|  |  |  |
| --- | --- | --- |
| **Nama & NPM** | **Topik:** | **Tanggal:** |
| **Michelia Erza Annadhira**  **G1F024035** | **Operator Aritmatika : +, -, \*, /** | **6 September 2024** |
| **[1] Identifikasi Masalah:** | | |
| 1. Uraikan permasalahan dan variabel   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!   * Pada soal masih terdapat kesalahan sintaksis pada baris :   System.out.println("a + b = "  (a + b));  Untuk memperbaikinya tambahkan tanda tambah (+) antara string dan hasil perhitungan aritmatika :  System.out.println("a + b = " + (a + b));   * Tambahkan baris untuk menampilkan perhitungan dengan operator (-, \*, /, %) :   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));   1. Rincikan sumber informasi yang relevan (buku / webpage)   **Video Materi 1** – Definisi operator, unary, binary, ternary, operasi aritmatika, penugasan, relasional, increment/decrement pada laman   <https://www.youtube.com/watch?v=PzCMZObexZM>  **Video Materi 2** – Operator Logika, Kondisional, Bitwise, dan contoh pembahasan soal pada laman <https://www.youtube.com/watch?v=LcFgl0yrKEw> | | |
| **[2] Analisis dan Argumentasi** | | |
| 1. Saya mengusulkan permasalahan ini dapat diatasi dengan cara menambahkan operator tanda tambah (+) di antara string dan ekspresi aritmatika pada pernyataan System.out.println yang mencetak hasil penjumlahan (+), pengurangan (-), perkalian (\*), pembagian (/) dan modulus (%) 2. Alasan solusi ini karena dalam Java, saat menggabungkan teks (string) dengan ekspresi atau nilai numerik, kita perlu menggunakan operator + agar Java tahu bahwa kita ingin menggabungkan string dengan hasil perhitungan. Tanpa tanda tambah (+), program tidak akan dapat menyatukan teks dan hasil perhitungan, yang menyebabkan kesalahan sintaks 3. Perbaikan kode program dengan cara menambah operator (+) setelah string di setiap baris System.out.println yang mencetak hasil operasi aritmatika. Seperti :  * Menambahkan operator tanda tambah (+) :   System.out.println("a + b = "  (a + b));  menjadi :  System.out.println("a + b = " + (a + b));   * Tambahkan baris untuk menampilkan perhitungan dengan operator (-, \*, /, %) :   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)); | | |
| **[3] Penyusunan Algoritma dan Kode Program** | | |
| * Algoritma  1. Mulai program 2. Deklarasikan variabel a dan b sebagai integer :  * a = 20 * b = 3  1. Cetak nilai variabel a dan b : System.out.println("a: " +a); System.out.println("b: " +b); 2. Lakukan operasi aritmatika antara a dan b :  * Penjumlahan a + b * Pengurangan a - b * Perkalian a \* b * Pembagian a / b * Modulus a % b  1. Cetak hasil dari setiap operasi aritmatika : 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)); 2. Jalankan kode untuk memeriksa apakah program bias dijalankan dengan baik dan tidak ada kesalahan 3. Simpan kode program yang telah diperbaiki dan catat perubahan yang telah dilakukan 4. Selesai     Luaran (output) :    Kode program sudah disusun dengan struktur yang benar. Hal ini dapat dilihat dari luaran (output) yang sesuai dengan kode program dan tidak terjadi eror lagi | | |
| **[4] Kesimpulan** | | |
| * Pada program ini, saya menggunakan operator aritmatika dasar seperti penjumlahan (+), pengurangan (-), perkalian (\*), pembagian (/), dan modulus (%). Permasalahan utamanya yang saya temukan adalah kurangnya operator + untuk menggabungkan string dan ekspresi dalam System.out.println. Setelah menambahkan operator yang hilang, program berhasil berjalan. * Dasar alasan pengambilan keputusan karena setiap ekspresi dalam System.out.println memerlukan operator + untuk menggabungkan string dengan hasil perhitungan. Tanpa operator tersebut, Java tidak dapat memahami cara menggabungkan string dan ekspresi. | | |
|  | | |

|  |  |  |
| --- | --- | --- |
| **Nama & NPM** | **Topik:** | **Tanggal:** |
| **Michelia Erza Annadhira**  **G1F024035** | **Operator Penugasan : =** | **6 September 2024** |
| **[1] Identifikasi Masalah:** | | |
| 1. Uraikan permasalahan dan variabel   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!   * Tambahkan baris untuk menampilkan perhitungan dengan operator (-=,\*=, /=,%=) :   b -= a; System.out.println("Penambahan : " + b);  b \*= a; System.out.println("Penambahan : " + b);  b /= a; System.out.println("Penambahan : " + b);  b %= a; System.out.println("Penambahan : " + b);   * Berikan argumentasi tentang perbedaan luaran dan waktu eksekusi contoh 1 dan 2:   Perbedaan luaran : Contoh 1, hanya menggunakan operator aritmatika dasar seperti +, -, \*, / dan % tanpa memodifikasi nilai variabel aslinya. Hasil dari operasi hanya ditampilkan, tetapi nilai asli dari variabel tidak berubah. Contoh 2, menggunakan operator penugasan seperti +=, -=, \*=, /= dan %= yang secara langsung memodifikasi nilai variabel b berdasarkan operasi yang dilakukan. Nilai b diperbarui setelah setiap operasi.  Waktu eksekusi : contoh 1 dan contoh 2 memiliki waktu eksekusi yang sama cepat karena kedua program melakukan operasi aritmatika sederhana pada variabel.   1. Rincikan sumber informasi yang relevan (buku / webpage)   **Video Materi 1** – Definisi operator, unary, binary, ternary, operasi aritmatika, penugasan, relasional, increment/decrement pada laman   <https://www.youtube.com/watch?v=PzCMZObexZM>  **Video Materi 2** – Operator Logika, Kondisional, Bitwise, dan contoh pembahasan soal pada laman <https://www.youtube.com/watch?v=LcFgl0yrKEw> | | |
| **[2] Analisis dan Argumentasi** | | |
| 1. Saya mengusulkan permasalahan ini dapat diatasi dengan cara menambahkan operator penugasan lainnya (-=, \*=, /=, %=) pada kode, sehingga program dapat menampilkan hasil dari operasi tersebut selain hanya menggunakan += 2. Alasan solusi ini karena dengan menggunakan operator penugasan, kita bias mengubah nilai variabel langsung berdasarkan hasil operasi aritmatika, dan menampilkan hasil dari operasi tersebut dengan lebih ringkas. 3. Perbaikan kode program dengan cara menambah operator penugasan yang lain seperti -=, \*=, /= dan %= setelah baris penambahan (+=) untuk memberikan hasil dari setiap operasi aritmatika secara langsung. Seperti :  * Tambahkan baris untuk menampilkan perhitungan dengan operator (-=,\*=, /=,%=) :   b -= a; System.out.println("Penambahan : " + b);  b \*= a; System.out.println("Penambahan : " + b);  b /= a; System.out.println("Penambahan : " + b);  b %= a; System.out.println("Penambahan : " + b) | | |
| **[3] Penyusunan Algoritma dan Kode Program** | | |
| * Algoritma  1. Mulai program 2. Deklarasikan variabel a dan b sebagai integer :  * a = 20 * b = 3  1. Lakukan operasi aritmatika menggunakan operator penugasan dan perbarui nilai b :  * Penambahan : b += a; (b = b + a) * Pengurangan : b -= a; (b = b – a) * Perkalian : b \*= a; (b = b \* a) * Pembagian : b /= a; (b = b / a) * Modulus : b %= a; (b = b % a)  1. Jalankan kode untuk memeriksa apakah program bias dijalankan dengan baik dan tidak ada kesalahan 2. Simpan kode program yang telah diperbaiki dan catat perubahan yang telah dilakukan 3. Selesai     Luaran (output) :    Kode program sudah disusun dengan struktur yang benar. Hal ini dapat dilihat dari luaran (output) yang sesuai dengan kode program dan tidak terjadi eror lagi | | |
| **[4] Kesimpulan** | | |
| * Pada program ini, saya menggunakan berbagai operator penugasan untuk mengoperasikan variabel integer dan menampilkan hasilnya. Dengan menggunakan operator penugasan, program ini secara efisien mengubah nilai variabel b dan menampilkan hasilnya setiap operasi. Operator penugasan memberikan cara ringkas dan jelas untuk melakukan perhitungan dan memodifikasi nilai variabel secara langsung. * Dasar alasan pengambilan keputusan, program diperbaiki dengan menambahkan operasi menggunakan semua operator penugasan (-=, \*=, /=, %=) selain += yang sudah ada. Ini memberikan pandangan lengkap tentang bagaimana setiap operator penugasan memodifikasi nilai variabel secara langsung dan memperbarui hasil yang ditampilkan. | | |
|  | | |

|  |  |  |
| --- | --- | --- |
| **Nama & NPM** | **Topik:** | **Tanggal:** |
| **Michelia Erza Annadhira**  **G1F024035** | **Operator Relasional : <, >, <=, >=, =, ==, !=** | **6 September 2024** |
| **[1] Identifikasi Masalah:** | | |
| 1. Uraikan permasalahan dan variabel   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!   * Ubahlah nilai A = 4 dan B = 4 dan simpulkan yang terjadi :   Karena nilai A dan B sama (4), maka hasil perbadingan untuk A > B dan A < B adalah false. untuk operator relasional >= dan <= , hasilnya true karena kedua nilai sama dan memenuhi syarat “lebih besar sama dengan” dan lebih kecil sama dengan”. A == B menghasilkan true karena kedua nilai sama. A != B menghasilkan false karena nilai A dan B sama, sehingga mereka tidak “berbeda”   1. Rincikan sumber informasi yang relevan (buku / webpage)   **Video Materi 1** – Definisi operator, unary, binary, ternary, operasi aritmatika, penugasan, relasional, increment/decrement pada laman   <https://www.youtube.com/watch?v=PzCMZObexZM>  **Video Materi 2** – Operator Logika, Kondisional, Bitwise, dan contoh pembahasan soal pada laman <https://www.youtube.com/watch?v=LcFgl0yrKEw> | | |
| **[2] Analisis dan Argumentasi** | | |
| 1. Saya mengusulkan permasalahan ini dapat diatasi dengan cara mengganti nilai variabel A dan B sesuai dengan yang ingin diuji, misalnya jika kita ingin menguji dua nilai yang sama, tetapkan A = 4 dan B = 4 . Jika ingin menguji nilai yang berbeda, gunakan nilai lain untuk kedua variabel tersebut 2. Alasan solusi ini karena variabel A dan B menentukan hasil perbandingan operator relasional. Dengan mengubah nilai variabel tersebut, kita bisa menguji berbagai berbagai kondisi dan memahami bagaimana masing-masing operator bekerja dalam berbagai situasi. 3. Perbaikan kode program dengan cara memperbarui nilai variabel A dan B di bagian deklarasi variabel. Misalnya, untuk menguji kondisi di mana kedua nilai sama, ubah int nilaiA = 4 dan int nilaiB = 4. Jika ingin menguji kondisi lain, atur variabel tersebut dengan nilai yang sesuai. | | |
| **[3] Penyusunan Algoritma dan Kode Program** | | |
| * Algoritma  1. Mulai 2. Deklarasikan dua variabel nilaiA dan nilaiB dengan tipe data integer 3. Deklarasikan variabel hasil dengan tipe data Boolean 4. Tampilkan nilai A dan B. 5. Bandingkan nilai A dan B menggunakan operator relasional :  * Periksa apakah A lebih besar dari B (A > B). // apakah A lebih besar dari B? hasil = nilaiA > nilaiB; System.out.println("\n Hasil A > B = "+ hasil); * Periksa apakah A lebih kecil dari B (A < B). // apakah A lebih kecil dari B? hasil = nilaiA < nilaiB; System.out.println("\n Hasil A < B = "+ hasil); * Periksa apakah A lebih besar atau sama dengan B (A >= B). // apakah A lebih besar samadengan B? hasil = nilaiA >= nilaiB; System.out.println("\n Hasil A >= B = "+ hasil); * Periksa apakah A lebih kecil atau sama dengan B (A <= B). // apakah A lebih kecil samadengan B? hasil = nilaiA <= nilaiB; System.out.println("\n Hasil A <= B = "+ hasil) * Periksa apakah A sama dengan B (A == B). // apakah nilai A sama dengan B? hasil = nilaiA == nilaiB; System.out.println("\n Hasil A == B = "+ hasil); * Periksa apakah A tidak sama dengan B (A != B). // apakah nilai A tidak samadengan B? hasil = nilaiA != nilaiB; System.out.println("\n Hasil A != B = "+ hasil);  1. Jalankan kode untuk memeriksa apakah program bias dijalankan dengan baik dan tidak ada kesalahan 2. Simpan kode program yang telah diperbaiki dan catat perubahan yang telah dilakukan 3. Selesai     Luaran (output) :    Kode program sudah disusun dengan struktur yang benar. Hal ini dapat dilihat dari luaran (output) yang sesuai dengan kode program dan tidak terjadi eror lagi | | |
| **[4] Kesimpulan** | | |
| * Pada program ini, saya menggunakan operator relasional seperti >, <, >=, <=, == dan != untuk membandingkan dua nilai integer, yaitu nilaiA dan nilaiB. Hasil dari perbadingan ini ditampilkan untuk setiap operator, sehingga dapat dilakukan apakah suatu nilai lebih besar, lebih kecil, sama dengan, atau berbeda dari nilai lainnya. Dengan ini, program dapat memberikan hasil perbandingan logis antara dua angka yang diberikan. * Perbaikan program dengan mengubah nilai variabel nilaiA dan nilaiB sesuai dengan kondisi yang ingin diuji memungkinkan kita untuk melihat hasil dari berbagai scenario perbandingan. Dasar alasan pengambilan keputusan ini adalah agar kita dapat memahami cara kerja operator relasional dalam kondisi yang berbeda, baik ketika nilai-nilai sama maupun ketika salah satunya lebih besar atau lebih kecil. | | |
|  | | |

|  |  |  |
| --- | --- | --- |
| **Nama & NPM** | **Topik:** | **Tanggal:** |
| **Michelia Erza Annadhira**  **G1F024035** | **Operator Increment dan Decrement : ++, --** | **6 September 2024** |
| **[1] Identifikasi Masalah:** | | |
| * + - 1. Uraikan permasalahan dan variabel   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!   * Saran operasi agar menghasilkan a = 5 dan b = 6 : untuk mendapat luaran a = 5 dan b = 6, kita harus menggunakan pre-increment (++a). Hal ini karena pre-increment akan menaikkan nilai variabel sebelum dievaluasi dalam operasi * Post-increment (a++) : nilai variabel a akan dikembalikan dulu dalam operasi (sehingga b = 5) kemudian baru ditambahkan 1. Oleh Karena itu, meskipun setelahnya nilai a menjadi 6, tetapi nilai b tetap 5 karena penambahan dilakukan setelahnya. * Pre-increment (++a) : nilai variabel a akan langsung ditambah 1 sebelum digunakan dalam operasi (sehingga b = 6), tetapi a tetap sebagai 5 pada baris pertama sebelum ditambah. * kesimpulan hasil eksperimen : hasil eksperimen saya menunjukkan bahwa : * Post-increment (a++) mengembalikan nilai a terlebih dahulu sebelum increment terjadi, sehingga pada output awal nilai a dan b tetap sama. * Pre-increment (++a) mengubah nilai a sebelum dievaluasi, sehingga dapat menghasilkan output yang berbeda untuk a dan b.   Untuk kasus ini, menggunakan pre-increment akan memberikan hasil sesuai permintaan soal (yaitu a = 5 dan b = 6).   * + - 1. Rincikan sumber informasi yang relevan (buku / webpage)   **Video Materi 1** – Definisi operator, unary, binary, ternary, operasi aritmatika, penugasan, relasional, increment/decrement pada laman   <https://www.youtube.com/watch?v=PzCMZObexZM>  **Video Materi 2** – Operator Logika, Kondisional, Bitwise, dan contoh pembahasan soal pada laman <https://www.youtube.com/watch?v=LcFgl0yrKEw> | | |
| **[2] Analisis dan Argumentasi** | | |
| 1. Saya mengusulkan permasalahan ini dapat diatasi dengan cara menggunakan pre-increment (++a) untuk variabel a pada baris yang mencetak nilai b. Dengan cara ini, nilai variabel a akan ditingkatkan sebelum digunakan dalam ekspresi yang mencetak nilai b 2. Alasan solusi ini karena pre-increment () akan menambah nilai variabel sebelum digunakan dalam operasi. Dengan demikian, a akan dicetak sebagai 5 pada baris pertama, tetapi ketika digunakan dalam ekspresi berikutnya, nilai a sudah ditingkatkan menjadi 6, sehingga hasik untuk variabel b adalah 6 3. Perbaikan kode program dengan cara mengubah baris : System.out.println("b: " + (a++)); menjadi : System.out.println("b: " + (++a)); sehinga nilai variabel b akan menjadi 6 setelah pre-increment diterapkan pada a. | | |
| **[3] Penyusunan Algoritma dan Kode Program** | | |
| * Algoritma  1. Mulai 2. Deklarasikan variabel a dengan nilai awal 5  * int a = 5;  1. Cetak nilai variabel a  * System.out.println("a: " +a);  1. Lakukan pre-increment pada variabel a (tambahkan 1 ke nilai a sebelum digunakan)  * System.out.println("b: " + (a++));  1. Cetak nilai variabel b yang sama dengan hasil pre-increment a  * Output : b = 6  1. Jalankan kode untuk memeriksa apakah program bias dijalankan dengan baik dan tidak ada kesalahan 2. Simpan kode program yang telah diperbaiki dan catat perubahan yang telah dilakukan 3. Selesai     Luaran (output)    Kode program sudah disusun dengan struktur yang benar. Hal ini dapat dilihat dari luaran (output) yang sesuai dengan kode program dan tidak terjadi eror lagi | | |
| **[4] Kesimpulan** | | |
| * Pada program ini, saya menggunakan menggunakan pre-increment (++a) untuk mengatasi permasalahan output yang diinginkan, yaitu mendapatkan nilai a = 5 dan b = 6. Dalam algoritma, langkah pre-increment akan menaikan nilai variabel a terlebih dahulu sebelum digunakan dalam ekspresi yang mencetak b. hal memastikan bahwa ketika variabel a dicetak, nilai awalnya masih 5, tetapi nilai b adalah hasil dari a yang di pre-increment menjadi 6. * Dasar alasan pengambilan keputusan karena operasi ini memungkinkan peningkatan nilai variabel a sebelum variabel tersebut digunakan dalam ekspresi. Dengan cara ini, output yang diminta, yaitu a = 5 dan b = 6 dapat tercapai. Jika menggunakan post-increment (a++), variabel a akan tetap bernilai 5 ketika digunakan dalam ekspresi, sehingga tidak sesuai dengan kebutuhan hasil soal. | | |
|  | | |

|  |  |  |
| --- | --- | --- |
| **Nama & NPM** | **Topik:** | **Tanggal:** |
| **Michelia Erza Annadhira**  **G1F024035** | **Operator Logika : &&, ||, !** | **6 September 2024** |
| **[1] Identifikasi Masalah:** | | |
| 1. Uraikan permasalahan dan variabel   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.   * Agar hasil dari operator && bernilai true , kedua variabel harus bernilai true. Seperti : * a = true * a = true * Agar hasil dari || bernilai true, cukup salah satu dari variabel a atau b ernilai true. Seperti : * a = true * a = false (atau a = false dan b = true, atau kedua bernilai true) * Kesimpulan dari latihan 5.1 : * Operator && (AND) hanya akan menghasilkan true kedua operand (variabel a dan b) bernilai true * Operator || (OR) akan menghasilkan true jika salah satu atua kedua operand bernilai true  1. Rincikan sumber informasi yang relevan (buku / webpage)   **Video Materi 1** – Definisi operator, unary, binary, ternary, operasi aritmatika, penugasan, relasional, increment/decrement pada laman   <https://www.youtube.com/watch?v=PzCMZObexZM>  **Video Materi 2** – Operator Logika, Kondisional, Bitwise, dan contoh pembahasan soal pada laman <https://www.youtube.com/watch?v=LcFgl0yrKEw> | | |
| **[2] Analisis dan Argumentasi** | | |
| 1. Saya mengusulkan permasalahan ini dapat diatasi dengan cara menyesuaikan nilai variabel a dan b agar sesuai dengan kebutuhan hasil logika yang diinginkan untuk operator && dan ||. 2. Alasan solusi ini karena :  * Operator && membutuhkan kedua operand bernilai true untuk mengasilkan true * Operator || hanya membutuhkan salah satu dari operand bernilai true untuk menghasilkan true  1. Perbaikan kode program dengan cara :  * Untuk mengahasil luaran true dengan operator &&, setel a dan b keduanya menjadi true. * Untuk menghasilkan luaran true dengan operator ||, setel minimal salah satu dari a atau b menjadi true. Atau bisa disetel dua-duanya menjadi true. | | |
| **[3] Penyusunan Algoritma dan Kode Program** | | |
| * Algoritma  1. Mulai 2. Deklarasikan dua variabel boolean a dan b 3. Inisialisasi variabel  * Set nilai a ke true : boolean a = true; * Set nilai b ke true : boolean b = false;  1. Perhitungan operator logika  * Hitung hasil dari a && b * Hitung hasil dari a || b  1. Tampilkan hasil dari a && b dengan pesan yang sesuai System.out.println("Hasil logika (a && b) : " + (a && b));  //menampilkan hasil logika AND 2. Tampilkan hasil dari a || b dengan pesan yang sesuai System.out.println("Hasil logika (a || b) : " + (a || b));  //menampilkan hasil logika OR 3. Jalankan kode untuk memeriksa apakah program bias dijalankan dengan baik dan tidak ada kesalahan 4. Simpan kode program yang telah diperbaiki dan catat perubahan yang telah dilakukan 5. Selesai     Luaran (output) :    Kode program sudah disusun dengan struktur yang benar. Hal ini dapat dilihat dari luaran (output) yang sesuai dengan kode program dan tidak terjadi eror lagi | | |
| **[4] Kesimpulan** | | |
| * Pada program ini, saya menggunakan operator logika && untuk menentukan hasil logika AND dan menampilkan hasil operasi tersebut. Dengan nilai awal a = true dan b = false, hasil dari a && b adalah false, karena dala operator AND, kedua operand harus bernilai true agar hasilnya true. Dalam algoritma logika, jika salah satu dari operand tidak memenuhi kondisi true, maka hasilnya adalah false. * Perbaikan program dilakukan dengan memastikan bahwa pemahaman dan penerapan operator logika benar. Dengan mengubah nilai variabel a dan b sesai dengan logika yang diterapkan, kita bisa memverifikasi hasil yang diharapkan. Untuk kasus ini, pemilihan nilai variabel yang tepat memastikan bahwa hasil output sesuai dengan ekspresi logika AND dan OR. | | |
|  | | |

|  |  |  |
| --- | --- | --- |
| **Nama & NPM** | **Topik:** | **Tanggal:** |
| **Michelia Erza Annadhira**  **G1F024035** | **Operator Kondisional (Ternary): ?:** | **6 September 2024** |
| **[1] Identifikasi Masalah:** | | |
| 1. Uraikan permasalahan dan variabel   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 !   * Diketahui dari soal, variabel nilai saat ini memiliki nilai 80, dan hasil dari status adalah “lulus” karena kondisi nilai > 60 terpenuhi. * Untuk memenuhi syarat agar nilai = 60 juga menghasilkan “lulus”, kita perlu mengubah operator perbandingan dari > menjadi >= pada ekspresi kondisional. Ini akan memastikan bahwa nilai 60 juga bisa memenuhi syarat dan menghasilkan “lulus”  1. Rincikan sumber informasi yang relevan (buku / webpage)   **Video Materi 1** – Definisi operator, unary, binary, ternary, operasi aritmatika, penugasan, relasional, increment/decrement pada laman   <https://www.youtube.com/watch?v=PzCMZObexZM>  **Video Materi 2** – Operator Logika, Kondisional, Bitwise, dan contoh pembahasan soal pada laman <https://www.youtube.com/watch?v=LcFgl0yrKEw> | | |
| **[2] Analisis dan Argumentasi** | | |
| 1. Saya mengusulkan permasalahan ini dapat diatasi dengan cara mengubah operator perbandingan dari > menjadi >=. 2. Alasan solusi ini karena operator >= akan memasukkan nilai yang sama dengan batas (dalam hal ini 60) ke dalam kondisi yang memenuhi, sehingga ketika nilai adalah 60, hasilnya tetap “lulus”. 3. Perbaikan kode program dengan cara mengganti bagian kondisi pada operator ternary dari nilai > 60 menjadi nilai >= 60, sehingga menjadi seperti ini :  * status = (nilai >= 60)?"Lulus":"Gagal"; | | |
| **[3] Penyusunan Algoritma dan Kode Program** | | |
| * Algoritma   Mulai  Deklarasi variabel :   * Deklarasikan variabel nilai dengan tipe data String dan inisialisasikan dengan nilai kosong String status = ""; * Deklarasikan variabel nilai dengan tipe data int dan inisialisasikan dengan nilai 60 int nilai = 60;   Tentukan kondisi :   * Gunakan operator ternary untuk memeriksa nilai dari variabel nilai. Jika nilai >= 60, maka tetapkan status menjadi “lulus”. Jika tidak, tetapkan status menjadi “gagal” status = (nilai > 60)?"Lulus":"Gagal";   Tampilkan hasil :   * Cetak nilai dari variabel status ke layar System.out.println( status );   Jalankan kode untuk memeriksa apakah program bias dijalankan dengan baik dan tidak ada kesalahan  Simpan kode program yang telah diperbaiki dan catat perubahan yang telah dilakukan  Selesai    Luaran (output) :    Kode program sudah disusun dengan struktur yang benar. Hal ini dapat dilihat dari luaran (output) yang sesuai dengan kode program dan tidak terjadi eror lagi | | |
| **[4] Kesimpulan** | | |
| * Pada program ini, saya menggunakan operator ternary untuk menetapan nilai variabel status berdasarkan kondisi nilai dari variabel nilai. Awalnya, program tersebut menggunakan operator perbandingan > untuk menentukan apakah nilai lebih besar dari 60, yang berarti hanya nilai di atas 60 yang dianggap “lulus”. Namun, untuk memenuhi syarat agar nilai 60 juga dianggap “lulus”, saya mengganti operator perbandingan menjadi >=, sehingga nilai 60 termasuk dalam kondisi yang memenuhi syarat. * Dasar alasan pengambilan keputusan, memperbaiki program dengan mengganti operator perbandingan dari > menjadi >= diperlukan untuk memastikan bahwa nilai pada batas (60) juga memastikan syarat dan menghasilkan output yang sesuai. Dengan menggunakan operator >=, kita dapat memastikan bahwa setiap nilai yang sama dengan atau lebih besar dari 60 akan menghasilkan “lulus”, sesuai dengan kebutuhan logika program yang diinginkan. | | |
|  | | |

|  |  |  |
| --- | --- | --- |
| **Nama & NPM** | **Topik:** | **Tanggal:** |
| **Michelia Erza Annadhira**  **G1F024035** | **Operator Bitwise : &, |, ^, ~, <<, >>, >>>** | **6 September 2024** |
| **[1] Identifikasi Masalah:** | | |
| * + - 1. Uraikan permasalahan dan variabel   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!   * Untuk memahami hasil dari operasi bitwise ~a dalam kode program tersebut, kita perlu mengevaluasi bagaimana operasi ~ (bitwise NOT) bekerja pada angka biner. * Langkah-langkah evaluasi :   + - * 1. Konversi bilangan ke Biner : * a = 10 * Representasi biner dari 10 adalah 00001010 (dalam 8 bit)   + - * 1. Operasi Bitwise NOT (~) : * Operasi ~ membalik setiap bit dari bilangan biner * Jadi, ~00001010 menjadi 11110101.   + - * 1. Konversi hasil kembali ke decimal : * 11110101 adalah bilangan biner dalam format komplemen dua (two`s compleme * Untuk mengubahnya ke desimal :  1. Hitung komplemen dua dari 11110101 : 2. 00001011 dalam desimal adalah 11 3. Karena bilangan asli negatif, hasilnya adalah -11  * ~a menghasilkan -11 karena bitwise NOT membalik setiap bit dan dalam system two`s complement, hasil tersebut berfungsi sebagai -11   + - 1. Rincikan sumber informasi yang relevan (buku / webpage)   **Video Materi 1** – Definisi operator, unary, binary, ternary, operasi aritmatika, penugasan, relasional, increment/decrement pada laman   <https://www.youtube.com/watch?v=PzCMZObexZM>  **Video Materi 2** – Operator Logika, Kondisional, Bitwise, dan contoh pembahasan soal pada laman <https://www.youtube.com/watch?v=LcFgl0yrKEw> | | |
| **[2] Analisis dan Argumentasi** | | |
| 1. Saya mengusulkan permasalahan ini dapat diatasi dengan cara menggunakan representasi biner dan memahami system komplemen dua (two`s complement) untuk menginterpretasikan hasil dari operasi bitwise NOT (~) 2. Alasan solusi ini karena operasi bitwise NOT membalik setiap bit dari bilangan biner. Dalam system two`s complement, hasil dari bitwise NOT adalah bilangan negative yang diperoleh dengan membalik bit dan menambahkan 1 ke nilai yang dibalik. 3. Perbaikan kode program dengan cara, jika kita ingin menghindari kebingungan dengan nilai negatif atau hanya menunjukkan hasil dalam format decimal positif, kita bias menggunakan fungsi tambahan untuk menampilkan nilai biner dan hasil decimal secara terpisah, atau memperjelas hasil dengan komentar tambahan. Seperti :  * Menambahkan operator tanda tambah (+) :   hasil = ~a; System.out.println("Hasil dari ~a : " + hasil );  menjadi :  hasil = ~a; System.out.println("Hasil dari ~a : " + hasil ); System.out.println("Hasil biner dari ~a : " + Integer.toBinaryString(hasil));  Dengan tambahan Integer.toBinaryString(hasil). Kita dapat melihat represiasi biner dari hasil operasi bitwise NOT dan lebih memahami bagaimana hasil tersebut diperoleh. | | |
| **[3] Penyusunan Algoritma dan Kode Program** | | |
| * Algoritma  1. Mulai program 2. Deklarasi variabel  * Buat dua variabel a dan b bertipe integer * Inisialisasi nilai a = 10 dan b = 7. * Deklasrasi variabel hasil untuk menyimpan hasil operasi  1. Lakukan operasi bitwise AND (&) :  * Hitung hasil = a & b * Tampilkan hasil dari a & b dalam bentuk desimal  1. Lakukan operasi bitwise OR (|) :  * Hitung hasil = a | b * Tampilkan hasil dari a | b dalam bentuk desimal  1. Lakukan operasi bitwise XOR (^) :  * Hitung hasil = a ^ b * Tampilkan hasil dari a ^ b dalam bentuk desimal  1. Lakukan operasi bitwise NOT (~) pada a :  * Hitung hasil = ~a * Tampilkan hasil dari ~a dalam bentuk desimal * Tampilkan hasil dari ~a dalam bentuk biner menggunakan integer.toBinaryString()  1. Lakukan operasi bitwise right shift (>>) pada a :  * Hitung hasil = a >> 1 * Tampilkan hasil dari a >> 1 dalam bentuk desimal  1. Lakukan operasi bitwise left shift (<<) pada b :  * Hitung hasil = b << 2 * Tampilkan hasil dari b << 2 dalam bentuk desimal  1. Jalankan kode untuk memeriksa apakah program bias dijalankan dengan baik dan tidak ada kesalahan 2. Simpan kode program yang telah diperbaiki dan catat perubahan yang telah dilakukan 3. Selesai     Luaran (output) :    Kode program sudah disusun dengan struktur yang benar. Hal ini dapat dilihat dari luaran (output) yang sesuai dengan kode program dan tidak terjadi eror lagi | | |
| **[4] Kesimpulan** | | |
| * Pada program ini, saya menggunakan operasi bitwise seperti AND (&), OR (|), XOR (^), NOT (~), right shift (>>) dan left shift (<<) untuk memanipulasi bilangan dalam bentuk biner. Setiap operasi menghasilkan output yang dihitung berdasarkan operasi bit pada representasi biner dari bilangan a dna b. operasi NOT pada a menghasilkan bilangan negatif karena menggunakan representasi komplemen dua (two`s complement), di mana bit-bit di balik dan hasilnya menjadi negatif setelah ditambahkan 1. Untuk mempermudah analisis, hasil operasi ditampilkan dalam format desimal dan biner * Perbaikan program dilakukan dengan menambahkan tampilan representasi biner dari hasil operasi NOT (~a). Alasanya, untuk memberikan pemahaman yang lebih baik tentang bagaimana operasi bitwise NOT bekerja pada level bit. Dengan menunjukkan hasil dalam bentuk biner, lebih mudah untuk melihat bagaimana bit-bit dibalik dan mengapa hasil akhirnya adalah bilangan negatif ketika dikonversi kembali ke desimal. | | |
|  | | |