)'.

Luaran sudah sesuai dengan program yang disusun.

## [Nomor Soal 5] Kesimpulan

Analisa

Operator logika `&&` hanya menghasilkan `true` jika kedua operand-nya bernilai `true`.

Operator logika `||` menghasilkan `true` jika salah satu operand bernilai `true`.

Negasi (!) membalik nilai boolean.

Pada ekspresi kompleks seperti `a && (a || !b)`, urutan evaluasi dimulai dari dalam kurung dan operator ! dievaluasi terlebih dahulu, sesuai dengan aturan prioritas operator.

Program ini mengajarkan cara kerja dasar operator logika AND (&&), OR (||), dan NOT (!).

--

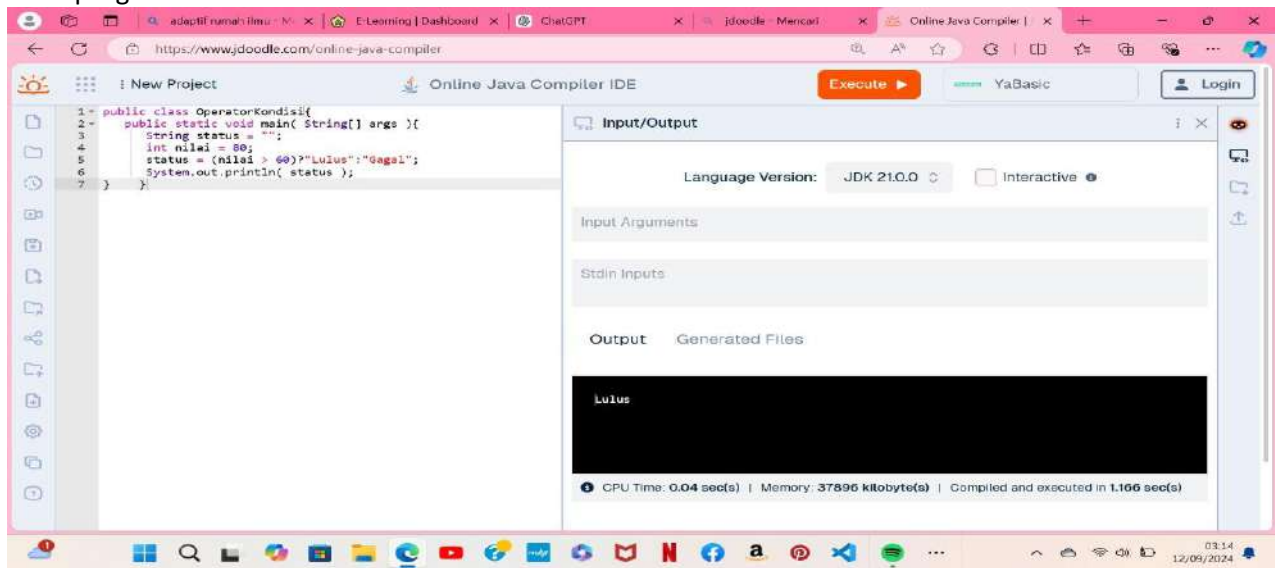
<b>Nama &amp; NPM Template Lembar Kerja Individu dan Kelompok</b>	<b>Topik:</b>	<b>Tanggal:</b>
<b>Yeni kusherawati G1F024013</b>	<b>OPERATOR JAVA</b>	<b>7 september 2024</b>
<b>[Nomor 6] Identifikasi Masalah:</b>		
<pre> public class OperatorKondisi{     public static void main( String[] args ){         String status = "";         int nilai = 80;         status = (nilai &gt; 60)?"Lulus":"Gagal";         System.out.println( status );     } } </pre> <p><b>Luaran:</b> Lulus</p> <p>Program ini menentukan status kelulusan berdasarkan nilai yang diberikan. Jika nilai lebih dari 60, maka hasilnya adalah "Lulus". Jika nilai kurang atau sama dengan 60, hasilnya adalah "Gagal". Dalam contoh ini, karena nilai adalah 80 (lebih dari 60), hasil yang dicetak adalah "Lulus".</p> <p>Meskipun program ini bekerja dengan baik secara dasar, terdapat beberapa potensi masalah dan area untuk perbaikan, seperti kondisi batas nilai yang mungkin tidak sesuai dengan harapan umum. Tidak adanya validasi input untuk memastikan nilai berada dalam rentang yang masuk akal (0-100).</p>		
<b>[Nomor 6] Analisis dan Argumentasi</b> <b>Latihan 6</b> Berdasarkan Contoh 6, ubahlah nilai = 60. Analisis hasil dan proses yang terjadi! jawab: Perubahan Nilai ke 60: Saat nilai diubah menjadi 60, hasil yang muncul adalah "Gagal". Hal ini terjadi karena kondisi yang diperiksa adalah nilai > 60. Dengan nilai 60, kondisi ini salah (karena 60 tidak lebih besar dari 60), sehingga program memberikan status "Gagal". Analisis: Ketika nilai sama dengan 60, kondisi nilai > 60 menghasilkan false, karena 60 tidak lebih besar dari 60. Dengan demikian, program mencetak "Gagal". Namun, jika dalam konteks penilaian nilai 60 dianggap sebagai kelulusan, logika dalam program perlu diubah untuk mempertimbangkan bahwa nilai 60 adalah "Lulus".		

## [Nomor 6] Penyusunan Algoritma dan Kode Program

### 1. Algoritma

- a) Mulai.
- b) Inisialisasi variabel status dengan nilai kosong.
- c) Inisialisasi variabel nilai dengan nilai 80.
- d) Jika nilai > 60, maka:  
Set status menjadi "Lulus".
- e) Jika tidak:  
Set status menjadi "Gagal".
- f) Tampilkan nilai status.
- g) Selesai.

### Kode program dan luaran



Luaran sudah sesuai dengan program yang disusun.

## [Nomor 6] Kesimpulan

### Analisa

#### Efisiensi dan Kejelasan

Operator ternary ? : memungkinkan untuk menulis kondisi sederhana dalam satu baris, sehingga membuat kode lebih ringkas dan mudah dibaca. Ini sangat cocok untuk keputusan yang sederhana, seperti memilih antara dua nilai berdasarkan kondisi tertentu.

#### Kesimpulan:

Operator ternary adalah alat yang berguna untuk menulis kondisi sederhana secara lebih ringkas. Namun, untuk kasus-kasus yang lebih kompleks, penggunaannya dapat membuat kode lebih sulit dipahami. Jadi, penggunaannya harus dipertimbangkan berdasarkan kebutuhan dan kompleksitas logika yang dihadapi.

<b>Nama &amp; NPM Template Lembar Kerja Individu dan Kelompok</b>	<b>Topik:</b>	<b>Tanggal:</b>
<b>Yeni kusherawati G1F024013</b>	<b>OPERATOR JAVA</b>	<b>7 september 2024</b>
<b>[Nomor 7] Identifikasi Masalah:</b> Berikut penjelasan singkat tentang operator bitwise pada kode tersebut: <p>1.&amp; (AND bitwise): Menghasilkan 1 jika kedua bit pada posisi yang sama bernilai 1. Misalnya, 10 &amp; 7 menghasilkan 2.</p> <p>2 (OR bitwise): Menghasilkan 1 jika salah satu atau kedua bit bernilai 1. Misalnya, 10   7 menghasilkan 15.</p> <p>3(XOR bitwise): Menghasilkan 1 jika kedua bit berbeda. Misalnya, 10 ^ 7 menghasilkan 13.</p> <p>4 (NOT bitwise): Membalik setiap bit. Misalnya, ~10 menghasilkan -11 karena menggunakan komplement dua.</p> <p>5 (Right Shift): Menggeser bit ke kanan, membuang bit yang paling kanan. Misalnya, 10 &gt;&gt; 1 menghasilkan 5.</p> <p>6(Left Shift): Menggeser bit ke kiri, menambah 0 di kanan. Misalnya, 7 &lt;&lt; 2 menghasilkan 28.</p> <p>Setiap operator bekerja pada level bit, memanipulasi angka dalam representasi biner.</p>		
<b>[Nomor 7] Analisis dan Argumentasi</b>		
<b>Latihan 7</b> Pilihlah 3 perhitungan Contoh 7, kemudian uraikan perhitungan biner! Simpulkan hasilnya! jawab:		

tiga contoh perhitungan:  $a \& b$ ,  $a | b$ , dan  $a \gg 1$ . Berikut uraian perhitungan biner dari masing-masing operasi:

1.  $a \& b$  (AND bitwise)

-  $a = 10$  (1010 dalam biner)

-  $b = 7$  (0111 dalam biner)

Operasi AND bitwise membandingkan setiap bit dari kedua bilangan. Hanya bit yang sama-sama 1 yang akan menghasilkan 1, selebihnya 0.

Perhitungan biner:

1010 ( $a = 10$ )

0111 ( $b = 7$ )

----

0010 (hasil = 2)

Kesimpulan: Hasil dari  $a \& b$  adalah 2, karena hanya bit ke-2 dari kanan yang sama-sama 1 pada kedua bilangan.

2.  $a | b$  (OR bitwise)

-  $a = 10$  (1010 dalam biner)

-  $b = 7$  (0111 dalam biner)

Operasi OR bitwise menghasilkan 1 jika salah satu atau kedua bit pada posisi yang sama bernilai 1.

Perhitungan biner:

1010 ( $a = 10$ )

0111 ( $b = 7$ )

----

1111 (hasil = 15)

Kesimpulan: Hasil dari  $a | b$  adalah 15, karena setiap posisi bit minimal ada satu 1.

3.  $a \gg 1$  (Right Shift bitwise)

-  $a = 10$  (1010 dalam biner)

Operasi  $\gg$  menggeser bit ke kanan sebanyak satu posisi. Bit yang paling kanan dibuang, dan bit paling kiri diisi dengan 0 (untuk bilangan positif).

Perhitungan biner:

1010 ( $a = 10$ )

$\gg 1$

----

0101 (hasil = 5)

Kesimpulan: Hasil dari  $a \gg 1$  adalah 5, karena bit-bit dalam  $a$  digeser satu posisi ke kanan, menghilangkan bit paling kanan dan mengisi bit paling kiri dengan 0.

## [Nomor 7] Penyusunan Algoritma dan Kode Program

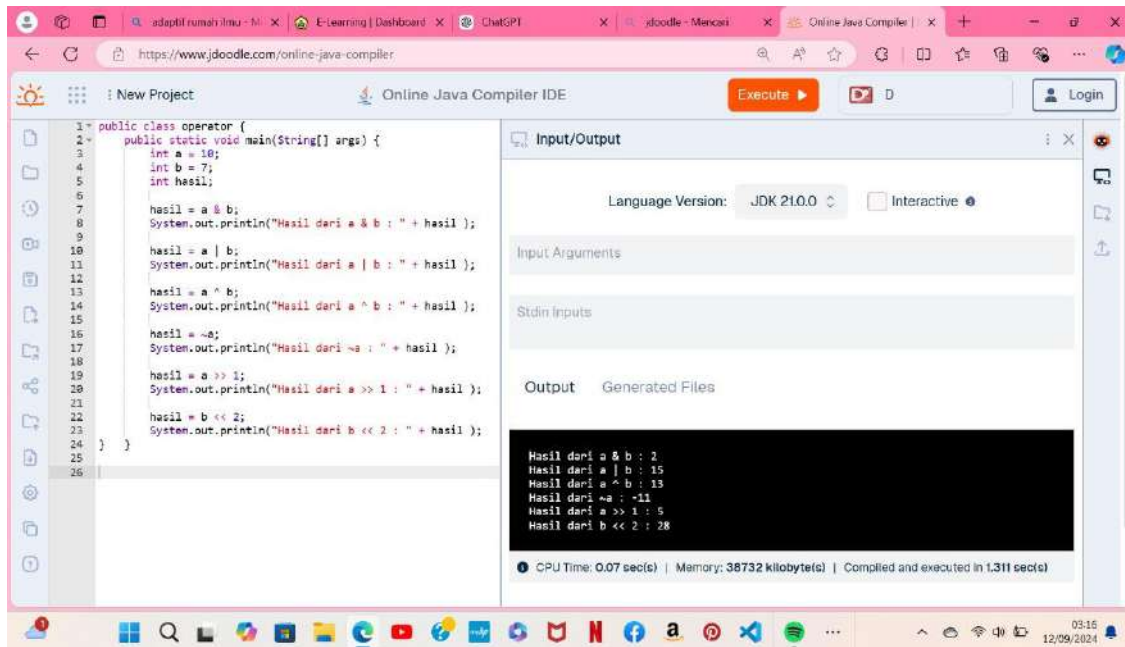
### 1. Algoritma

1. Mulai

2. Deklarasikan variabel  $a$  dengan nilai 10 dan  $b$  dengan nilai 7.

3. Deklarasikan variabel hasil untuk menyimpan hasil operasi.
4. Lakukan operasi `a

## 2. Kode program dan luaran



The screenshot shows a web browser window with the URL <https://www.jdoodle.com/online-java-compiler>. The page title is "Online Java Compiler IDE". The code editor contains the following Java code:

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

The "Input/Output" panel on the right shows the output of the program:

```
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
```

Below the output, it shows performance metrics: CPU Time: 0.07 sec(s) | Memory: 38732 kilobyte(s) | Compiled and executed in 1.311 sec(s).

Luaran sudah sesuai dengan program yang disusun.

## [Nomor 7] Kesimpulan

### Analisa

Operator bitwise memungkinkan manipulasi langsung pada bit-bit individual dari bilangan biner. Setiap operator memiliki cara kerja yang unik dan berguna untuk berbagai tujuan, terutama dalam pemrograman yang membutuhkan pemrosesan data pada level rendah atau optimasi bit.