|  |  |  |
| --- | --- | --- |
| **Nama & NPM** | **Topik:** | **Tanggal:** |
| **Winda Dwi Salistianna**  **G1F024004** | **Operator** | **12 september 2024** |
| **Operator Aritmatika** | | |
| **Contoh 1:**Salin dan tempel kode program berikut ke Eclipse atau JDoodle 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 }   }  **Luaran:**         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  **Latihan 1** 1.1. Rekomendasikan perbaikan kode agar program Contoh 1 dapat berjalan! 1.2. Tambahkan baris untuk menampilkan perhitungan dengan operator ( -, \*, /, %) pada Contoh 1  Jawab  Pada kode Contoh 1, terdapat kesalahan pada baris System.out.println("a + b = " (a + b));. Kesalahan ini terjadi karena sintaks yang salah saat mencetak hasil penjumlahan. Untuk mencetak hasil penjumlahan, Anda harus menggunakan operator + untuk menggabungkan string dan hasil operasi aritmatika. Ubah baris tersebut menjadi System.out.println("a + b = " + (a + b)); dengan menambahkan operator + untuk menggabungkan string dan hasil operasi.  Sebuah gambar berisi teks, cuplikan layar, software, tampilan  Deskripsi dibuat secara otomatis  Penjelasan kode :  Kode Java yang Anda berikan adalah contoh dasar penggunaan operator aritmatika dalam bahasa pemrograman Java. Program ini mendeklarasikan dua variabel integer, `a` dan `b`, dengan nilai 20 dan 3 masing-masing, dan kemudian menggunakan operator aritmatika untuk melakukan berbagai operasi matematis. Melalui baris-baris kode yang menggunakan `System.out.println`, program ini menampilkan hasil dari penjumlahan, pengurangan, perkalian, pembagian, dan sisa pembagian dari variabel-variabel tersebut. Dengan cara ini, Anda dapat melihat hasil dari operasi aritmatika dasar langsung di konsol. Program ini juga menunjukkan cara format output dengan menggabungkan string dan hasil operasi menggunakan operator `+`. Ini adalah cara yang berguna untuk menyajikan data dengan jelas dan informatif, serta membantu dalam debugging dan pemahaman hasil eksekusi program. Implementasi seperti ini adalah dasar bagi pemrograman lebih lanjut, memberikan fondasi yang kuat untuk mempelajari konsep-konsep lebih kompleks dalam Java.  Output :  Sebuah gambar berisi teks, cuplikan layar, software, tampilan  Deskripsi dibuat secara otomatis | | |
| ****Operator Penugasan**** | | |
| **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!  **Jawaban**  **2.1**  **Sebuah gambar berisi teks, cuplikan layar, software, Ikon komputer  Deskripsi dibuat secara otomatis**  Penjelasan Kode :  kode yang Anda berikan adalah contoh penggunaan operator penugasan dalam bahasa pemrograman Java. Program ini mendemonstrasikan bagaimana operator penugasan seperti +=, -=, \*=, /=, dan %= dapat digunakan untuk melakukan berbagai operasi aritmatika dan memperbarui nilai variabel dalam prosesnya. Program dimulai dengan mendeklarasikan dua variabel integer, a dengan nilai 20 dan b dengan nilai 3. Operator penugasan pertama yang digunakan adalah +=, yang menambahkan nilai a ke b, mengubah nilai b menjadi 23. Setelah itu, b diubah dengan operator -=, mengurangi nilai a dari b, menghasilkan -17.Selanjutnya, nilai b yang telah dimodifikasi diproses dengan operator \*=, yang mengalikan nilai a dengan b, menghasilkan 60. Program kemudian menggunakan operator /=, yang membagi nilai b dengan a dan hasilnya adalah 0 karena pembagian bilangan bulat 60 oleh 20 adalah 3. Terakhir, operasi modulus dilakukan dengan operator %= yang menghitung sisa pembagian dari b oleh a, menghasilkan 3. Program ini secara sistematis menunjukkan bagaimana setiap operator penugasan mempengaruhi nilai variabel dan mencetak hasil akhir dari setiap operasi ke konsol.  Output :    2.2 Perbedaan luaran antara Contoh 1 dan Contoh 2 terletak pada jenis operasi yang dilakukan. Contoh 1 hanya menampilkan hasil dari operasi aritmatika langsung tanpa menggunakan operator penugasan, sehingga hasil operasi aritmatika tidak mempengaruhi nilai variabel yang lain. Sedangkan Contoh 2 menggunakan operator penugasan yang tidak hanya melakukan operasi aritmatika tetapi juga menyimpan hasil operasi tersebut ke dalam variabel b. Ini berarti setiap baris yang menggunakan operator penugasan memodifikasi nilai variabel b secara bertahap, mempengaruhi hasil dari perhitungan berikutnya. Dalam hal waktu eksekusi, kedua contoh biasanya memiliki performa yang mirip karena keduanya melibatkan operasi aritmatika dasar, tetapi Contoh 2 mungkin memerlukan waktu tambahan untuk mengupdate dan menampilkan nilai variabel yang telah dimodifikasi. | | |
| ****Operator Relasional**** | | |
| **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!  Jawab :  **Sebuah gambar berisi teks, cuplikan layar, nomor, Font  Deskripsi dibuat secara otomatis**  **Penjelasan kode:**  Kode Java yang diberikan berfungsi untuk menunjukkan penggunaan operator relasional dalam membandingkan dua nilai integer, yaitu `nilaiA` dan `nilaiB`. Dalam kode ini, kedua variabel tersebut diatur dengan nilai yang sama, yaitu 4. Program ini menggunakan operator relasional (`>`, `<`, `>=`, `<=`, `==`, dan `!=`) untuk membandingkan nilai `nilaiA` dan `nilaiB`, dan kemudian mencetak hasil dari setiap perbandingan ke konsol dalam format yang jelas. Hasil setiap perbandingan disimpan dalam variabel boolean `hasil`, yang kemudian dicetak dengan deskripsi yang sesuai untuk setiap operator.  Hasil dari perbandingan tersebut adalah sebagai berikut: Operator `>` dan `<` akan menghasilkan `false` karena kedua nilai sama, sehingga tidak ada satu pun yang lebih besar atau lebih kecil dari yang lain. Sebaliknya, operator `>=` dan `<=` menghasilkan `true` karena nilai `nilaiA` sama dengan `nilaiB`, sehingga memenuhi kondisi "lebih besar atau sama dengan" dan "lebih kecil atau sama dengan". Operator `==` juga menghasilkan `true` karena kedua nilai identik, sementara operator `!=` menghasilkan `false` karena nilai-nilai tersebut tidak berbeda. Dengan cara ini, kode ini secara efektif menggambarkan cara operator relasional bekerja dalam perbandingan nilai di Java.  Dengan nilai nilaiA dan nilaiB yang diubah menjadi 4, hasil dari perbandingan relasional menunjukkan bahwa semua operator yang membandingkan kesamaan (==) dan ketidaksamaan (!=) sekarang menunjukkan hasil yang berbeda. Perbandingan yang melibatkan operator lebih besar dari (>) dan lebih kecil dari (<) menghasilkan hasil false karena kedua nilai sama. Operator >= dan <= menunjukkan true karena 4 sama dengan 4, memenuhi kedua kondisi tersebut. Ini mengilustrasikan bagaimana perubahan nilai variabel mempengaruhi hasil dari perbandingan relasional dalam program. | | |
| ****Operator Increment dan Decrement**** | | |
| **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!  Jawab:  4.1 untuk mencapai nilai a = 5 dan b = 6, Anda harus menggunakan operator pre-increment pada variabel a ketika mengassign nilai ke b. Operator pre-increment (++a) meningkatkan nilai a sebelum nilai tersebut digunakan dalam ekspresi, sehingga b mendapatkan nilai yang telah ditingkatkan.    Penjelasan Kode :  Kode Java yang diberikan mendemonstrasikan penggunaan operator pre-increment (`++a`) dalam konteks program sederhana. Dalam program ini, variabel `a` diinisialisasi dengan nilai 5. Baris pertama output menampilkan nilai `a` sebelum perubahan, yang akan menjadi `a: 5`. Selanjutnya, baris kedua menggunakan operator pre-increment untuk meningkatkan nilai `a` sebesar 1 sebelum nilai tersebut dicetak. Karena operator pre-increment meningkatkan nilai `a` terlebih dahulu, nilai `a` yang awalnya 5 menjadi 6 sebelum digunakan dalam ekspresi. Jadi, hasil dari baris kedua adalah `b: 6`, yang menunjukkan nilai terbaru dari `a` setelah peningkatan.Program ini menunjukkan perbedaan antara pre-increment dan post-increment. Dengan pre-increment, nilai `a` diubah terlebih dahulu sebelum digunakan, sedangkan post-increment akan menggunakan nilai saat ini sebelum melakukan perubahan. Dalam hal ini, penggunaan pre-increment (`++a`) memastikan bahwa perubahan nilai `a` terjadi segera sebelum hasilnya dicetak, mengakibatkan output `b: 6`. Ini memberikan pemahaman dasar tentang bagaimana operator aritmatika mempengaruhi nilai variabel dan bagaimana hasilnya ditampilkan.  Output :  Sebuah gambar berisi teks, cuplikan layar, Font, garis  Deskripsi dibuat secara otomatis  4.2 Dalam eksperimen ini, menggunakan pre-increment (++a) membuat nilai variabel a menjadi 6 sebelum nilai tersebut digunakan dalam ekspresi yang dicetak ke konsol. Dengan kode ini, b mendapatkan nilai 6, dan variabel a juga diperbarui menjadi 6. Hasil yang dicetak adalah a: 6 dan b: 6. Untuk mencapai hasil yang diinginkan dengan a = 5 dan **b = 6**, Anda perlu memastikan bahwa:Variabel a harus tetap 5 setelah statement print.b harus mendapatkan nilai hasil dari peningkatan a menjadi 6.Jika menggunakan post-increment (a++), a tetap 5 setelah statement System.out.println("b: " + (a++)), namun nilai b yang dicetak adalah 5. Menggunakan pre-increment pada a memastikan bahwa b mendapatkan nilai 6, dan Anda harus mencetak nilai a secara terpisah jika ingin menampilkan a sebagai 5 setelah b telah diperbarui. | | |
| ****Operator Logika**** | | |

**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. 

Jawaban :

5.1

Sebuah gambar berisi teks, Font, garis, nomor

Deskripsi dibuat secara otomatis

Penjelasan kode :

Kode Java yang diberikan menunjukkan penggunaan operator logika AND (&&) dalam konteks variabel boolean. Dalam program ini, dua variabel boolean a dan b diinisialisasi dengan nilai true. Operator && digunakan untuk mengevaluasi ekspresi logika (a && b), yang berarti bahwa hasil dari ekspresi ini akan true hanya jika kedua operandnya adalah true. Karena kedua variabel a dan b memiliki nilai true, ekspresi (a && b) menghasilkan true. Baris System.out.println("Hasil logika (a && b) : " + (a && b)); mencetak hasil dari evaluasi ekspresi ke konsol. Dengan kedua operand bernilai true, hasil logika dari a && b adalah true, dan ini dicetak sebagai Hasil logika (a && b) : true. Program ini secara efektif menunjukkan bagaimana operator AND digunakan dalam Java untuk menggabungkan dua kondisi boolean dan menghasilkan nilai berdasarkan logika AND.

5.2 Dari latihan ini, kita dapat menyimpulkan bahwa operator logika && dan || memiliki perilaku yang berbeda dalam menentukan hasil boolean:

* **Operator && (AND)**: Memerlukan kedua operand untuk bernilai true agar hasilnya true. Jadi, jika Anda ingin ekspresi (a && b) menghasilkan true, maka kedua a dan b harus diatur ke true.
* **Operator || (OR)**: Hanya memerlukan satu operand yang bernilai true agar hasilnya true. Jadi, untuk ekspresi (a || b) menghasilkan true, setidaknya salah satu dari a atau b harus true.

Dengan memahami perbedaan ini, Anda dapat lebih efektif dalam mengatur logika program berdasarkan kondisi yang diinginkan.

### ****Operator Kondisional (Ternary)****

**Contoh 6:**  
public class OperatorKondisi{  
   public static void main( String[] args ){  
      String status = "";  
      int nilai = 80;   
      status = (nilai > 60)?"Lulus":"Gagal";  
      System.out.println( status );  
}    }

**Luaran:**  
Lulus

**Latihan 6**  
Rekomendasikan apa bentuk tanda operator agar nilai = 60 memenuhi untuk Lulus !  
  Jawab :

Sebuah gambar berisi teks, cuplikan layar, Font, deasin

Deskripsi dibuat secara otomatisSebuah gambar berisi teks, Font, garis, cuplikan layar

Deskripsi dibuat secara otomatis

Penjelasan Kode :

**Operator >=**: Operator ini memeriksa apakah nilai di sebelah kiri lebih besar dari atau sama dengan nilai di sebelah kanan. Dalam hal ini, nilai >= 60 akan bernilai true jika nilai adalah 60 atau lebih besar.

Dengan mengubah operator dari > ke >=, program ini akan mencetak "Lulus" jika nilai sama dengan atau lebih besar dari 60, sehingga nilai 60 akan memenuhi syarat untuk mendapatkan "Lulus".

### ****Operator Bitwise****

**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!

### Jawab

Hasil dari `~a` yang bernilai `-11` pada kode Java tersebut dapat dijelaskan dengan menggunakan konsep representasi biner dan operasi bitwise. Dalam kasus ini, `a` adalah `10`, yang dalam biner 32-bit adalah `0000 0000 0000 0000 0000 0000 0000 1010`. Operator bitwise NOT (`~`) membalikkan semua bit dari nilai ini, mengubah `0` menjadi `1` dan sebaliknya. Hasil dari operasi ini adalah `1111 1111 1111 1111 1111 1111 1111 0101`, yang merupakan representasi biner dari angka negatif dalam format komplemen dua.

Dalam format komplemen dua, bit paling kiri menunjukkan tanda angka. Untuk mendapatkan nilai desimal dari hasil biner `1111 1111 1111 1111 1111 1111 1111 0101`, kita harus mengkonversinya dengan langkah-langkah khusus. Pertama, kita balikkan semua bit, menghasilkan `0000 0000 0000 0000 0000 0000 0000 1010`, kemudian tambahkan `1` untuk mendapatkan `0000 0000 0000 0000 0000 0000 0000 1011`. Konversi nilai biner `1011` ini ke desimal memberi kita `11`, dan karena angka asli adalah negatif, hasil akhirnya adalah `-11`. Dengan demikian, operasi bitwise NOT pada `10` menghasilkan `-11` karena proses ini mengubah representasi angka positif ke bentuk negatif sesuai dengan aturan komplemen dua.