|  |  |  |
| --- | --- | --- |
| **Nama & NPM** | **Topik:** | **Tanggal:** |
| **Zaira ayu wandira**  **G1F024055** | **Operator java** | **5 septemper 2024** |
| **[No.1] Identifikasi Masalah:** | | |
| 1.1.  Tambahkan baris system.out.println(“a + b= “ + (a +b));  Ubahlah operator ( + ) dengan tanda ( -, \*, /, %) 1.2.  Analisa perhitungan matematika yang terjadi! | | |
| **[No.1l] Analisis dan Argumentasi** | | |
| 1.2.analisa setiap perhitungan:  1.Penjumlahan (a + b):  20 + 3 = 23  Hasil dari penjumlahan variabel a dan b.  2.Pengurangan (a - b):  20 - 3 = 17  Hasil dari pengurangan variabel a dan b.  3.Perkalian (a \* b):  20 \* 3 = 60  Hasil dari perkalian variabel a dan b.  4.Pembagian (a / b):  20 / 3 = 6  Hasil dari pembagian variabel a dengan b, dalam hal ini hasil pembagian adalah bilangan bulat (integer) yang mendekati nilai sebenarnya.  5.Sisa bagi (a % b):  20 % 3 = 2  Hasil dari operasi modulo, yaitu sisa dari pembagian 20 dengan 3. | | |
| **[No.1] Penyusunan Algoritma dan Kode Program** | | |
| **Algorima:**  1).mulai  2).berikan nama class dan file  3). tambahkan method main  4).deklarasi nilai a dan b  5). tambah kan system.out.println demgan mengunakan operator aritmatik  6).run  7).selesai    Luaran sudah sesuai dengan program yang disusun | | |
| **[No.1] Kesimpulan** | | |
| operasi aritmatika dasar dalam Java, seperti penjumlahan, pengurangan, perkalian, pembagian, dan modulo, masing-masing memberikan hasil yang sesuai dengan aturan matematika. | | |
| **[No. 2] Identifikasi Masalah:** | | |
| 2.1.  Bandingkan hasil Contoh 1 dengan Contoh 2! | | |
| **[No.2] Analisis dan Argumentasi** | | |
| Contoh 1 fokus pada operasi aritmatika sederhana dengan nilai tetap.  Contoh 2 menunjukkan bagaimana operator penugasan dapat digunakan untuk melakukan operasi berturut-turut pada variabel, yang mengubah nilai variabel tersebut secara bertahap.  Perbedaan utama adalah bahwa Contoh 1 hanya menampilkan hasil operasi aritmatika  tanpa perubahan nilai variabel, sementara Contoh 2 secara aktif mengubah nilai variabel b dengan setiap operator penugasan. | | |
| **[No.2 ] Penyusunan Algoritma dan Kode Program** | | |
| **Algoritma:**  1).mulai  2).berikan nama class dan file  3). tambahkan method main  4).deklarasikan variable integer a dan b  5). Print variable dengan menggunakan operator penugasan  6).run  7.)selesai    Luaran sudah sesuai dengan program yang disusun | | |
| **[No.2] Kesimpulan** | | |
| Contoh 1 menekankan pada perhitungan matematis sederhana dengan hasil langsung, sementara Contoh 2 menggambarkan bagaimana nilai variabel dapat dimodifikasi secara berurutan dengan operator penugasan dan efeknya terhadap hasil akhir | | |
| **[No. 3] Identifikasi Masalah:** | | |
| 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.3l] Analisis dan Argumentasi**  3.1 analisis Perubahan:   1. **A > B:**    * Dengan nilai A = 4 dan B = 4, A > B (4 > 4) adalah false karena 4 tidak lebih besar dari 4. 2. **A < B:**    * Dengan nilai A = 4 dan B = 4, A < B (4 < 4) adalah false karena 4 tidak lebih kecil dari 4. 3. **A >= B:**    * Dengan nilai A = 4 dan B = 4, A >= B (4 >= 4) adalah true karena 4 lebih besar atau sama dengan 4. 4. **A <= B:**    * Dengan nilai A = 4 dan B = 4, A <= B (4 <= 4) adalah true karena 4 lebih kecil atau sama dengan 4. 5. **A == B:**    * Dengan nilai A = 4 dan B = 4, A == B (4 == 4) adalah true karena 4 sama dengan 4. 6. **A != B:**    * Dengan nilai A = 4 dan B = 4, A != B (4 != 4) adalah false karena 4 tidak berbeda dari 4.   3.2   Bandingkan bagaimana perbedaan nilai A dan B mempengaruhi nilai luaran   * Ketika**A**dan**B**berbeda (misalnya,**A = 12**dan**B = 4**):   Perbandingan seperti A > B, A < B, dan lainnya memberikan hasil yang bervariasi berdasarkan hubungan numerik antara A dan B.   * Ketika**A**dan**B**sama (misalnya,**A = 4**dan**B = 4**):   + A == B adalah true dan A != B adalah false karena nilai-nilai sama.   + A > B dan A < B menjadi false karena tidak ada perbedaan nilai.   + A >= B dan A <= B menjadi true karena A sama dengan B, sehingga memenuhi kondisi lebih besar atau sama dengan dan lebih kecil atau sama dengan.   Dengan demikian, perubahan nilai A dan B mempengaruhi hasil perbandingan relasional secara langsung, dengan nilai yang sama menghasilkan hasil perbandingan yang stabil di sekitar kesetaraan dan ketidaksetaraan, sementara nilai yang berbeda mengarahkan pada hasil yang sesuai dengan hubungan numerik antara kedua nilai. | | |
| **[No.3] Penyusunan Algoritma dan Kode Program** | | |
| **Algoritma:**  1).mulai  2).berikan nama class dan file  3). tambahkan method main  4).membuat variabel nilaiA,variabel NilaiB,dan boolen hasil  5)mendeklarasikan dengan operator relasional  6)run  7.selesai    Luaran sudah sesuai dengan program yang disusun. | | |
| **[No.3] Kesimpulan** | | |
| Operator relasional dalam Java digunakan untuk membandingkan dua nilai dan menghasilkan hasil boolean (True  atau false) berdasarkan hubungan antara nilai tersebut. Perubahan nilai variabel mempengaruhi hasil dari perbandingan: operator == dan != mengukur kesamaan atau perbedaan, sementara >, <, >=, dan <= mengevaluasi hubungan ukuran relatif. Dengan memahami operator ini, kita dapat menulis logika program yang lebih tepat dan terkontrol. | | |
| **[No. 4] Identifikasi Masalah:** | | |
| 4.1  Berdasarkan luaran program Contoh 4, bandingkan hasil Post dan Pre untuk  Increment dan decrement | | |
| **[No.4] Analisis dan Argumentasi** | | |
| Perbandingan:  * **Increment**:   + **Post Increment** (a++): Mencetak nilai sebelum increment dan kemudian menaikkan nilai variabel.   + **Pre Increment** (++b): Mencetak nilai setelah increment, menaikkan nilai variabel sebelum mencetak. * **Decrement**:   + **Post Decrement** (c--): Mencetak nilai sebelum decrement dan kemudian menurunkan nilai variabel.   + **Pre Decrement** (--d): Mencetak nilai setelah decrement, menurunkan nilai variabel sebelum mencetak.   Secara umum, Post Increment/Decrement (a++, c--) mempengaruhi nilai variabel setelah mencetak hasil, sedangkan Pre Increment/Decrement (++b, --d) mempengaruhi nilai variabel sebelum mencetak hasil. | | |
| **[No.4] Penyusunan Algoritma dan Kode Program** | | |
| **Algorima:**  1).mulai  2).berikan nama class dan file  3). tambahkan method main  4).membuat variabel dengan tipe data integer  5)mendeklarasikan variabel  6)run  7.selesai    Luaran sudah sesuai dengan program yang disusun | | |
| **[No.4] Kesimpulan**  **Post Increment dan Post Decrement** menunjukkan nilai variabel lama sebelum operasi, dan kemudian melakukan perubahan (penambahan atau pengurangan).  **Pre Increment dan Pre Decrement** melakukan perubahan terlebih dahulu, kemudian menampilkan nilai variabel yang sudah diperbarui. | | |
| **[No. 5] Identifikasi Masalah:** | | |
| 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** | | |
| 5.2. Analisis hasil:  a && b (false && false) menghasilkan False  karena operasi AND (&&) memerlukan kedua operator bernilai True untuk menghasilkan True  a || b  (false || false) menghasilkan False karena operasi OR (||) memerlukan minimal satu operand bernilai True untuk menghasilkan True  5.3 Analisis:  analisis pernyataan a || b && a || !b dengan nilai a= true dan b = false  Pernyataan yang akan dianalisis  a || b && a || !b  Langkah-langkah Analisis  1. Evaluasi b && a:  •Operator && (AND) memiliki prioritas lebih tinggi dari || (OR).  •Dengan b = false dan a = true, maka b && a adalah false && true, yang hasilnya adalah `false.  2. Evaluasi !b:  •Operator ! (NOT) memiliki prioritas lebih tinggi daripada || (OR).  •Dengan b = false, maka !b adalah true.  3. Substitusi hasil dalam pernyataan:  •Setelah evaluasi, pernyataan menjadi a || false || true.  4. Evaluasi keseluruhan pernyataan:  •Dengan a = true, pernyataan true || false || true adalah true karena operator || (OR) memerlukan setidaknya satu operand true agar hasil keseluruhan menjadi true.  Hasil Akhir  Jadi, dengan a = true dan b = false, pernyataan a || b && a || !b akan dievaluasi sebagai true.  Dengan penjelasan ini, kita bisa melihat bagaimana operator logika && dan || serta operator ! berperilaku sesuai prioritasnya dan mempengaruhi hasil akhir dari pernyataan logika. | | |
| **[No.5] Penyusunan Algoritma dan Kode Program** | | |
| 5.1 Tambahkan baris kode untuk memeriksa a || b.  **Algorima:**  1).mulai  2).berikan nama class dan file  3). tambahkan method main  4).membuat variabel  5)mendeklarasikan variabel  6)run  7.selesai  5.2 Ubahlah nilai a = false dan b = false.  **Algorima:**  1).mulai  2).berikan nama class dan file  3). tambahkan method main  4).membuat variabel  5)mendeklarasikan variabel  6)run  7.selesai  5.3. Apabila diketahui pernyataan a || b && a || !b.  **Algorima:**  1).mulai  2).berikan nama class dan file  3). tambahkan method main  4).membuat variabel  5)mendeklarasikan variabel  6)run  7)selesai | | |
| **[No.5]Kesimpulan** | | |
| Kesimpulannya adalah sebagai berikut: Operator logika && (AND) dan || (OR) memiliki peran penting dalam menentukan hasil ekspresi boolean. Operator && memerlukan kedua operand bernilai true untuk menghasilkan true, sementara || hanya memerlukan satu operand bernilai true. Perubahan nilai boolean dari true ke false mempengaruhi hasil operasi logika; misalnya, false && false dan false || false keduanya menghasilkan false. Dalam pernyataan kompleks seperti a || b && a || !b, urutan evaluasi mengikuti prioritas operator, dengan && dan ! dievaluasi sebelum ||. Dengan nilai a = false dan b = false, pernyataan tersebut menghasilkan true karena !b bernilai true, yang mendominasi hasil akhir. | | |
| **[No. 6] Identifikasi Masalah:** | | |
| Berdasarkan Contoh 6, ubahlah nilai = 60. Analisis hasil dan proses yang terjadi | | |
| **[No.6] Analisis dan Argumentasi** | | |
| **Analisis:**   1. **Evaluasi Kondisi:**   Dengan nilai yang baru yaitu 60, kondisi (nilai > 60) akan mengevaluasi ke false karena 60 tidak lebih besar dari 60.   1. **Hasil dari Operator Kondisional:**   Karena kondisi adalah false, maka nilai hasil dari operator kondisional adalah "Gagal".   1. **Hasil Output:**   System.out.println(status); akan mencetak "Gagal". | | |
| **[No.6] Penyusunan Algoritma dan Kode Program** | | |
| **Algorima:**  1).mulai  2).berikan nama class dan file  3). tambahkan method main  4).membuat variabel  5)mendeklarasikan variabel  6)run  7)selesai | | |
| **[No.6 ] kesimpulan** | | |
| Ketika nilai adalah 80, kondisi (nilai > 60, )adalah True, sehingga status menjadi “Lulus” Namun, ketika nilaidiubah menjadi 60, kondisi nilai > 60,)  menjadi false, sehingga status menjadi “gagal” | | |
| **[No. 7]] Identifikasi Masalah:** | | |
| Pilihlah 3 perhitungan Contoh 7, kemudian uraikan perhitungan biner! Simpulkan hasilnya | | |
| **[No.7] Analisis dan Argumentasi** | | |
| Berikut adalah uraian perhitungan biner untuk operator &, |, dan ^:   1. **Perhitungan a & b:**   **Nilai variabel:**  a = 10 (biner: 1010)  b = 7 (biner: 0111)  **Operasi:**  a & b (AND bitwise)  **Perhitungan biner:**  1010  & 0111  ------  0010  **Hasil:**  Biner: 0010  Desimal: 2  **2.Perhitungan a | b:**   * **Nilai variabel:**   a = 10 (biner: 1010)  b = 7 (biner: 0111)   * **Operasi:**   a|b (OR bitwise)  **hitungan biner:**  1010  | 0111  ------  1111   * **Hasil:**   Biner: 1111  Desimal: 15  **3. Perhitungan a ^ b:**   * **Nilai variabel:**   a = 10 (biner: 1010)  b = 7 (biner: 0111)   * **Operasi:**   a ^ b (XOR bitwise)   * **Perhitungan biner:**   1010  ^ 0111  ------  1101   * **Hasil:**   Biner: 1101  Desimal: 13 | | |
| **[No.7] Penyusunan Algoritma dan Kode Program** | | |
| **Algorima:**  1).mulai  2).berikan nama class dan file  3). tambahkan method main  4).membuat variabel  5)mendeklarasikan variabel  6)run  7)selesai | | |
| **[No.7] kesimpulan** | | |
| Untuk a & b, yang merupakan operasi AND bitwise, hasilnya adalah 2 karena bit yang sama antara 1010 dan 0111 hanya pada posisi tertentu yang menghasilkan 0010 dalam bentuk biner. Sedangkan pada operasi OR bitwise (a | b), hasilnya adalah 15, yang menunjukkan bahwa setiap bit di posisi yang sama antara kedua angka menghasilkan 1, menghasilkan 1111. Terakhir, operasi XOR bitwise (a ^ b) menghasilkan 13, karena XOR menghasilkan 1 di posisi bit yang berbeda antara kedua angka, yang dalam kasus ini adalah 1101. Kesimpulannya, operator bitwise memberikan alat yang efisien untuk manipulasi bit dan operasional yang memerlukan pengolahan data pada level yang sangat mendetail | | |
|  | | |
| **Refleksi:**  **Dari pembelajaran ini saya mendapat banyak pengetahuan baru seperti belajar tentang penggunaan operator dan penggunaan operator untuk memecahkan masalah.** | | |