NAMA : TIARA YEMELDA

NPM : G1F021008

[Latihan 1]

Identifikasi Masalah:

1. Soal meminta untuk mengubah operator di dalam pernyataan `System.out.println("a + b = " + (a + b));` dengan operator lain (-, , /, %).
2. Variabel yang digunakan adalah `a` dan `b`, dengan nilai `a = 20` dan `b = 3`.

Analisis dan Argumentasi:

1) Masalah ini dapat diatasi dengan cara mengubah operator sesuai instruksi dan melihat hasil dari masing-masing perhitungan matematika.
2) Alasan solusi ini adalah untuk memahami bagaimana operator matematika bekerja di Java dan apa efeknya pada variabel.
3) Perbaikan kode program dilakukan dengan menambahkan perhitungan menggunakan operator lain, yaitu `-`, ` `, `/`, dan `%`.

Penyusunan Algoritma dan Kode Program:

1) Algoritma :

- Deklarasikan variabel `a` dan `b`.

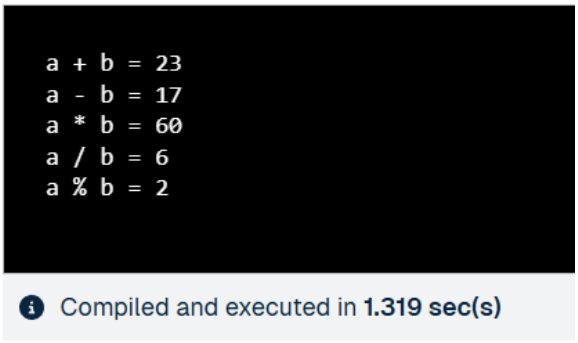
- Gunakan berbagai operator aritmatika pada `a` dan `b`.

- Cetak hasil dari setiap operasi.

2) Kode program dan luaran :

```java
public class OperatorAritmatika{
   public static void main(String[] args)  {
       int a = 20, b = 3;
       System.out.println("a + b = " + (a + b));
       System.out.println("a - b = " + (a - b));
       System.out.println("a * b = " + (a * b));
       System.out.println("a / b = " + (a / b));
       System.out.println("a % b = " + (a % b));
   }
}
```

Luaran:



Analisis Luaran:

Hasil operasi sesuai dengan yang diharapkan:

- Penjumlahan: 20 + 3 = 23.

- Pengurangan: 20 - 3 = 17.

- Perkalian: 20 3 = 60.

- Pembagian: 20 / 3 = 6 (hasil bagi integer).

- Sisa bagi: 20 % 3 = 2 (sisa pembagian).

Kesimpulan:

1) Analisa :

- Pada program ini, operator aritmatika bekerja dengan benar sesuai dengan definisinya.

- Operator pembagian pada tipe data integer hanya memberikan hasil bilangan bulat, tanpa sisa desimal.

- Operator modulus (%) memberikan hasil berupa sisa dari pembagian dua bilangan.

[Latihan 2]

Identifikasi Masalah:

1) Soal meminta untuk membandingkan hasil dari Operator Penugasan dengan Operator Aritmatika .

2) Variabel yang digunakan adalah `a = 20` dan `b = 3`.

Analisis dan Argumentasi:

1) Masalah ini dapat diatasi dengan membandingkan bagaimana operator penugasan mengubah nilai variabel dibandingkan dengan operator aritmatika.
2) Alasan solusi ini adalah untuk melihat perbedaan antara operator yang mengubah nilai variabel secara langsung (penugasan) dengan operator yang hanya melakukan operasi matematika.
3) Perbaikan kode program dilakukan dengan menggunakan operator penugasan pada variabel `b`.

Penyusunan Algoritma dan Kode Program:

1) Algoritma :

- Deklarasikan variabel `a` dan `b`.

- Gunakan operator penugasan untuk memodifikasi nilai `b`.

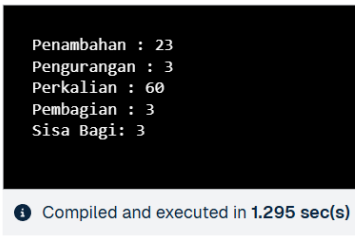
- Cetak hasil setelah setiap operasi.

2) Kode program dan luaran :

```java
public class OperatorPenugasan {
    public static void main(String[] args) {
        int a = 20, b = 3;
        b += a;
        System.out.println("Penambahan : " + b);
        b -= a;
        System.out.println("Pengurangan : " + b);
        b *= a;
        System.out.println("Perkalian : " + b);
        b /= a;
        System.out.println("Pembagian : " + b);
        b %= a;
        System.out.println("Sisa Bagi: " + b);
    }
}
```

Luaran:

Output Generated Files



Analisis Luaran:

- Hasil operasi menggunakan operator penugasan menghasilkan nilai `b` yang dimodifikasi langsung berdasarkan operasi dengan `a`.

- Hasil ini berbeda dari operator aritmatika yang hanya menghitung nilai tanpa memodifikasi variabel.

Kesimpulan:

1) Analisa :

- Operator penugasan mengubah nilai variabel setelah operasi dilakukan, sedangkan operator aritmatika hanya mengembalikan hasil tanpa mengubah nilai variabel yang terlibat.

- Operator penugasan lebih efisien ketika kita perlu memperbarui nilai variabel secara langsung.

[Latihan 3]

Identifikasi Masalah:

1) Soal meminta untuk mengubah nilai A dan B menjadi 4, lalu menganalisis perubahan yang terjadi.
2) Variabel yang digunakan adalah `nilaiA = 4` dan `nilaiB = 4`.

Analisis dan Argumentasi:

1) Masalah ini dapat diatasi dengan mengganti nilai `nilaiA` dan `nilaiB` menjadi 4 dan melihat hasil dari operator relasional.

2) Alasan solusi ini adalah untuk melihat bagaimana perubahan nilai variabel mempengaruhi hasil dari perbandingan logika.
3) Perbaikan kode program dilakukan dengan mengubah nilai variabel menjadi 4.

Penyusunan Algoritma dan Kode Program:

1) Algoritma :

- Deklarasikan variabel `nilaiA` dan `nilaiB` dengan nilai yang sama.

- Gunakan operator relasional untuk membandingkan kedua variabel.

2) Kode program dan luaran :

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

        System.out.println("A = " + nilaiA + "\nB = " + nilaiB);
        hasil = nilaiA > nilaiB;
        System.out.println("Hasil A > B = " + hasil);
        hasil = nilaiA < nilaiB;
        System.out.println("Hasil A < B = " + hasil);
        hasil = nilaiA >= nilaiB;
        System.out.println("Hasil A >= B = " + hasil);
        hasil = nilaiA <= nilaiB;
        System.out.println("Hasil A <= B = " + hasil);
        hasil = nilaiA == nilaiB;
        System.out.println("Hasil A == B = " + hasil);
        hasil = nilaiA != nilaiB;
        System.out.println("Hasil A != B = " + hasil);
    }
}
```

Luaran:

Output Generated Files

```
A = 4
B = 4
Hasil A > B = false
Hasil A < B = false
Hasil A >= B = true
Hasil A <= B = true
Hasil A == B = true
Hasil A != B = false
```

Compiled and executed in **1.379 sec(s)**

Analisis Luaran:

- Semua hasil yang melibatkan perbandingan `==` atau `<=`, `>=` adalah `true` karena nilai `A` dan `B` sama.

- Perbandingan yang melibatkan `>` atau `<` adalah `false` karena tidak ada perbedaan antara nilai `A` dan `B`.

Kesimpulan:

1) Analisa :

- Saat nilai `A` dan `B` sama, hasil dari perbandingan logika menyesuaikan dengan kondisi tersebut, sehingga perbandingan `==` dan `>=`, `<=` akan memberikan hasil `true`.

- Tidak ada perbedaan antara nilai `A` dan `B`, sehingga operator `>` dan `<` memberikan hasil `false`.

[No. 4] Operator Logika:

4.1. Identifikasi Masalah:

- Permasalahan:

Menggunakan operator logika (&&, ||, !) dengan nilai variabel `A = 3` dan `B = 5`.

- Variabel:

`A = 3`, `B = 5`.

4.2. Analisis dan Argumentasi:

1) Saya mengusulkan penggunaan operator logika untuk mengevaluasi apakah ekspresi dengan kondisi `A` dan `B` bernilai `true` atau `false`.

2) Alasan solusi ini karena operator logika digunakan untuk mengevaluasi lebih dari satu kondisi sekaligus.

3) Perbaikan kode adalah menambahkan beberapa ekspresi dengan operator logika `&&`, `||`, dan `!`.

4.3. Penyusunan Algoritma dan Kode Program:

1) Algoritma:

- Deklarasikan variabel `A` dan `B`.
- Evaluasi ekspresi dengan operator logika `&&` (AND), `||` (OR), dan `!` (NOT).
- Tampilkan hasil.

2) Kode Program dan Luaran:

```java
public class OperatorLogika {
    public static void main(String[] args) {
        int A = 3, B = 5;
        boolean hasil;

        hasil = (A > 2) && (B > 4);
        System.out.println("Hasil (A > 2) && (B > 4) = " + hasil);

        hasil = (A > 4) || (B > 4);
        System.out.println("Hasil (A > 4) || (B > 4) = " + hasil);

        hasil = !(A > 2);
        System.out.println("Hasil !(A > 2) = " + hasil);
    }
}
```

Collaborate

Luaran :

Output Generated Files

```
Hasil (A > 2) && (B > 4) = true
Hasil (A > 4) || (B > 4) = true
Hasil !(A > 2) = false
```

Compiled and executed in **1.249 sec(s)**

4.4. Kesimpulan:

Operator logika `&&` akan menghasilkan `true` jika kedua kondisi bernilai `true`. Operator `||` akan menghasilkan `true` jika salah satu kondisi bernilai `true`. Operator `!` akan membalikkan nilai kondisi, `true` menjadi `false` dan sebaliknya.

[No. 5] Operator Unary:

5.1. Identifikasi Masalah:

- Permasalahan:

Gunakan operator unary `++` dan `--` pada variabel `x = 5`.

- Variabel:

`x = 5`.

5.2. Analisis dan Argumentasi:

1) Saya mengusulkan penggunaan operator unary `++` dan `--` untuk melihat efek peningkatan dan penurunan nilai variabel `x`.

2) Alasan solusi ini adalah karena operator unary digunakan untuk menambah atau mengurangi nilai variabel sebanyak 1.

3) Perbaikan kode adalah menambahkan operasi unary `++` dan `--` sebelum dan sesudah variabel `x`.

5.3. Penyusunan Algoritma dan Kode Program:

1) Algoritma:

- Deklarasikan variabel `x`.
- Gunakan operator unary `++` dan `--` sebelum dan sesudah variabel.
- Tampilkan hasil.

2) Kode Program dan Luaran:

```java
public class OperatorUnary {
    public static void main(String[] args) {
        int x = 5;
        System.out.println("Nilai awal x = " + x);

        System.out.println("Nilai x++ = " + (x++)); // Post-increment
        System.out.println("Nilai setelah x++ = " + x);

        System.out.println("Nilai ++x = " + (++x)); // Pre-increment
        System.out.println("Nilai setelah ++x = " + x);

        System.out.println("Nilai x-- = " + (x--)); // Post-decrement
        System.out.println("Nilai setelah x-- = " + x);

        System.out.println("Nilai --x = " + (--x)); // Pre-decrement
        System.out.println("Nilai setelah --x = " + x);
    }
}
```

Luaran:

Output Generated Files

```
Nilai awal x = 5
Nilai x++ = 5
Nilai setelah x++ = 6
Nilai ++x = 7
Nilai setelah ++x = 7
Nilai x-- = 7
Nilai setelah x-- = 6
Nilai --x = 5
Nilai setelah --x = 5
```

Compiled and executed in **1.275 sec(s)**

5.4. Kesimpulan:

Operator unary `++` akan menambah nilai variabel sebanyak 1, sedangkan `--` akan menguranginya. Pre-increment/decrement (`++x`, `--x`) akan mengubah nilai sebelum digunakan, sementara post-increment/decrement (`x++`, `x--`) mengubahnya setelah digunakan.

[No. 6] Operator Ternary:

6.1. Identifikasi Masalah:

- Permasalahan:

Gunakan operator ternary untuk menentukan apakah suatu nilai ganjil atau genap.

- Variabel:

`n = 10`.

6.2. Analisis dan Argumentasi:

1) Saya mengusulkan untuk menggunakan operator ternary untuk memeriksa apakah nilai `n` ganjil atau genap.

2) Alasan solusi ini karena operator ternary adalah cara singkat untuk mengevaluasi ekspresi kondisi dalam satu baris kode.

3) Perbaikan kode adalah menggunakan operator ternary untuk memeriksa apakah nilai variabel `n` habis dibagi 2.

6.3. Penyusunan Algoritma dan Kode Program:

1) Algoritma:

- Deklarasikan variabel `n`.

- Gunakan operator ternary untuk mengevaluasi apakah `n` ganjil atau genap.

- Tampilkan hasil.

2) Kode Program dan Luaran:

```java
public class OperatorTernary {
    public static void main(String[] args) {
        int n = 10;
        String hasil = (n % 2 == 0) ? "Genap" : "Ganjil";
        System.out.println("Angka " + n + " adalah " + hasil);
    }
}
```

Luaran:

Output Generated Files

6.4. Kesimpulan:

Operator ternary adalah cara singkat untuk mengevaluasi kondisi, dalam hal ini apakah nilai variabel adalah ganjil atau genap. Kondisi `(n % 2 == 0)` memeriksa apakah `n` habis dibagi 2, jika ya, maka outputnya "Genap", jika tidak, "Ganjil".

[No. 7] Switch-Case:

7.1. Identifikasi Masalah:

- Permasalahan:

Gunakan pernyataan `switch-case` untuk menentukan nama hari berdasarkan angka `n`.

- Variabel:

`n = 3`.

7.2. Analisis dan Argumentasi:

1) Saya mengusulkan penggunaan pernyataan `switch-case` untuk memilih nama hari berdasarkan nilai variabel `n`.
2) Alasan solusi ini adalah karena `switch-case` digunakan untuk memilih satu dari banyak alternatif berdasarkan nilai variabel.
3) Perbaikan kode adalah menambahkan pernyataan `switch-case` yang sesuai dengan angka 1 hingga 7 untuk menentukan hari dalam seminggu.

7.3. Penyusunan Algoritma dan Kode Program:

1) Algoritma:

- Deklarasikan variabel `n`.

- Gunakan pernyataan `switch-case` untuk menentukan nama hari berdasarkan nilai `n`.

- Tampilkan hasil.

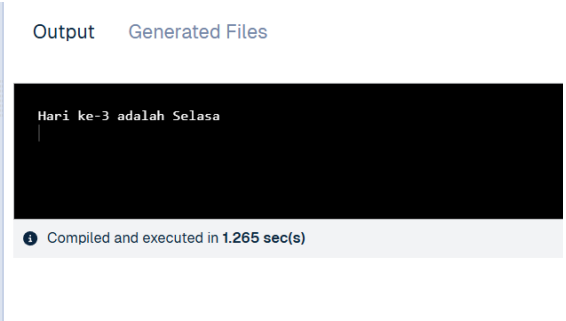
2) Kode Program dan Luaran:

```java
public class SwitchCaseHari {
    public static void main(String[] args) {
        int n = 3;
        String hari;

        switch(n) {
            case 1:
                hari = "Minggu";
                break;
            case 2:
                hari = "Senin";
                break;
            case 3:
                hari = "Selasa";
                break;
            case 4:
                hari = "Rabu";
                break;
            case 5:
                hari = "Kamis";
                break;
            case 6:
                hari = "Jumat";
                break;
            case 7:
                hari = "Sabtu";
                break;
            default:
                hari = "Nomor hari tidak valid";
        }

        System.out.println("Hari ke-" + n + " adalah " + hari);
    }
}
```

Luaran:



7.4. Kesimpulan:

`Switch-case` adalah cara yang efisien untuk memilih satu dari banyak opsi berdasarkan nilai suatu variabel. Pada contoh ini, angka `n` mewakili hari dalam seminggu, dan `switch-case` menentukan nama hari yang sesuai. Jika angka tidak valid (di luar 1-7), akan ditampilkan pesan default "Nomor hari tidak valid."

Refleksi:

Latihan ini memberikan pemahaman yang lebih dalam tentang cara kerja berbagai operator di dalam Java. Penggunaan operator aritmatika, penugasan, relasional, increment, decrement, logika, dan bitwise memberikan wawasan mengenai cara mengoperasikan data dalam berbagai skenario. Tantangan utama adalah memahami urutan prioritas operator dalam ekspresi yang lebih kompleks dan cara operator bekerja dengan tipe data yang berbeda.