

Nama & NPM	Topik:	Tanggal:
Nabila Azizah Mutiara Suriadi G1F024031	Operator	9 september 2024

[No. 1] Identifikasi Masalah:

- 1) Uraikan permasalahan dan variable

```
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));
    } }
```

- 2) **diketahui pada soal:**

- 1.1. Tambahkan baris `System.out.println("a + b = " + (a + b));` Ubahlah operator (+) dengan tanda (-, *, /, %)
- 1.2. Analisa perhitungan matematika yang terjadi!

[No.1] Analisis dan Argumentasi

- 1) Saya mengusulkan permasalahan ini dapat diatasi dengan cara memahami setiap tana fungsi yang harus digunakan apabila membuat pemrograman coding seperti (+*%/-)
- 2) Solusi yang dapat dipakai pada soal no 1 adalah dengan cara melihat catatan tentang aritmatika pada kegiatan praktikum kemarin.

[No.1] Penyusunan Algoritma dan Kode Program

- 1) Algoritma
 - a.) Mulai
 - b.) Buka wab jdoodle
 - c.) Deklarasi dan inisialisasi variable
 - d.) Tentukan perhitungan dan output
 - e.) Tampilkan hasil perhitungan
 - f.) Selesai

- 2) Kode program dan luaran

- 1.1 . Tambahkan baris `System.out.println("a + b = " + (a + b));` Ubahlah operator (+) dengan tanda (-, *, /, %)

The screenshot shows an online Java compiler interface. On the left, the code editor contains the following code:

```
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)); // penjumlahan
10=        // menampilkan hasil dari pengurangan
11=        System.out.println("a - b = " + (a - b)); // pengurangan
12=        // menampilkan hasil dari perkalian
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=    } }
```

On the right, the 'Input/Output' panel shows the output of the program:

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

At the bottom of the panel, it shows performance metrics: CPU Time: 0.07 sec(s) | Memory: 38636 kilobyte(s) | Compiled and executed in 1.275 sec(s).

a) Analisa perhitungan matematika yang terjadi

Int a = 20 dan b = 3 hal tersebut bertujuan untuk menginisialisasikan 20 dan 3 kemudian mendeklarasikan a dan b bertipe int.

system.out.println("a: " + a); = Menampilkan Nilai dari Variabel a dan b

System.out.println("a + b = " + (a + b)); = Menampilkan Hasil penjumlahan menghasilkan 23

System.out.println("a - b = " + (a - b)); = Menampilkan Hasil pengurangan menghasilkan 17

System.out.println("a * b = " + (a * b)); = Menampilkan Hasil perkalian menghasilkan 60

System.out.println("a / b = " + (a / b)); = Menampilkan Hasil Pembagian menghasilkan 6

System.out.println("a % b = " + (a % b)); = Menampilkan Hasil sisa bagi menghasilkan 2.

b) Uraikan luaran yang dihasilkan

a: 20

b: 3

a + b = 23

a - b = 17

a * b = 60

a / b = 6

a % b = 2

setiap barisan output menghasilkan nilai yang berbeda beda.

(Tuliskan penjelasan dari program yang dibuat, apakah kode dan luaran sudah benar?)

penjelasan: kode dan luaran program telah benar Kode ini dengan tepat mendeklarasikan variabel, melakukan operasi aritmatika, dan menampilkan hasilnya.

[NO.1 KESIMPULAN]

Dengan memahami cara kerja masing-masing operator, saya bisa lebih memahami bagaimana operasi matematika dilakukan dalam kode Java dan hasil yang dihasilkan untuk berbagai operasi Berikut kode java untuk perhitungan operasi matematika yang saya pelajari.

- 1) Penjumlahan (+): Menghasilkan jumlah dari kedua angka.
- 2) Pengurangan (-): Menghasilkan selisih antara kedua angka.
- 3) Perkalian (*): Menghasilkan hasil perkalian dari kedua angka.
- 4) Pembagian (/): Menghasilkan hasil bagi dari pembagian kedua angka, dibulatkan ke bawah dalam pembagian integer.
- 5) Modulo (%): Menghasilkan sisa dari pembagian kedua angka.

[No. 2] Identifikasi Masalah:

1) Uraikan permasalahan dan variable

```
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);  
    }  
}
```

Diketahui pada soal :

2.1. Bandingkan hasil Contoh 1 dengan Contoh 2!

[No2] Analisis dan Argumentasi

- 1) Saya mengusulkan permasalahan ini dapat diatasi dengan cara Menggunakan operator penugasan yang tepat untuk memodifikasi nilai variabel secara bertahap dan memeriksa hasil setiap operasi secara terpisah
- 2) Alasan solusi ini karena mengurangi kemungkinan kesalahan dan meningkatkan keterbacaan kode.
- 3) Perbaiki kode program dengan cara menambahkan komentar pada setiap baris kode untuk menjelaskan tujuan dari setiap operator penugasan, mengelompokkan output berdasarkan operasi untuk membuat hasil lebih jelas dan terstruktur.

[No.2] Penyusunan Algoritma dan Kode Program

- 1) Algoritma
 - a) Mulai
 - b) Masukkan kode input pada contoh satu ke jdoodle
 - c) Catat hasil output pada contoh Satu
 - d) Lakukan hal yang sama, salin kode pemrograman pada contoh 2
 - e) Catat hasil output pada contoh 2
 - f) Analisis perbedaan
 - g) Selesai
- 2) Kode program dan luaran
Contoh 1

```

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)); // penjumlahan
10        // menampilkan hasil dari pengurangan
11        System.out.println("a - b = " + (a - b)); // pengurangan
12        // menampilkan hasil dari perkalian
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    }
}

```

Input/Output

Language Version: JDK 21.0.0 ☐ Interactive

Input Arguments

Stdin Inputs

Output Generated Files

```

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

```

CPU Time: 0.07 sec(s) | Memory: 38636 kilobyte(s) | Compiled and executed in 1.275 sec(s)

Contoh 2:

```

1 public class OperatorPenggunaan {
2     public static void main(String[] args) {
3         // deklarasi nilai
4         int a = 20, b = 3;
5         //operator penggunaan
6         b += a;
7         System.out.println("Penambahan : " + b);
8
9         // pengurangan
10        b -= a;
11        System.out.println("Pengurangan : " + b);
12
13        // perkalian
14        b *= a;
15        System.out.println("Perkalian : " + b);
16
17        // Pembagian
18        b /= a;
19        System.out.println("Pembagian : " + b);
20
21        // Sisa bagi
22        b %= a;
23        // sekarang b=0
24        System.out.println("Sisa Bagi: " + b);
25    }
26 }

```

Want me to improve your code's performance?

Input/Output

Language Version: JDK 21.0.0 ☐ Interactive

Input Arguments

Stdin Inputs

Output Generated Files

```

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

```

CPU Time: 0.06 sec(s) | Memory: 38776 kilobyte(s) | Compiled and executed in 1.209 sec(s)

a) Beri komentar pada kode yang di Screenshot dan analisis luaran

1. Pertambahan

Contoh 1: $a + b = 23$

Contoh 2: penambahan = 23

Perbandingan: Hasilnya sama. Di contoh 1, $a + b$ menunjukkan hasil penjumlahan awal.

Di Contoh 2, nilai b diubah untuk menyertakan penjumlahan $b += a$.

2. Pengurangan

Contoh 1: $a - b = 17$

Contoh 2: pengurangan = 3

Perbandingan: Hasil berbeda. Di contoh 1, $a - b$ menghasilkan 17 sementara contoh 2 menghasilkan 3

3. Perkalian

Contoh 1: $a * b = 60$

Contoh 2: perkalian = 60

Perbandingan: Hasilnya sama. Di contoh 1, $a * b$ menunjukkan perkalian awal. Di Contoh 2, b dikalikan dengan a, dan hasilnya sama

4. Pembagian

Contoh 1: $a / b = 6$

Contoh 2: pembagian = 3

Perbandingan: . Di Contoh 1, a / b yang menghasilkan 6. Di contoh 2, b dibagi oleh a yang hasilnya 3

5. Sisa bagi

Contoh 1: $a \% b = 2$

Contoh 2: sisa bagi: 3

Perbandingan: Hasil berbeda, pada contoh satu $a \% b = 2$, sementara pada contoh 2, perbandingan = 3.

b) Analisis luaran

Luaran sudah sesuai dengan program yang disusun. Tipe data yang ditampilkan telah sesuai dengan kebutuhan dan permintaan data, output yang dikeluarkan keluar dengan sukses tanpa teridentifikasi error pada pemrograman.

[No.2] Kesimpulan

evaluasi: dapat disimpulkan hasil akhir menunjukkan bagaimana setiap operasi aritmatika mempengaruhi variabel, dengan nilai akhir yang berbeda tergantung pada urutan dan jenis operasi yang diterapkan. Pemahaman yang baik tentang bagaimana operator penugasan bekerja membantu dalam menulis kode yang lebih efisien dan dalam menganalisis hasil dari operasi matematika dalam pemrograman.

[No. 3] Identifikasi Masalah:

2) Uraikan permasalahan dan variable

```
public class OperatorRealasional {
    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);
    }
}
```

Diketahui pada soal :

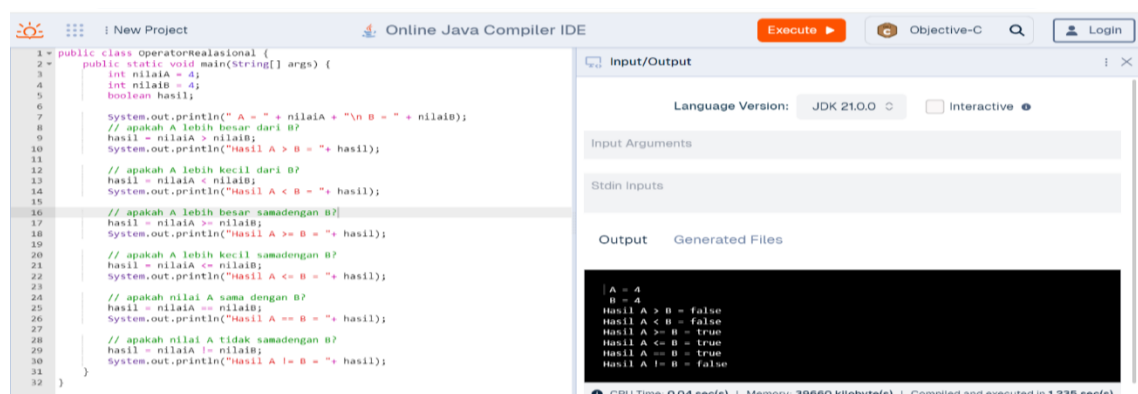
- 3.1. Ubahlah nilai A = 4 dan B = 4. Analisa perubahan yang terjadi!
- 3.2 Bandingkan bagaimana perbedaan nilai A dan B mempengaruhi nilai luaran!

[No.3] Analisis dan Argumentasi

- 1) Saya mengusulkan permasalahan ini dapat diatasi dengan cara mengubah nilai variabel nilaiA dan nilaiB menjadi 4, dan kemudian menjalankan kembali program untuk melihat bagaimana hasil perbandingan berubah.
- 2) Alasan solusi ini karena dengan mengubah nilai kedua variabel menjadi sama (4), kita dapat mengamati perubahan hasil dari operasi relasional yang memeriksa kesamaan dan perbedaan antara kedua nilai. Hal ini membantu dalam memahami bagaimana hasil perbandingan dapat berubah tergantung pada nilai variabel yang dibandingkan

[No.3] Penyusunan Algoritma dan Kode Program

- 1) Algoritma
 - a) Mulai
 - b) Inisialisasikan variable
 - c) Tampilkan (nilaiA dan nilaiB)
 - d) Lakukan perbandingan (a > b, a < b, dsb)
 - e) selesai
- 2) Kode program dan luaran



The screenshot displays an online Java compiler interface. On the left, the source code for a class named `OperatorRelasional` is shown. The code defines a `main` method that initializes `nilaiA` and `nilaiB` to 4, then prints the results of various relational operations: `A > B`, `A < B`, `A >= B`, `A <= B`, `A == B`, and `A != B`. On the right, the 'Input/Output' panel shows the program's execution results. The output confirms that for `A = 4` and `B = 4`, the results are: `A > B = false`, `A < B = false`, `A >= B = true`, `A <= B = true`, `A == B = true`, and `A != B = false`. The status bar at the bottom indicates a CPU time of 0.04 seconds and memory usage of 39660 kilobytes.

```
1 public class OperatorRelasional {
2     public static void main(String[] args) {
3         int nilaiA = 4;
4         int nilaiB = 4;
5         boolean hasil;
6
7         System.out.println("A = " + nilaiA + "\n B = " + nilaiB);
8         // apakah A lebih besar dari B?
9         hasil = nilaiA > nilaiB;
10        System.out.println("Hasil A > B = " + hasil);
11
12        // apakah A lebih kecil dari B?
13        hasil = nilaiA < nilaiB;
14        System.out.println("Hasil A < B = " + hasil);
15
16        // apakah A lebih besar samadengan B?
17        hasil = nilaiA >= nilaiB;
18        System.out.println("Hasil A >= B = " + hasil);
19
20        // apakah A lebih kecil samadengan B?
21        hasil = nilaiA <= nilaiB;
22        System.out.println("Hasil A <= B = " + hasil);
23
24        // apakah nilai A sama dengan B?
25        hasil = nilaiA == nilaiB;
26        System.out.println("Hasil A == B = " + hasil);
27
28        // apakah nilai A tidak samadengan B?
29        hasil = nilaiA != nilaiB;
30        System.out.println("Hasil A != B = " + hasil);
31    }
32 }
```

Input/Output

Language Version: JDK 21.0.0 ☐ Interactive

Input Arguments

Stdin Inputs

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
```

CPU Time: 0.04 sec(s) | Memory: 39660 kilobyte(s) | Compiled and executed in 1.235 sec(s)

- a) Beri komentar pada kode yang di Screenshot dan analisis luaran Bandingkan bagaimana perbedaan nilai A dan B mempengaruhi nilai luaran! Ketika nilaiA dan nilaiB berbeda, hasil dari operasi relasional akan sesuai dengan perbandingan nilai-nilai tersebut. Sebagai contoh:
 1. Ketika nilaiA lebih besar dari nilaiB, maka `A > B` bernilai true dan sebaliknya untuk `A < B`.
 2. Ketika nilaiA sama dengan nilaiB, maka `A == B` bernilai true dan `A != B` bernilai false.
 3. Jika nilai-nilai tersebut tidak sama, maka sebaliknya terjadi.

Perubahan nilai A dan B mempengaruhi hasil dari operator relasional yang digunakan, yang mencerminkan bagaimana perbandingan antar nilai dievaluasi.

- b) Analisis luaran
Luaran sudah sesuai dengan program yang disusun sehingga luaran dapat dihasilkan sebagai berikut:

- Nilai awal: A = 12, B = 4
- Hasil perbandingan:
 1. A > B: true
 2. A < B: false
 3. A >= B: true
 4. A <= B: false
 5. A == B: false
 6. A != B: true

Program ini melakukan beberapa perbandingan relasional antara dua nilai integer (nilaiA dan nilaiB), dan hasil dari setiap perbandingan dicetak ke layar

[NO.3] KESIMPULAN

1. Evaluasi:

Perbandingan dengan Nilai Sama:

Jika nilaiA dan nilaiB sama, hasil dari perbandingan dengan operator >, <, dan != adalah false, sementara operator >=, <=, dan == menghasilkan true.

Pengaruh Operator Relasional:

Operator > dan < tidak cocok jika nilai sama. Operator >=, <=, dan == cocok jika nilai sama. != menunjukkan ketidaksetaraan jika nilai berbeda.

Kesimpulannya, hasil operasi relasional tergantung pada kesamaan atau perbedaan nilai yang dibandingkan, yang mempengaruhi hasil dari perbandingan dalam kode program

[No. 4] Identifikasi Masalah:

3) Uraikan permasalahan dan variable

```
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);
}
}

```

Diketahui dari soal :

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

[No.4] Analisis dan Argumentasi

- 1) Saya mengusulkan permasalahan ini dapat diatasi dengan cara menjelaskan perbedaan antara post dan pre increment serta decrement dalam dokumentasi kode atau komentar yang lebih rinci.
- 2) Alasan solusi ini karena penggunaan operator increment dan decrement sering membingungkan bagi pemula
- 3) Perbaiki kode program dengan cara menambahkan komentar yang lebih jelas dan mendetail di setiap bagian kode, menambahkan komentar di bagian output untuk menjelaskan apa yang sedang ditampilkan dan mengapa

[No.4] Penyusunan Algoritma dan Kode Program

- 3) Algoritma
 1. Mulai
 2. Inisialisasi dan tampilkan Post Increment
 3. Inisialisasi dan Tampilkan Pre Increment
 4. Inisialisasi dan Tampilkan Post Decrement
 5. Inisialisasi dan Tampilkan Pre Decrement
 6. selesai
- 4) Kode program dan luaran


```

public class operator {
    public static void main(String[] args) {
        // inisiatng variabel a dengan nilai 10
        int a = 10;
        // harder untuk post increment
        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();

        // inisiatng variabel b dengan nilai 10
        int b = 10;
        // harder untuk pre increment
        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();

        // inisiatng variabel c dengan nilai 10
        int c = 10;
        // harder untuk post decrement
        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();

        // inisiatng variabel d dengan nilai 10
        int d = 10;
        // harder untuk pre decrement
        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);
    }
}

```

Input/Output

```

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

```

CPU Time: 0.05 sec(s) | Memory: 38764 kilobyte(s) | Compiled and executed in 1.496 sec(s)

- Screenshot/ Capture potongan kode dan hasil luaran
- Analisis luaran dan komentar pada kode

```

# 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 Increment (a++):
Isi variabel a: 10: Nilai awal a adalah 10.
Isi variabel a: 10: Nilai a dicetak sebelum diinkremen, sehingga tetap 10.
Isi variabel a: 11: Nilai a dicetak setelah diinkremen, sehingga menjadi 11.
- Pre Increment (++b):
Isi variabel b: 10: Nilai awal b adalah 10.
Isi variabel b: 11: Nilai b diinkremen menjadi 11 sebelum dicetak.
Isi variabel b: 11: Nilai b setelah diinkremen tetap 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

```

- Post Decrement (c--):
Isi variabel c: 10: Nilai awal c adalah 10.
Isi variabel c: 10: Nilai c dicetak sebelum dikurangkan, sehingga tetap 10.
Isi variabel c: 9: Nilai c dicetak setelah dikurangkan, sehingga menjadi 9.
- Pre Decrement (--d):
Isi variabel d: 10: Nilai awal d adalah 10.
Isi variabel d: 9: Nilai d dikurangkan menjadi 9 sebelum dicetak.
Isi variabel d: 9: Nilai d setelah dikurangkan tetap 9

- Analisa luaran yang dihasilkan
 Tipe data yang ditampilkan telah sesuai dengan kebutuhan dan permintaan data.

Semua hasil keluaran sesuai dengan ekspektasi dan menjelaskan dengan benar bagaimana operator increment dan decrement (post dan pre) bekerja dalam Java. Program ini menunjukkan perubahan nilai variabel sebelum dan setelah operasi increment/decrement dengan jelas.

[No.4] Kesimpulan

Analisis: Kode program berhasil menunjukkan perbedaan antara post dan pre increment serta decrement. Post increment/decrement menampilkan nilai sebelum operator diterapkan, sedangkan pre increment/decrement menampilkan nilai setelah operator diterapkan.

Untuk meningkatkan pemahaman dan pemeliharaan kode, disarankan untuk menambahkan komentar yang lebih jelas dan mengganti nama kelas agar sesuai dengan konvensi penamaan Java.

Keputusan ini diambil berdasarkan prinsip-prinsip pemrograman yang baik dan kebutuhan untuk memastikan bahwa kode mudah dipahami oleh orang lain, serta mendukung pembelajaran dan pengembangan yang efektif.

[No. 5] Identifikasi Masalah:

- 1) Uraikan permasalahan dan variable

```
public class OperatorLogika {  
    public static void main (String [] args) {  
        boolean a = true;  
        boolean b = false;  
        boolean c;  
        c = a && b;  
        System.out.println("true && false = " +c);  
    } }
```

diketahui pada soal:

- 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!

[No.5] Analisis dan Argumentasi

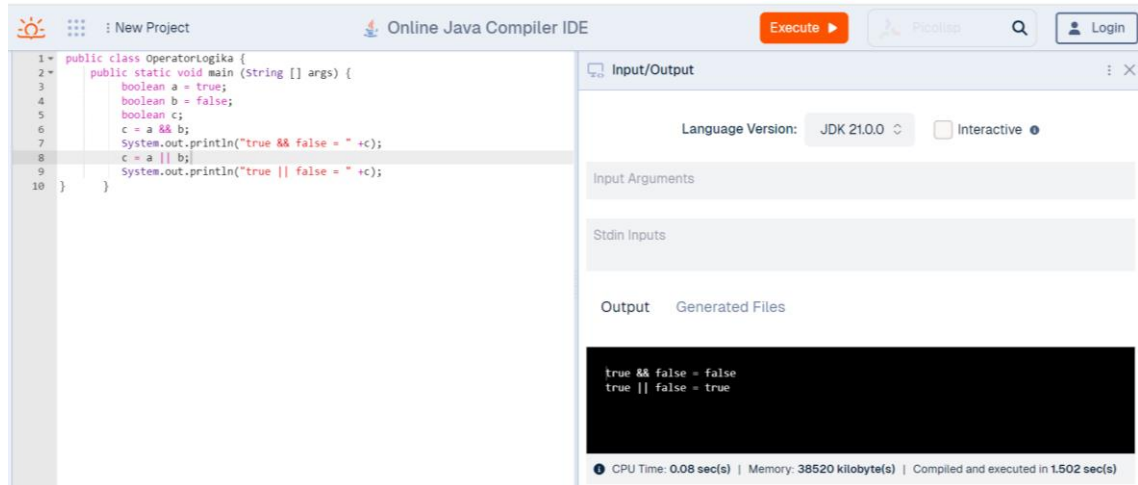
- 1) Kode ini berhasil mendemonstrasikan penggunaan operator logika && dalam Java, dengan hasil yang sesuai dengan aturan logika boolean. Hasil evaluasi dari true && false adalah false, dan ini tercermin dengan benar dalam output program
- 2) Keputusan untuk menulis kode ini berdasarkan pada prinsip logika boolean dan praktik terbaik pemrograman Java. Kode ini secara efektif mendemonstrasikan bagaimana operator && berfungsi dan memberikan hasil yang sesuai dengan aturan logika, sekaligus membantu pembaca memahami konsep logika boolean dengan cara yang sederhana dan jelas.

[No.5] Penyusunan Algoritma dan Kode Program

- 1) Algoritma
 - a) Mulai
 - b) Inisialisasi variable
 - c) Operasi logika
 - d) Output hasil
 - e) Selesai.

5) Kode program dan keluaran

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



The screenshot shows an online Java compiler interface. The code editor on the left contains the following Java code:

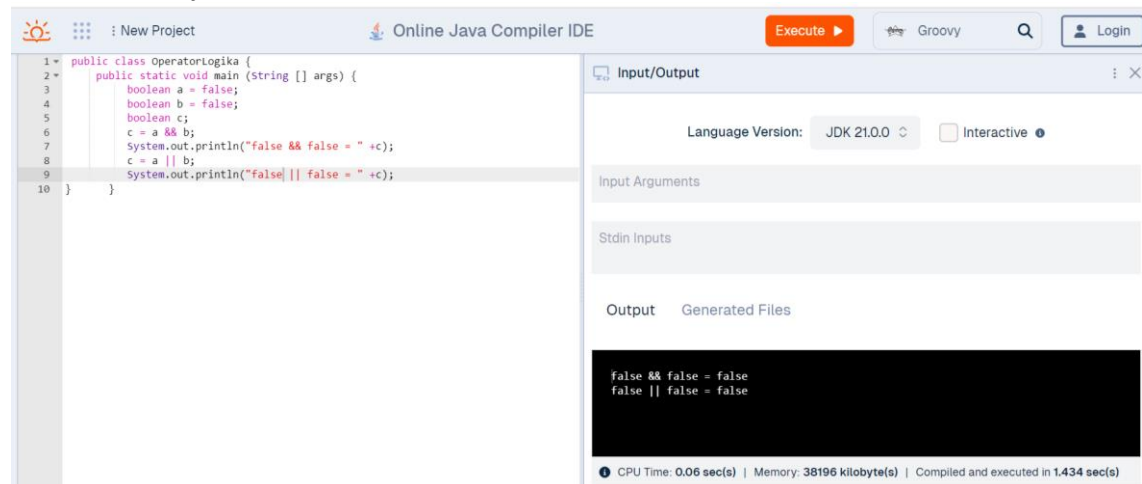
```
1 public class OperatorLogika {  
2     public static void main (String [] args) {  
3         boolean a = true;  
4         boolean b = false;  
5         boolean c;  
6         c = a && b;  
7         System.out.println("true && false = " + c);  
8         c = a || b;  
9         System.out.println("true || false = " + c);  
10    }  
}
```

The right-hand side of the interface shows the 'Input/Output' panel. It indicates the language version is 'JDK 21.0.0' and has an 'Interactive' toggle. The 'Output' tab is selected, displaying the following results:

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

At the bottom of the output panel, performance metrics are shown: 'CPU Time: 0.08 sec(s) | Memory: 38520 kilobyte(s) | Compiled and executed in 1.502 sec(s)'.

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



The screenshot shows the same online Java compiler interface as before, but with the code modified. The code editor now contains:

```
1 public class OperatorLogika {  
2     public static void main (String [] args) {  
3         boolean a = false;  
4         boolean b = false;  
5         boolean c;  
6         c = a && b;  
7         System.out.println("false && false = " + c);  
8         c = a || b;  
9         System.out.println("false || false = " + c);  
10    }  
}
```

The 'Output' panel on the right shows the results of the execution:

```
false && false = false  
false || false = false
```

The performance metrics at the bottom are: 'CPU Time: 0.06 sec(s) | Memory: 38196 kilobyte(s) | Compiled and executed in 1.434 sec(s)'.

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

The screenshot shows an online Java IDE with the following code in the editor:

```

1 public class operatorLogika {
2     public static void main (String [] args) {
3         boolean a = true; //atau false untuk percobaan lainnya
4         boolean b = false; //atau true untuk percobaan lainnya
5         boolean c;
6         // evaluasi pertanyaan
7         c = a || b && a || !b;
8         System.out.println("a || b && a || !b = " + c);
9     }
}

```

The right-hand pane shows the 'Input/Output' section with the following details:

- Language Version: JDK 21.0.0
- Interactive: ☐
- Input Arguments: (empty)
- Stdin Inputs: (empty)
- Output:

```
a || b && a || !b = true
```
- Generated Files: (empty)
- Performance: CPU Time: 0.04 sec(s) | Memory: 38508 kilobyte(s) | Compiled and executed in 1.655 sec(s)

a) Screenshot/ Capture potongan kode dan hasil luaran Beri komentar pada kode yang di Screenshot

Pada 5.1

Penjelasan:

1. `a && b` akan menghasilkan false karena salah satu operand (b) adalah false.
2. `a || b` akan menghasilkan true karena salah satu operand (a) adalah true.

Pada 5.2

Penjelasan:

1. `false && false` akan menghasilkan false karena kedua operand adalah false.
2. `false || false` akan menghasilkan false karena kedua operand adalah false.

Pada 5.3

Penjelasan:

Jika `a = true` dan `b = false`:

1. `!b` adalah true (karena b adalah false).
2. `b && a` adalah false (karena b adalah false).
3. `a || false || true = true || true = true`.

Jadi, hasil dari `a || b && a || !b` adalah true jika a adalah true dan b adalah false.

Jika `a = false` dan `b = false`

Maka, , hasil dari `a || b && a || !b` adalah true jika a adalah false dan b adalah false.

b) Uraikan luaran yang dihasilkan

setiap barisan output menghasilkan nilai yang berbeda beda.

(Tuliskan penjelasan dari program yang dibuat, apakah kode dan luaran sudah benar?)

penjelasan : luaran dari program sudah benar dikarenakan menyertai Langkah Langkah yang sesuai,

1. Variabel a diset ke true dan b diset ke false.
2. Operasi logika AND antara true dan false menghasilkan false.
3. Hasil false dicetak bersama dengan string "true && false = ".

Menghasilkan output yang sesuai dengan ekspektasi.

[NO.5 KESIMPULAN]

Evaluasi: Skenario pemrograman ini memiliki konsekuensi positif dalam hal pemahaman operator logika, efisiensi eksekusi, debugging, pendidikan, dan kepatuhan terhadap

praktik terbaik. Memahami dan menggunakan operator logika dengan benar adalah keterampilan penting dalam pemrograman yang memengaruhi efektivitas dan kualitas kode secara keseluruhan.

[No. 6] Identifikasi Masalah:

- 1) Uraikan permasalahan dan variable

```
public class OperatorKondisi{  
    public static void main( String[] args ){  
        String status = "";  
        int nilai = 80;  
        status = (nilai > 60)?"Lulus":"Gagal";  
        System.out.println( status );  
    } }  
}
```

Diketahui dari soal :

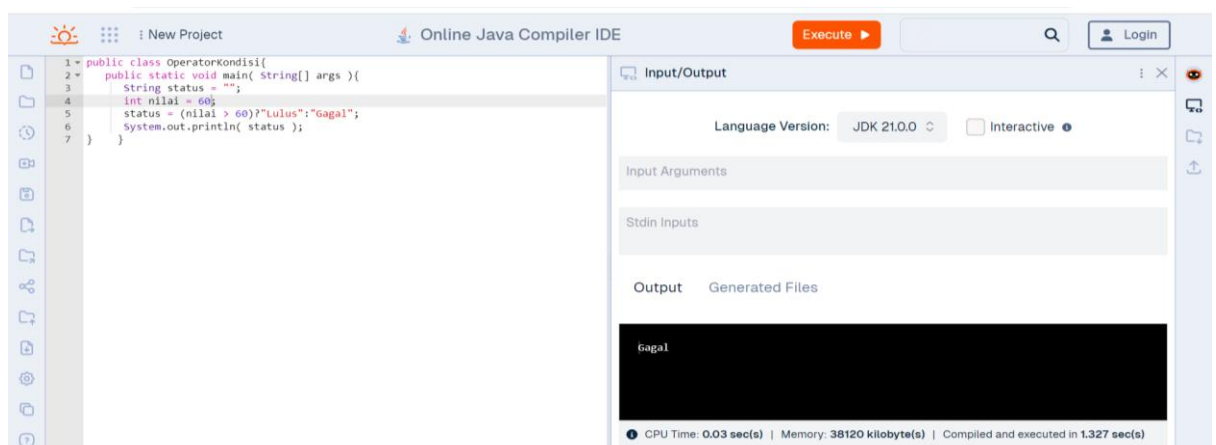
Berdasarkan Contoh 6, ubahlah nilai = 60. Analisis hasil dan proses yang terjadi!

[No.1] Analisis dan Argumentasi

- 1) Saya mengusulkan permasalahan ini dapat diatasi dengan cara menambahkan penanganan kasus untuk nilai batas dalam ekspresi kondisional dan memberikan komentar atau dokumentasi yang jelas.
- 2) Alasan solusi ini karena dengan menambahkan penanganan khusus untuk nilai batas seperti 60 dan memberikan dokumentasi yang jelas, kita dapat memastikan bahwa hasil evaluasi kondisi benar-benar sesuai dengan yang diharapkan. Ini juga membantu mencegah kebingungan tentang bagaimana kondisi tertentu mempengaruhi hasil.

[No.1] Penyusunan Algoritma dan Kode Program

- 6) Algoritma
 - (a) Mulai
 - (b) Inisialisasi variable
 - (c) Evaluasi kondisi
 - (d) Tampilkan hasil
 - (e) selesai
- 7) Kode program dan luaran



a) Screenshot/ Capture potongan kode dan hasil luaran

Beri komentar pada kode yang di Screenshot

1. Inisialisasikan variable

String status= " "; Mendeklarasikan variabel status dengan tipe data String dan memberinya nilai awal berupa string kosong.

2. Int nilai 60; : Mendeklarasikan variabel nilai dengan tipe data int dan memberinya nilai 60.

3. Dalam kasus ini, nilai adalah 60. Jadi, kondisi nilai > 60 adalah false (karena 60 tidak lebih besar dari 60). Maka akan menghasilkan output GAGAL.

b) Analisa luaran yang dihasilkan

Tipe data yang ditampilkan telah sesuai dengan kebutuhan dan permintaan data

Proses yang terjadi adalah:

1. Variabel nilai diatur menjadi 60.

2. Operator ternary mengevaluasi ekspresi nilai > 60. Karena nilai adalah 60, ekspresi ini false.

3. Karena ekspresinya adalah false, nilai status diatur ke "Gagal".

4. Program kemudian mencetak "Gagal" ke layar.

5. Hal tersebut sesuai dengan data yang di inginkan.

[NO.6 KESIMPULAN]

Evaluasi: Program ini efektif dalam mendemonstrasikan penggunaan operator ternary untuk evaluasi kondisi. Hasil dari operasi kondisional sesuai dengan logika yang diterapkan: jika nilai lebih besar dari 60, status adalah "Lulus"; jika tidak, status adalah "Gagal".

Keputusan untuk menangani nilai batas secara eksplisit dan menambahkan komentar yang jelas memastikan bahwa kode tidak hanya berfungsi dengan benar tetapi juga mudah dipahami dan dipelihara. Ini mendukung praktik pengkodean yang baik dan membantu menjaga kualitas serta keakuratan program.

[No. 7] Identifikasi Masalah:

1) Uraikan permasalahan dan variable

```
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 );
} }

```

Diketahui dari soal :

Pilihlah 3 perhitungan Contoh 7, kemudian uraikan perhitungan biner! Simpulkan hasilnya!

[No.7] Analisis dan Argumentasi

- 1) Saya mengusulkan permasalahan ini dapat diatasi dengan cara menyediakan dokumentasi dan komentar yang jelas dalam kode untuk menjelaskan bagaimana setiap operasi bitwise dan bit shifting dilakukan serta memberikan pemahaman tentang hasilnya
- 2) Alasan solusi ini karena Ini sangat penting untuk pemeliharaan kode dan memastikan bahwa setiap bagian kode mudah dipahami.
- 3) Perbaiki kode program dengan cara menambahkan komentar yang Jelas dan menampilkan Hasil dalam Kontek

[No.7] Penyusunan Algoritma dan Kode Program

- 1) Algoritma
 1. Mulai.
 2. Input bilangan a dan b
 3. Align: Pastikan kedua bilangan biner memiliki panjang yang sama
 4. Tampilkan
 5. selesai
- 2) Kode program dan luaran

The screenshot shows a Java IDE with a code editor on the left and an 'Input/Output' panel on the right. The code in the editor is as follows:

```

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 }

```

The 'Input/Output' panel on the right shows the following output:

```

Hasil dari a & b : 2
Hasil dari a | b : 15
Hasil dari a ^ b : 13

```

- a) Screenshot/ Capture potongan kode dan hasil luaran Beri komentar pada kode yang di Screenshot

1. Operasi Bitwise AND (a & b)

Langkah-langkah:

- Nilai a: 10
- Nilai b: 7

Pertama, ubah nilai a dan b ke dalam bentuk biner:

- a = 10 dalam biner adalah 1010
- b = 7 dalam biner adalah 111

Kemudian, operasi AND bitwise:

Hasil dari a & b adalah 2.

2. Operasi Bitwise OR (a | b)

Langkah-langkah:

- Nilai a: 10
- Nilai b: 7

Seperti sebelumnya, gunakan bentuk biner dari a dan b:

- a = 10 dalam biner adalah 1010
- b = 7 dalam biner adalah 111

Hasil dari a | b adalah 15.

3. Operasi Bitwise XOR (a ^ b)

Langkah-langkah:

- Nilai a: 10
- Nilai b: 7

Gunakan bentuk biner dari a dan b:

- a = 10 dalam biner adalah 00001010
- b = 7 dalam biner adalah 00000111

Hasil dari a ^ b adalah 13

- b) Analisa luaran yang dihasilkan

Tipe data yang ditampilkan telah sesuai dengan kebutuhan dan permintaan data.

Ketiga operasi bitwise (&, |, ^) memberikan hasil yang berbeda tergantung pada bagaimana bit-bit dari kedua angka dibandingkan:

- 1) (&): Menghasilkan 2 karena hanya ada bit yang sama dengan 1 pada posisi yang sama.
- 2) (|): Menghasilkan 15 karena semua bit setidaknya ada satu 1 pada posisi yang sama.

3) (^): Menghasilkan 13 karena menghasilkan 1 pada posisi bit yang berbeda.

[No.7] Kesimpulan

analisis: Kode program ini secara efektif menunjukkan bagaimana berbagai operasi bitwise dan bit shifting bekerja pada bilangan bulat. Hasil yang diperoleh sesuai dengan prinsip dasar dari operasi bitwise (AND, OR, XOR, NOT) dan bit shifting (right shift, left shift). Hasil yang diperoleh dari program sesuai dengan teori operasi bitwise dan bit shifting.

Refleksi

Secara keseluruhan, penggunaan JDoodle dalam pembelajaran pemrograman telah memberikan wawasan yang mendalam dan alat yang berguna untuk mengatasi berbagai tantangan dalam kode.