

<b>Nama &amp; NPM</b>	<b>Topik:</b>	<b>Tanggal:</b>
<b>Ariiq Ashar Sofyan G1F02405</b>	<b>Operator Aritmatika</b>	<b>14 September 2024</b>
<b>[No. 1] Identifikasi Masalah:</b>		
<p><b>Contoh 1:</b> Salin dan tempel kode program berikut ke Eclipse atau JDoodle</p> <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)); // menampilkan hasil         penjumlahan     } }</pre> <p><b>Luaran:</b></p> <pre>Exception in thread "main" java.lang.Error: Unresolved compilation problems:   Syntax error on token ""a + b = "", AssignmentOperator expected after this token   The left-hand side of an assignment must be a variable</pre> <p><b>Latihan 1</b></p> <p><b>1.1.</b> Rekomendasikan perbaikan kode agar program Contoh 1 dapat berjalan!</p> <p><b>1.2.</b> Tambahkan baris untuk menampilkan perhitungan dengan operator ( -, *, / , %) pada Contoh 1!</p>		
<b>[No. 1] Analisis dan Argumentasi</b>		
<p><b>1.1. Rekomendasikan perbaikan kode agar program Contoh 1 dapat berjalan!</b> Kesalahan pada kode adalah kurangnya tanda plus (+) dalam pernyataan <code>System.out.println("a + b = " + (a + b));</code>. Seharusnya operator + digunakan untuk menggabungkan string dan ekspresi aritmatika. Berikut adalah perbaikan kode yang benar:</p> <p><b>Input :</b></p> <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)); // Menampilkan hasil         penjumlahan     } }</pre>		

**Output:**

```
a: 20
b: 3
a + b = 23
```

**1.2. Tambahkan baris untuk menampilkan perhitungan dengan operator ( -, \*, /, %) pada Contoh 1!**

Berikut adalah kode yang diperbarui dengan tambahan operator aritmatika lain:

**Input :**

```
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);

        // Menampilkan hasil operasi aritmatika
        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
bulat)
        System.out.println("a % b = " + (a % b)); // Modulus (sisa
bagi)
    }
}
```

**Output :**

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

**[No. 1] Penyusunan Algoritma dan Kode Program****1.1. Rekomendasikan perbaikan kode agar program Contoh 1 dapat berjalan!****Algoritma**

1. Mulai
2. Deklarasikan dua variabel a dan b dengan nilai awal (contoh: a = 20, b = 3).
3. Cetak nilai dari a dan b.
4. Hitung penjumlahan a + b dan cetak hasilnya.
5. Selesai

### 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
6         // Operator aritmatika
7         System.out.println("a: " + a);
8         System.out.println("b: " + b);
9         System.out.println("a + b = " + (a + b)); // Menampilkan hasil penjumlahan
10    }
11 }
12
```

#### Output Generated Files

```
a: 20
b: 3
a + b = 23
```

Luaran sudah sesuai dengan program yang disusun. Tipe data yang ditampilkan telah sesuai dengan kebutuhan dan permintaan data.

### 1.2. Tambahkan baris untuk menampilkan perhitungan dengan operator ( -, \*, /, %) pada Contoh 1!

#### Algoritma

1. Mulai
2. Deklarasikan dua variabel a dan b dengan nilai awal (contoh: a = 20, b = 3).
3. Cetak nilai dari a dan b.
4. Hitung penjumlahan a + b dan cetak hasilnya.
5. Hitung pengurangan a - b dan cetak hasilnya.
6. Hitung perkalian a \* b dan cetak hasilnya.
7. Hitung pembagian a / b dan cetak hasilnya.
8. Hitung modulus a % b (sisu pembagian) dan cetak hasilnya.
9. Selesai

### 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
6         // Operator aritmatika
7         System.out.println("a: " + a);
8         System.out.println("b: " + b);
9
10        // Menampilkan hasil operasi aritmatika
11        System.out.println("a + b = " + (a + b)); // Penjumlahan
12        System.out.println("a - b = " + (a - b)); // Pengurangan
13        System.out.println("a * b = " + (a * b)); // Perkalian
14        System.out.println("a / b = " + (a / b)); // Pembagian (hasil bulat)
15        System.out.println("a % b = " + (a % b)); // Modulus (sisu bagi)
16    }
17 }
18
```

#### Output Generated Files

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

Luaran sudah sesuai dengan program yang disusun. Tipe data yang ditampilkan telah sesuai dengan kebutuhan dan permintaan data.

<b>[No. 1] Kesimpulan</b>
<b>1) Analisa</b> <ul style="list-style-type: none"><li>a) Permasalahan terjadi karena kesalahan penulisan tanda, dan setelah diperbaiki, kode dapat berjalan dengan baik. Algoritma sederhana dan mudah dipahami, serta program dapat ditingkatkan dengan menambahkan operasi aritmatika lainnya.</li></ul>

<b>Nama &amp; NPM</b>	<b>Topik:</b>	<b>Tanggal:</b>
<b>Ariiq Ashar Sofyan G1F02405</b>	<b>Operator Penugasan</b>	<b>14 September 2024</b>

**[No. 2] Identifikasi Masalah:**

**Contoh 2:**

Salin dan tempel kode program berikut ke Eclipse atau JDoodle

```
public class OperatorPenugasan {
    public static void main(String[] args) {
        // deklarasi nilai
        int a = 20, b = 3;
        //operator penugasan
        b += a; //melakukan perhitungan penjumlahan
        System.out.println("Penambahan : " + b); // menampilkan hasil
        perhitungan penjumlahan
    }
}
```

**Luaran:**

Penambahan : 23

**Latihan 2.**

**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. 1] Analisis dan Argumentasi**

**2.1. Tambahkan baris Contoh 2 untuk menampilkan perhitungan dengan operator ( -=, \*=, /=, %=)!**

Kode program di bawah ini adalah contoh modifikasi untuk menambahkan operasi penugasan lain, seperti pengurangan (-=), perkalian (\*=), pembagian (/=), dan modulus (%=).

**Input :**

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

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

        // reset nilai b untuk operasi selanjutnya
        b = 3;
        b -= a; // pengurangan
        System.out.println("Pengurangan : " + b); // hasil pengurangan

        b = 3;
        b *= a; // perkalian
        System.out.println("Perkalian : " + b); // hasil perkalian

        b = 3;
```

```

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

        b = 3;
        b %= a; // modulus (sisir bagi)
        System.out.println("Modulus : " + b); // hasil modulus
    }
}

```

#### Output:

```

Penambahan : 23
Pengurangan : -17
Perkalian : 60
Pembagian : 0
Modulus : 3

```

## 2.2. Berikan argumentasi tentang perbedaan luaran dan waktu eksekusi Contoh 1 dan Contoh 2!

Perbedaan Luaran:

- Contoh 1 menampilkan hasil operasi aritmatika dasar (+, -, \*, /, %) pada dua variabel a dan b, dan setiap operasi dilakukan secara terpisah. Misalnya, hasil penjumlahan, pengurangan, dll. ditampilkan satu per satu, tanpa memodifikasi nilai a dan b.
- Contoh 2 menggunakan **operator penugasan** (+=, -=, \*=, /=, %=) yang mengubah nilai variabel secara langsung setelah setiap operasi. Setiap hasil operasi memperbarui variabel b, sehingga nilai b berubah setiap kali operator penugasan digunakan.

Perbedaan Waktu Eksekusi:

- Secara umum, **waktu eksekusi** untuk kedua contoh **tidak akan signifikan berbeda** karena kedua program melakukan operasi aritmatika dasar yang memiliki kompleksitas waktu konstan ( $O(1)$ ). Namun, dalam **Contoh 2**, karena nilai variabel b diubah setelah setiap operasi, program mungkin terlihat lebih efisien dalam penggunaan memori, karena tidak membuat variabel baru untuk setiap operasi.

Contoh 1 hanya menampilkan hasil operasi dan tidak mengubah nilai asli a dan b. Sedangkan Contoh 2 mengubah nilai b setelah setiap operasi penugasan.

## [No. 2] Penyusunan Algoritma dan Kode Program

### 2.1. Tambahkan baris Contoh 2 untuk menampilkan perhitungan dengan operator ( -=, \*=, /=, %=)!

#### Algoritma

1. Mulai program.
2. Deklarasikan dua variabel a dan b dengan nilai 20 dan 3.
3. Lakukan operasi penugasan:
  - Tambahkan a ke b menggunakan +=, dan tampilkan hasilnya.
  - Atur ulang nilai b ke 3.
  - Kurangkan a dari b menggunakan -=, dan tampilkan hasilnya.
  - Atur ulang nilai b ke 3.
  - Kalikan b dengan a menggunakan \*=, dan tampilkan hasilnya.
  - Atur ulang nilai b ke 3.
  - Bagi b dengan a menggunakan /=, dan tampilkan hasilnya.
  - Atur ulang nilai b ke 3.

- Hitung sisa pembagian b oleh a menggunakan %= dan tampilkan hasilnya.
4. Akhiri program.

#### 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-
6-         // operator penugasan
7-         b += a; // penjumlahan
8-         System.out.println("Penambahan : " + b); // hasil penjumlahan
9-
10-        // reset nilai b untuk operasi selanjutnya
11-        b = 3;
12-        b -= a; // pengurangan
13-        System.out.println("Pengurangan : " + b); // hasil pengurangan
14-
15-        b = 3;
16-        b *= a; // perkalian
17-        System.out.println("Perkalian : " + b); // hasil perkalian
18-
19-        b = 3;
20-        b /= a; // pembagian
21-        System.out.println("Pembagian : " + b); // hasil pembagian (integer)
22-
23-        b = 3;
24-        b %= a; // modulus (sisa bagi)
25-        System.out.println("Modulus : " + b); // hasil modulus
26-    }
27- }
28-

```

Output    Generated Files

```

Penambahan : 23
Pengurangan : -17
Perkalian : 60
Pembagian : 0
Modulus : 3

```

Luaran sudah sesuai dengan program yang disusun. Tipe data yang ditampilkan telah sesuai dengan kebutuhan dan permintaan data.

#### 2.2. Berikan argumentasi tentang perbedaan luaran dan waktu eksekusi Contoh 1 dan Contoh 2! Algoritma

1. Analisis dan bandingkan luaran yang dihasilkan dari Contoh 1 dan Contoh 2.
2. Perhatikan bahwa Contoh 1 menampilkan hasil operasi aritmatika secara eksplisit tanpa mengubah nilai variabel, sementara Contoh 2 mengubah nilai variabel setelah setiap operasi dengan operator penugasan.
3. Perhatikan bahwa waktu eksekusi pada kedua contoh secara umum serupa karena keduanya menggunakan operasi aritmatika dasar yang cepat dan efisien ( $O(1)$ ).
4. Eksekusi dalam Contoh 2 lebih efisien dalam penggunaan variabel karena nilai b diperbarui langsung tanpa menggunakan variabel tambahan.

#### [No. 2] Kesimpulan

##### 1) Analisa

- a. Operator Penugasan (seperti +=, -=, \*=, /=, %=) mempermudah proses perhitungan karena hasil operasi langsung disimpan kembali ke dalam variabel yang dioperasikan, membuat kode lebih ringkas dan efisien dalam hal memori.
- b. Perbedaan Luaran antara Contoh 1 dan Contoh 2 terletak pada cara pengelolaan nilai variabel:
  - Contoh 1 melakukan operasi aritmatika pada variabel tetap dan tidak mengubah nilai variabel.
  - Contoh 2 mengubah nilai variabel b setelah setiap operasi dengan operator penugasan, sehingga hasil operasi berikutnya tergantung pada nilai yang diperbarui.

- c. Waktu Eksekusi pada kedua contoh tidak memiliki perbedaan yang signifikan karena keduanya menggunakan operasi dasar yang sangat efisien. Namun, Contoh 2 memiliki keunggulan dalam efisiensi memori karena tidak perlu membuat variabel baru untuk menyimpan hasil sementara dari setiap operasi.

Kesimpulannya, penggunaan operator penugasan dalam Contoh 2 lebih ringkas dan efisien, terutama untuk program yang membutuhkan pembaruan nilai variabel secara langsung.



<b>Nama &amp; NPM</b>	<b>Topik:</b>	<b>Tanggal:</b>
<b>Ariiq Ashar Sofyan G1F02405</b>	<b>Operator Relasional</b>	<b>14 September 2024</b>

**[No. 3] Identifikasi Masalah:**

**Contoh 3:**

Salin dan tempel kode program berikut ke Eclipse atau JDoodle

```
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("\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);
    }
}
```

**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
```

**Latihan 3**

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

**[No. 3] Analisis dan Argumentasi**

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

Ketika nilai A dan B sama (A = 4, B = 4), maka perubahan yang terjadi adalah pada hasil operasi relasional karena nilai A sekarang sama dengan nilai B. Berikut adalah kode yang sudah dimodifikasi dan luaran yang dihasilkan

1.  $A > B$  dan  $A < B$ : Kedua operasi ini memberikan false karena nilai A sama dengan B, sehingga A tidak lebih besar atau lebih kecil dari B.
2.  $A \geq B$  dan  $A \leq B$ : Kedua operasi ini menghasilkan true karena nilai A sama dengan B, dan kesamaan memenuhi kondisi "lebih besar atau sama dengan" maupun "lebih kecil atau sama dengan".
3.  $A == B$ : Operator kesamaan menghasilkan true, karena nilai A memang sama dengan B.
4.  $A != B$ : Operator ketidaksamaan memberikan false karena A dan B memiliki nilai yang sama, sehingga mereka tidak berbeda.

**Input :**

```
public class OperatorRelasional {
    public static void main(String[] args) {
        int nilaiA = 4; // Ubah nilai A menjadi 4
        int nilaiB = 4; // Ubah nilai B menjadi 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);
    }
}
```

**Output:**

```
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
```

### [No. 3] Penyusunan Algoritma dan Kode Program

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

##### Algoritma

6. Mulai
7. Deklarasikan dua variabel a dan b dengan nilai awal (contoh: a = 20, b = 3).
8. Cetak nilai dari a dan b.
9. Hitung penjumlahan a + b dan cetak hasilnya.
10. Selesai

##### 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 *
6 *         // Operator aritmatika
7 *         System.out.println("a: " + a);
8 *         System.out.println("b: " + b);
9 *         System.out.println("a + b = " + (a + b)); // Menampilkan hasil penjumlahan
10 *     }
11 * }
12
```

Output      Generated Files

```
a: 20
b: 3
a + b = 23
```

Luaran sudah sesuai dengan program yang disusun. Tipe data yang ditampilkan telah sesuai dengan kebutuhan dan permintaan data.

#### 1.2. Tambahkan baris untuk menampilkan perhitungan dengan operator ( -, \*, /, %) pada Contoh 1!

##### Algoritma

1. Mulai program.
2. Deklarasikan dua variabel nilaiA dan nilaiB masing-masing dengan nilai 4.
3. Tampilkan nilai nilaiA dan nilaiB.
4. Bandingkan nilaiA dan nilaiB menggunakan operator relasional:
  - a. Apakah nilaiA lebih besar dari nilaiB?
  - b. Apakah nilaiA lebih kecil dari nilaiB?
  - c. Apakah nilaiA lebih besar atau sama dengan nilaiB?
  - d. Apakah nilaiA lebih kecil atau sama dengan nilaiB?
  - e. Apakah nilaiA sama dengan nilaiB?
  - f. Apakah nilaiA tidak sama dengan nilaiB?
5. Tampilkan hasil dari setiap perbandingan.
6. Akhiri program.

### Kode program dan luaran

```
6
7      System.out.println(" A = " + nilaiA + "\n B = " + nilaiB);
8
9      // apakah A lebih besar dari B?
10     hasil = nilaiA > nilaiB;
11     System.out.println("\n Hasil A > B = " + hasil);
12
13     // apakah A lebih kecil dari B?
14     hasil = nilaiA < nilaiB;
15     System.out.println("\n Hasil A < B = " + hasil);
16
17     // apakah A lebih besar samadengan B?
18     hasil = nilaiA >= nilaiB;
19     System.out.println("\n Hasil A >= B = " + hasil);
20
21     // apakah A lebih kecil samadengan B?
22     hasil = nilaiA <= nilaiB;
23     System.out.println("\n Hasil A <= B = " + hasil);
24
25     // apakah nilai A sama dengan B?
26     hasil = nilaiA == nilaiB;
27     System.out.println("\n Hasil A == B = " + hasil);
28
29     // apakah nilai A tidak samadengan B?
30     hasil = nilaiA != nilaiB;
31     System.out.println("\n Hasil A != B = " + hasil);
32 }
33
34
```

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
```

Luaran sudah sesuai dengan program yang disusun. Tipe data yang ditampilkan telah sesuai dengan kebutuhan dan permintaan data.

### [No. 3] Kesimpulan

#### 1. Analisa

- a) Ketika nilai A dan B sama, beberapa operator relasional seperti ==, >=, dan <= akan memberikan hasil true karena kedua nilai dianggap sama. Sebaliknya, operator yang membandingkan apakah satu nilai lebih besar atau lebih kecil dari yang lain (>, <, !=) akan memberikan hasil false, karena tidak ada perbedaan antara nilai A dan B.

<b>Nama &amp; NPM</b>	<b>Topik:</b>	<b>Tanggal:</b>
<b>Ariiq Ashar Sofyan G1F02405</b>	<b>Operator Increment dan Decrement</b>	<b>14 September 2024</b>

**[No. 4] Identifikasi Masalah:**

**Contoh 4: Salin dan tempel kode program berikut ke Eclipse atau JDoodle**

```
public class operator {
    public static void main(String[] args) {
        // deklarasi nilai
        int a = 5;

        System.out.println("a: " + a);
        System.out.println("b: " + (a++));
    }
}
```

**Luaran:**

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

**Latihan 4.**

- 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**

**4.1. Berikan saran operasi apa yang diperlukan (pre/post increment, pre/post decrement) agar Contoh 4 menghasilkan nilai a = 5 dan b = 6?**

Untuk mendapatkan nilai a = 5 dan b = 6 pada Contoh 4, kita perlu menggunakan pre-increment pada variabel b. Berikut penjelasannya:

- Post-increment (a++) berarti nilai a akan ditampilkan terlebih dahulu, kemudian setelah itu a ditambahkan 1. Itulah sebabnya dalam contoh program asli, nilai b tetap 5 (nilai asli a) dan baru setelah itu a menjadi 6.
- Pre-increment (++a) berarti nilai a ditambahkan 1 terlebih dahulu, kemudian ditampilkan. Jadi, untuk mendapatkan nilai b = 6 sementara a = 5 pada awalnya, kita perlu menggunakan pre-increment pada b sehingga hasil increment langsung diterapkan sebelum nilai ditampilkan.

**Input :**

```
public class Operator {
    public static void main(String[] args) {
        // Deklarasi nilai
        int a = 5;

        // Menampilkan nilai a dan b
        System.out.println("a: " + a);
        System.out.println("b: " + (++a)); // pre-increment, b menjadi 6
    }
}
```

**Output:**

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

#### 4.2. Simpulkan hasil eksperimen Anda!

Penjelasan:

- Pada baris kedua, nilai a akan tetap 5.
- Pada baris ketiga, dengan menggunakan pre-increment (++a), nilai a langsung bertambah menjadi 6 sebelum ditampilkan untuk variabel b.
- Dengan menggunakan post-increment (a++) pada contoh asli, nilai b tetap sama dengan nilai awal a (yaitu 5) dan baru setelahnya a bertambah menjadi 6.
- Ketika kita mengubahnya menjadi pre-increment (++a), nilai a bertambah terlebih dahulu menjadi 6 sebelum digunakan dalam perhitungan atau ditampilkan sebagai b. Hal ini menyebabkan nilai b menjadi 6, yang sesuai dengan hasil yang diinginkan.

#### [No. 4] Penyusunan Algoritma dan Kode Program

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

##### Algoritma

1. Mulai program.
2. Deklarasikan variabel a dengan nilai 5.
3. Tampilkan nilai awal dari a.
4. Gunakan pre-increment (++a) untuk menambah nilai a terlebih dahulu sebelum menampilkan nilai b.
5. Tampilkan nilai b (yang merupakan nilai a setelah di-increment).
6. Akhiri program.

##### Kode program dan luaran

```
1- public class Operator {
2-     public static void main(String[] args) {
3-         // Deklarasi nilai
4-         int a = 5;
5-
6-         // Menampilkan nilai a dan b
7-         System.out.println("a: " + a);
8-         System.out.println("b: " + (++a)); // pre-increment, b menjadi 6
9-     }
10 }
11
```

Output      Generated Files

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

Luaran sudah sesuai dengan program yang disusun. Tipe data yang ditampilkan telah sesuai dengan kebutuhan dan permintaan data.

#### 4.2. Simpulkan hasil eksperimen Anda!

##### Algoritma

1. Mulai program.
2. Deklarasikan variabel a dengan nilai 5.
3. Gunakan post-increment (a++) atau pre-increment (++a) untuk melihat efeknya pada nilai b.
4. Jika menggunakan post-increment (a++):
5. Tampilkan nilai b sebelum a di-increment, sehingga b akan tetap sama dengan nilai awal a.
6. Jika menggunakan pre-increment (++a):
7. Tambahkan nilai a terlebih dahulu, lalu tampilkan nilai b, yang akan menjadi hasil increment.
8. Amati perbedaan antara kedua operasi tersebut dalam memengaruhi nilai variabel yang ditampilkan.
9. Akhiri program.

## 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
6         // Operator aritmatika
7         System.out.println("a: " + a);
8         System.out.println("b: " + b);
9
10        // Menampilkan hasil operasi aritmatika
11        System.out.println("a + b = " + (a + b)); // Penjumlahan
12        System.out.println("a - b = " + (a - b)); // Pengurangan
13        System.out.println("a * b = " + (a * b)); // Perkalian
14        System.out.println("a / b = " + (a / b)); // Pembagian (hasil bulat)
15        System.out.println("a % b = " + (a % b)); // Modulus (siswa bagi)
16    }
17 }
18 }
```

### Output Generated Files

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

Luaran sudah sesuai dengan program yang disusun. Tipe data yang ditampilkan telah sesuai dengan kebutuhan dan permintaan data.

### [No. 4] Kesimpulan

#### 1. Analisa

- Post-increment (a++): Menampilkan nilai asli a, baru setelah itu nilai a ditambahkan.
- Pre-increment (++a): Menambahkan nilai a terlebih dahulu, baru kemudian menampilkannya.

<b>Nama &amp; NPM</b>	<b>Topik:</b>	<b>Tanggal:</b>
<b>Ariiq Ashar Sofyan G1F02405</b>	<b>Operator Logika</b>	<b>14 September 2024</b>

#### [No. 5] Identifikasi Masalah:

**Contoh 5:** Salin dan tempel kode program berikut ke Eclipse atau JDoodle

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

        System.out.println("Hasil logika (a && b) : " + (a &&
b)); //menampilkan hasil logika AND
    } }
```

**Luaran:**

Hasil logika (a && b) : false

#### Latihan 5

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

U

**Input :**

```
public class OperatorLogika {
    public static void main(String[] args) {
        // deklarasi nilai
        boolean a, b;

        // Kasus 1: Menghasilkan true dengan operator &&
        a = true;
        b = true;
        System.out.println("Hasil logika (a && b) : " + (a && b)); // Output:
true

        // Kasus 2: Menghasilkan true dengan operator ||
        a = true;
        b = false;
        System.out.println("Hasil logika (a || b) : " + (a || b)); // Output:
true

        a = false;
        b = true;
        System.out.println("Hasil logika (a || b) : " + (a || b)); // Output:
true

        a = true;
        b = true;
        System.out.println("Hasil logika (a || b) : " + (a || b)); // Output:
true
    } }
```



**Output:**

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

**Penjelasan output**

- Pada kasus pertama, nilai a = true dan b = true, sehingga hasil dari a && b adalah true.
- Pada kasus kedua, meskipun salah satu dari a atau b bernilai false, hasil dari a || b tetap true karena operator || hanya butuh satu operand yang bernilai true.

**5.2. Berikan kesimpulan dari latihan 5.1.**

- Untuk operator && (AND), agar hasilnya true, kedua operand harus bernilai true. Jika salah satu operand bernilai false, maka hasilnya akan false.
- Untuk operator || (OR), hasil akan true jika setidaknya salah satu operand bernilai true. Jika kedua operand bernilai false, maka hasilnya false.

**[No. 5] Penyusunan Algoritma dan Kode Program****5.1. Rekomendasikan berapa nilai a dan b apabila ingin menghasilkan luaran true dengan operator && dan operator || ?****Algoritma****1. Mulai Program****2. Deklarasikan dua variabel boolean a dan b.****3. Langkah 1: Operator AND (&&)**

- Set nilai a dan b menjadi true.
- Lakukan operasi logika a && b.
- Tampilkan hasil operasi logika tersebut.

**4. Langkah 2: Operator OR (||)**

- Set nilai a = true dan b = false.
- Lakukan operasi logika a || b.
- Tampilkan hasil operasi logika tersebut.
- Set nilai a = false dan b = true.
- Lakukan operasi logika a || b.
- Tampilkan hasil operasi logika tersebut.
- Set nilai a = true dan b = true.
- Lakukan operasi logika a || b.
- Tampilkan hasil operasi logika tersebut.

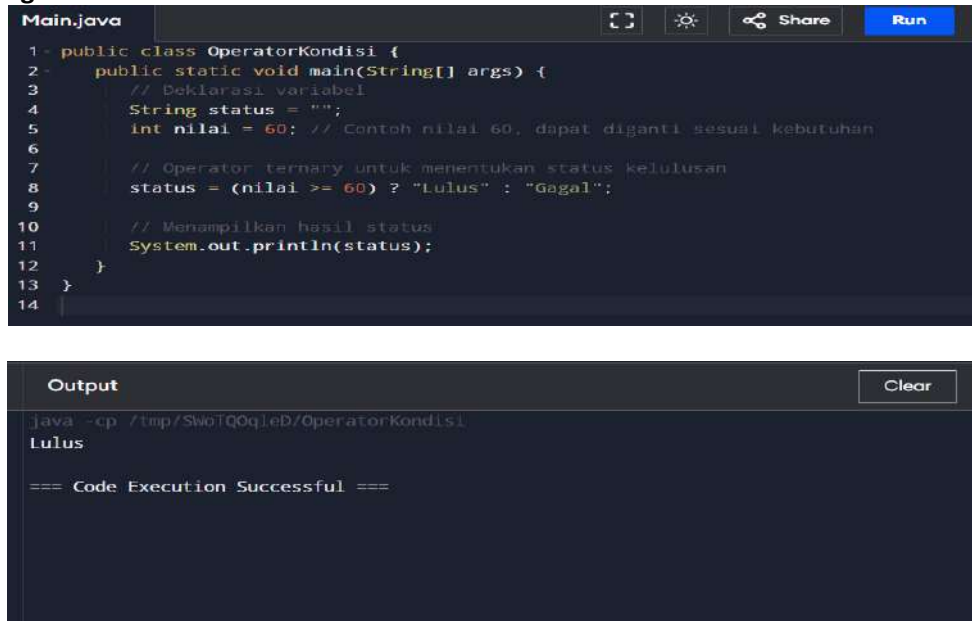
**5. Akhiri Program****Kode program dan luaran**

```
1 - public class OperatorLogika {
2 -     public static void main(String[] args) {
3 -         // Deklarasi variabel boolean
4 -         boolean a, b;
5 -
6 -         // Langkah 1: Menghasilkan true dengan operator AND (&&)
7 -         a = true;
8 -         b = true;
9 -         System.out.println("Hasil logika (a && b) : " + (a && b)); // Output: true
10 -
11 -         // Langkah 2: Menghasilkan true dengan operator OR (||)
12 -         // Kasus a = true, b = false
13 -         a = true;
14 -         b = false;
15 -         System.out.println("Hasil logika (a || b) : " + (a || b)); // Output: true
16 -
17 -         // Kasus a = false, b = true
18 -         a = false;
19 -         b = true;
20 -         System.out.println("Hasil logika (a || b) : " + (a || b)); // Output: true
21 -
22 -         // Kasus a = true, b = true
23 -         a = true;
24 -         b = true;
25 -         System.out.println("Hasil logika (a || b) : " + (a || b)); // Output: true
26 -     }
27 - }
```

	<div data-bbox="317 192 1275 434" data-label="Code-Block"> <pre> Output      Generated Files  Hasil logika (a &amp;&amp; b) : true Hasil logika (a    b) : true Hasil logika (a    b) : true Hasil logika (a    b) : true </pre> </div> <p>Luaran sudah sesuai dengan program yang disusun. Tipe data yang ditampilkan telah sesuai dengan kebutuhan dan permintaan data.</p> <p><b>5.2. Berikan kesimpulan dari latihan 5.1.</b></p> <ol style="list-style-type: none"> <li>1. Mulai Program</li> <li>2. Deklarasikan dua variabel boolean a dan b.</li> <li>3. Langkah 1: Evaluasi Operator &amp;&amp; (AND) <ul style="list-style-type: none"> <li>• Set nilai a dan b menjadi true.</li> <li>• Lakukan operasi logika a &amp;&amp; b dan simpan hasilnya.</li> <li>• Catat bahwa hasilnya adalah true.</li> <li>• Kesimpulan: Operator &amp;&amp; menghasilkan true jika kedua operand bernilai true.</li> </ul> </li> <li>4. Langkah 2: Evaluasi Operator    (OR) <ul style="list-style-type: none"> <li>• Kasus 1: Set nilai a = true dan b = false. <ul style="list-style-type: none"> <li>◦ Lakukan operasi logika a    b dan simpan hasilnya.</li> <li>◦ Catat bahwa hasilnya adalah true.</li> </ul> </li> <li>• Kasus 2: Set nilai a = false dan b = true. <ul style="list-style-type: none"> <li>◦ Lakukan operasi logika a    b dan simpan hasilnya.</li> <li>◦ Catat bahwa hasilnya adalah true.</li> </ul> </li> <li>• Kasus 3: Set nilai a = true dan b = true. <ul style="list-style-type: none"> <li>◦ Lakukan operasi logika a    b dan simpan hasilnya.</li> <li>◦ Catat bahwa hasilnya adalah true.</li> </ul> </li> <li>• Kesimpulan: Operator    menghasilkan true jika setidaknya salah satu dari operand bernilai true.</li> </ul> </li> <li>5. Langkah 3: Sintesis Kesimpulan <ul style="list-style-type: none"> <li>• Rangkum hasil evaluasi dari Langkah 1 dan Langkah 2.</li> <li>• Tuliskan kesimpulan umum tentang bagaimana masing-masing operator logika bekerja.</li> </ul> </li> <li>6. Akhiri Program</li> </ol> <p>Luaran sudah sesuai dengan program yang disusun. Tipe data yang ditampilkan telah sesuai dengan kebutuhan dan permintaan data.</p>
	<p><b>[No. 5] Kesimpulan</b></p> <p><b>1. Analisa</b></p> <ol style="list-style-type: none"> <li>a) Operator &amp;&amp; adalah lebih ketat dalam menghasilkan true, memerlukan semua kondisi yang dibandingkan untuk benar.</li> <li>b) Operator    adalah lebih longgar, hanya memerlukan salah satu kondisi benar untuk menghasilkan true.</li> </ol> <p>Analisa ini menunjukkan bagaimana kedua operator logika ini dapat digunakan secara strategis dalam pemrograman untuk mengontrol alur eksekusi berdasarkan beberapa kondisi.</p>

<b>Nama &amp; NPM</b>	<b>Topik:</b>	<b>Tanggal:</b>
<b>Ariiq Ashar Sofyan G1F02405</b>	<b>Operator Logika</b>	<b>14 September 2024</b>
<b>[No. 6] Identifikasi Masalah:</b>		
<p><b>Contoh 6:</b></p> <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><b>Latihan 6</b> Rekomendasikan apa bentuk tanda operator agar nilai = 60 memenuhi untuk Lulus !</p>		
<b>[No. 6] Analisis dan Argumentasi</b>		
<p><b>6.1 Rekomendasikan apa bentuk tanda operator agar nilai = 60 memenuhi untuk Lulus !</b> Untuk memastikan bahwa nilai 60 juga memenuhi syarat untuk "Lulus", Anda dapat mengubah kondisi operator menjadi lebih besar atau sama dengan 60. Berikut adalah modifikasi kode untuk mencapainya</p> <p><b>Input :</b></p> <pre>public class OperatorKondisi{     public static void main( String[] args ){         String status = "";         int nilai = 60;         status = (nilai &gt;= 60) ? "Lulus" : "Gagal";         System.out.println( status );     } }</pre> <p><b>Output:</b> Lulus</p>		
<b>[No. 6] Penyusunan Algoritma dan Kode Program</b>		
<p><b>6.1. Rekomendasikan apa bentuk tanda operator agar nilai = 60 memenuhi untuk Lulus !</b> <b>Algoritma</b></p> <ol style="list-style-type: none"> <li>1. Mulai program.</li> <li>2. Deklarasikan variabel status sebagai string dan nilai sebagai integer.</li> <li>3. Berikan nilai pada variabel nilai.</li> <li>4. Gunakan operator ternary untuk memeriksa apakah nilai lebih besar atau sama dengan 60:             <ul style="list-style-type: none"> <li>o Jika benar, tetapkan "Lulus" pada variabel status.</li> <li>o Jika salah, tetapkan "Gagal" pada variabel status.</li> </ul> </li> <li>5. Cetak nilai status.</li> <li>6. Selesai.</li> </ol>		

### Kode program dan luaran



The screenshot displays a Java IDE interface. The top section, titled 'Main.java', contains the following code:

```
1- public class OperatorKondisi {
2-     public static void main(String[] args) {
3-         // Deklarasi variabel
4-         String status = "";
5-         int nilai = 60; // Contoh nilai 60, dapat diganti sesuai kebutuhan
6-
7-         // Operator ternary untuk menentukan status kelulusan
8-         status = (nilai >= 60) ? "Lulus" : "Gagal";
9-
10-        // Menampilkan hasil status
11-        System.out.println(status);
12-    }
13- }
14-
```

The bottom section, titled 'Output', shows the execution result:

```
java -cp /tmp/SwoTQOqleD/OperatorKondisi
Lulus

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

Luaran sudah sesuai dengan program yang disusun. Tipe data yang ditampilkan telah sesuai dengan kebutuhan dan permintaan data.

### [No. 6] Kesimpulan

#### 2) Analisa

- a) Dalam program ini, kita menggunakan operator ternary untuk menentukan status kelulusan berdasarkan nilai yang diberikan. Dengan menggunakan operator lebih besar atau sama dengan ( $\geq$ ), kita memastikan bahwa nilai 60 atau lebih akan menghasilkan status "Lulus", sedangkan nilai di bawah 60 akan menghasilkan status "Gagal".
- b) Program ini bekerja dengan efisien untuk memeriksa kondisi sederhana seperti ini, dan cocok digunakan ketika hanya ada dua kemungkinan hasil yang ingin diperiksa, yaitu lulus atau gagal.

<b>Nama &amp; NPM</b>	<b>Topik:</b>	<b>Tanggal:</b>
<b>Ariiq Ashar Sofyan G1F02405</b>	<b>Operator Bitwise</b>	<b>14 September 2024</b>

**[No. 7] Identifikasi Masalah:**

**Contoh 7: Salin dan tempel kode program berikut ke Eclipse atau JDoodle**

```
public class OperatorBitwise {
    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 );
    } }
```

**Luaran:**

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

**Latihan 7**

Evaluasi penyebab hasil ~a = -11 ? Buktikan jawaban Anda dalam perhitungan biner!

**[No. 7] Analisis dan Argumentasi**

**7.1. Evaluasi penyebab hasil ~a = -11 ? Buktikan jawaban Anda dalam perhitungan biner!**

Untuk memahami mengapa hasil dari ~a = -11, kita perlu memahami bagaimana operator bitwise NOT (~) bekerja dalam sistem biner dan representasi bilangan negatif dalam komputer.

Bilangan negatif dalam komputer direpresentasikan menggunakan komplement dua. Langkah-langkah untuk menghitung komplement dua adalah sebagai berikut:

Representasikan bilangan positif dalam bentuk biner.

Lakukan operasi NOT pada setiap bit (mengubah 0 menjadi 1 dan 1 menjadi 0).

Tambahkan 1 pada hasilnya untuk mendapatkan representasi komplemen dua dari bilangan negatif.

Mari kita buktikan secara biner untuk bilangan  $a = 10$ :

Langkah 1: Representasi biner dari  $a = 10$

Dalam bentuk biner 8-bit, 10 adalah:

0000 1010

Langkah 2: Operasi NOT ( $\sim$ )

Operator bitwise NOT mengubah setiap bit:

0000 1010 (10)

$\sim$  -----

1111 0101

Sekarang, 1111 0101 adalah hasil dari operasi bitwise NOT.

Langkah 3: Komplemen dua

Untuk mendapatkan bilangan negatif, ini direpresentasikan dalam bentuk komplemen dua:

Langkah pertama adalah melakukan operasi NOT seperti yang sudah dilakukan.

Langkah kedua adalah menambah 1 ke hasil tersebut.

Mari kita lihat representasi biner 1111 0101. Ini sebenarnya adalah representasi dari -11 dalam komplemen dua. Kenapa?

Jika kita tambahkan 1 pada representasi 1111 0101, hasilnya adalah 1111 0110, yang merupakan representasi biner dari -10. Tapi karena kita mengoperasikan NOT pada  $a$ , kita sudah dalam bentuk negatif.

Jadi, 1111 0101 sudah merepresentasikan -11 dalam bentuk komplemen dua.

Kesimpulan:

Hasil  $\sim a = -11$  karena bitwise NOT mengubah setiap bit dari bilangan asli (positif) dan menggunakan representasi komplemen dua untuk bilangan negatif.

#### [No. 7] Penyusunan Algoritma dan Kode Program

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

###### Algoritma

1. Algoritma:
2. Masukkan bilangan  $a$ .
3. Representasikan bilangan  $a$  dalam bentuk biner 8-bit.
4. Lakukan operasi NOT pada setiap bit bilangan  $a$ .
5. Hasil dari operasi NOT merepresentasikan bilangan negatif dalam bentuk komplemen dua.
6. Jika ingin, tambahkan 1 pada hasil NOT untuk mengonfirmasi bilangan negatif sesuai dengan aturan komplemen dua.
7. Tampilkan hasil dari operasi NOT tersebut dan bandingkan dengan hasil yang diperoleh menggunakan operator  $\sim$ .

## Kode program dan luaran

```
Main.java
1- public class BitwiseNotDemo {
2-     public static void main(String[] args) {
3-         // Deklarasi variabel
4-         int a = 10;
5-
6-         // Tampilkan representasi biner dari a
7-         System.out.println("Nilai a dalam desimal: " + a);
8-         System.out.println("Nilai a dalam biner : " + Integer.toBinaryString(a));
9-
10-        // Lakukan operasi NOT pada a
11-        int notA = ~a;
12-
13-        // Tampilkan hasil operasi NOT
14-        System.out.println("Hasil dari ~a dalam desimal: " + notA);
15-        System.out.println("Hasil dari ~a dalam biner : " + Integer
16-            .toBinaryString(notA));
17-
18-        // Konfirmasi hasil: representasi desimal dari ~a sesuai dengan komplemen
19-        // dua
20-        System.out.println("Penjelasan: Nilai ~a adalah komplemen dua dari a.");
21-    }
22-}
```

### Penjelasan:

1. **Representasi biner:** Kode menggunakan `Integer.toBinaryString()` untuk mengonversi bilangan desimal ke bentuk biner.
2. **Operasi NOT:** Operator `~` melakukan operasi NOT pada setiap bit dari bilangan a.
3. **Komplemen dua:** Hasil operasi NOT adalah bilangan negatif yang dihasilkan dari komplemen dua.

Untuk nilai a = 10, output program akan menampilkan:

```
Output
java -cp ./tmp/6jsHGbJmKb/BitwiseNotDemo
Nilai a dalam desimal: 10
Nilai a dalam biner : 1010
Hasil dari ~a dalam desimal: -11
Hasil dari ~a dalam biner : 111111111111111111111111111110101
Penjelasan: Nilai ~a adalah komplemen dua dari a.

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

Dari hasil tersebut, dapat dilihat bahwa operasi bitwise NOT mengubah bilangan menjadi representasi negatif dalam komplemen dua, yang sesuai dengan hasil perhitungan `~a = -11`.

Luaran sudah sesuai dengan program yang disusun. Tipe data yang ditampilkan telah sesuai dengan kebutuhan dan permintaan data.

## [No. 7] Kesimpulan

### 3) Analisa

- a) Operasi bitwise NOT (`~`) membalik semua bit dari bilangan. Jika bilangan positif seperti a = 10 diubah menggunakan `~`, hasilnya adalah bilangan negatif.
- b) Hasil negatif ini muncul karena komputer menggunakan komplemen dua untuk merepresentasikan bilangan negatif.
- c) Dalam kasus a = 10, hasil `~a = -11` karena setelah membalik bit, bilangan yang terbentuk adalah representasi biner dari -11 dalam komplemen dua.
- d) Dengan kata lain, operator NOT mengubah bilangan positif menjadi negatif dengan membalik semua bit dan menggunakan metode komplemen dua.