**Template Lembar Kerja Individu**

|  |  |  |
| --- | --- | --- |
| **Nama & NPM** | **Topik:** | **Tanggal:** |
| **DIFA PUTRA PERDANA**  **G1F024072** | **IF DAN SWITH JAVA** | **07-10-2024** |
| **[Nomor Soal ]** | | |
| Soal:  1.    Bilangan genap merupakan bilangan yang habis dibagi 2. Bilangan ganjil adalah bilangan yang tidak habis dibagi 2. Analisa kode program yang tepat untuk menghitung masukan pengguna termasuk bilangan genap atau bilangan ganjil (lihat Contoh 1)?           (Petunjuk: hitung = nilai % 2 jika hitung = 0 maka bilangan genap, jika hitung = 1 maka bilangan ganjil)  2.    Cermati contoh 2, analisa kondisi pada IF bersarang!             Tambahkan satu kondisi IF dengan satu nilai input Quiz (nilaiQ). Jika nilai UTS, Tugas, dan Quiz lebih besar sama dengan 80 maka siswa mendapat nilai A.  3.    Apakah ketiga kondisi IF pada Contoh 1.2. dapat diringkas menjadi satu kondisi?           Periksa satu kondisi mana yang paling tepat menggantikan  ketiga kondisi itu!           a.    IF (nilaiU >= 80 || nilaiT >= 80 || nilaiQ >= 80)           b.    IF (nilaiU >= 80 || nilaiT >= 80 && nilaiQ >= 80)           c.    IF (nilaiU >= 80 && nilaiT >= 80 || nilaiQ >= 80)           d.    IF (nilaiU >= 80 && nilaiT >= 80 && nilaiQ >= 80)  4.    Uraikan gambar diagram flowchart dari Latihan 1.2! | | |
| **[Nomor Soal] Analisis dan Argumentasi** | | |
| 1. Uraikan rancangan solusi yang diusulkan. 2. Analisis solusi, kaitkan dengan permasalahan. | | |
| **[Nomor Soal] Penyusunan Algoritma dan Kode Program** | | |
|  | | |
| **[Nomor Soal] Kesimpulan** | | |
| 1. Analisa 2. Susunlah kesimpulan berdasarkan permasalahan, algoritma, dan kode program! 3. Apakah dasar alasan pengambilan keputusan Anda untuk kasus ini? 4. Evaluasi 5. Apa konsekuensi dari skenario pemprograman ini? 6. Evaluasi input, proses, dan luaran yang dihasilkan! (jika ada) 7. Kreasi 8. Apakah ada pengetahuan baru yang dikembangkan dan konsep baru sebagai usulan solusi? 9. Konstruksikan hubungan antara variabel yang berbeda dengan konsep yang anda ketahui! (jika ada)   **Kode di atas merupakan implementasi yang baik dari konsep percabangan if-else dalam bahasa Java.** Program ini dirancang untuk:   1. **Meminta input:** Meminta pengguna untuk memasukkan sebuah angka. 2. **Memvalidasi input:** Memeriksa apakah input yang diberikan pengguna adalah angka. 3. **Melakukan perbandingan:** Membandingkan angka yang dimasukkan dengan nilai 10. 4. **Memberikan output:** Memberikan output yang berbeda-beda berdasarkan hasil perbandingan.   **Berikut adalah rincian evaluasi untuk setiap bagian kode:**   * **Import:**   + import java.util.Scanner; digunakan untuk mengimpor kelas Scanner yang diperlukan untuk membaca input dari pengguna. * **Deklarasi variabel:**   + Scanner scanner = new Scanner(System.in); digunakan untuk membuat objek Scanner yang akan digunakan untuk membaca input dari keyboard. * **Permintaan input:**   + System.out.print("Masukkan angka: "); digunakan untuk menampilkan pesan ke layar agar pengguna memasukkan angka. * **Validasi input:**   + if (scanner.hasNextInt()) { digunakan untuk memeriksa apakah input yang diberikan pengguna adalah angka. Jika benar, maka program akan melanjutkan ke langkah berikutnya. * **Percabangan:**   + if (nilai == 10), else if (nilai > 10), dan else digunakan untuk membandingkan nilai yang dimasukkan dengan 10 dan memberikan output yang sesuai. * **Penutupan scanner:**   + scanner.close(); digunakan untuk menutup objek Scanner setelah selesai digunakan.   **Kelebihan kode ini:**   * **Validasi input:** Kode telah menambahkan validasi input untuk memastikan pengguna hanya memasukkan angka. Ini menghindari error yang mungkin terjadi jika pengguna memasukkan karakter selain angka. * **Percabangan yang jelas:** Penggunaan if-else if-else membuat struktur kode menjadi jelas dan mudah dipahami. * **Output yang informatif:** Output yang diberikan kepada pengguna cukup jelas dan memberikan informasi yang relevan. * **Penggunaan scanner.close():** Ini adalah praktik yang baik untuk menutup objek Scanner setelah selesai digunakan agar tidak terjadi kebocoran memori.   **Potensi Pengembangan:**   * **Lebih banyak kondisi:** Kita bisa menambahkan lebih banyak kondisi percabangan untuk memeriksa berbagai kemungkinan nilai input. * **Penggunaan fungsi:** Kode dapat dipecah menjadi beberapa fungsi untuk meningkatkan modularitas dan readability. * **Penggunaan loop:** Jika ingin melakukan perbandingan untuk banyak nilai, kita bisa menggunakan loop seperti for atau while.   **Kesimpulan:**  Kode ini merupakan contoh yang baik dari penerapan percabangan if-else dalam bahasa Java. Kode ini telah mencakup aspek-aspek penting seperti validasi input, percabangan, dan output. Dengan pemahaman yang baik tentang kode ini, Anda dapat mengembangkan program Java yang lebih kompleks dan bermanfaat.  **Kode program yang telah kita modifikasi ini berfungsi untuk menentukan apakah seorang siswa mendapatkan nilai A berdasarkan nilai tugas dan kuisnya.**  **Berikut adalah ringkasan cara kerja program:**   1. **Meminta input:** Program meminta pengguna untuk memasukkan nilai tugas dan nilai kuis. 2. **Memeriksa kondisi:** Program memeriksa apakah kedua nilai tersebut sama dengan atau lebih besar dari 70. 3. **Memberikan output:**    * Jika kedua nilai memenuhi syarat, program akan mencetak pesan "Anda mendapatkan nilai A".    * Jika salah satu atau kedua nilai tidak memenuhi syarat, program akan mencetak pesan "Anda TIDAK mendapatkan nilai A".   **Perbaikan yang telah dilakukan:**   * **Penyederhanaan kondisi:** Kondisi if dibuat lebih sederhana dengan menggunakan operator && untuk menggabungkan dua kondisi menjadi satu. * **Peningkatan efisiensi:** Dengan menghilangkan kondisi yang tidak perlu, kode menjadi lebih efisien. * **Peningkatan readability:** Kode menjadi lebih mudah dibaca dan dipahami dengan struktur yang lebih sederhana.   **Keunggulan kode yang telah dimodifikasi:**   * **Akurat:** Kode memberikan hasil yang akurat sesuai dengan kriteria yang ditentukan. * **Fleksibel:** Kode dapat dengan mudah dimodifikasi untuk mengubah kriteria nilai A atau menambahkan kriteria lainnya. * **Efisien:** Kode berjalan dengan efisien karena tidak ada percabangan yang tidak perlu.   **Potensi pengembangan:**   * **Skala nilai:** Kita bisa menambahkan kondisi untuk nilai B, C, dan seterusnya dengan menggunakan else if. * **Bobot nilai:** Kita bisa memberikan bobot yang berbeda untuk nilai tugas dan kuis untuk menghitung nilai akhir. * **Input validasi:** Kita bisa menambahkan validasi untuk memastikan pengguna hanya memasukkan nilai antara 0-100. * **Antarmuka pengguna:** Kita bisa membuat antarmuka pengguna yang lebih menarik dengan menggunakan library seperti Swing atau JavaFX.   **Kesimpulan:**  Kode program ini merupakan contoh sederhana namun efektif dari penggunaan struktur kontrol if dalam bahasa Java. Dengan memahami konsep dasar ini, Anda dapat membangun program yang lebih kompleks dan bermanfaat. | | |
|  | | |
| **Nama & NPM** | **Topik:** | **Tanggal:** |
| **DIFA PUTRA PERDANA**  **G1F024072** | **IF DAN SWITH JAVA** | **07-10-2024** |
|  |  |  |
| **[Nomor Soal]** | | |
| Soal:  1.    Cermati kode pada Contoh 3.            Hapuslah kode break; pada //baris 1, eksekusi kembali.            Kemudian hapuslah kode break; pada //baris 2, eksekusi kembali.           Analisis perbedaan hasil luaran ketika kode menggunakan break, ketika kode break baris 1 dihapus, dan ketika kode break baris 2 dihapus!            Analisa kegunaan baris kode dengan break dan kata kunci default!  2.    Cermati kode pada Contoh 4 yang menampilkan jumlah hari sesuai dengan bulannya. Namun kode tersebut baru sampai bulan ke-4. Tambahkan sampai bulan ke-12 pada baris ke-1 dan baris ke-2.  3.    Cermati permasalahan yang dipecahkan pada Contoh 3.             Apakah masalah ini bisa diubah menjadi perintah IF?            Jika bisa, rincikan analisa Anda!            Bandingkan masalah yang dapat diselesaikan percabangan dengan IF atau SWITCH !  4.    Buatlah dokumentasi gambar flowchart dari Latihan 2.1. dan Latihan 2.2! | | |
| **[Nomor Soal] Analisis dan Argumentasi** | | |
| 1. Uraikan rancangan solusi yang diusulkan. 2. Analisis solusi, kaitkan dengan permasalahan. | | |
| **[Nomor Soal] Penyusunan Algoritma dan Kode Program** | | |
|  | | |
| **[Nomor Soal] Kesimpulan** | | |
| 1. Analisa 2. Susunlah kesimpulan berdasarkan permasalahan, algoritma, dan kode program! 3. Apakah dasar alasan pengambilan keputusan Anda untuk kasus ini? 4. Evaluasi 5. Apa konsekuensi dari skenario pemprograman ini? 6. Evaluasi input, proses, dan luaran yang dihasilkan! (jika ada) 7. Kreasi 8. Apakah ada pengetahuan baru yang dikembangkan dan konsep baru sebagai usulan solusi? 9. Konstruksikan hubungan antara variabel yang berbeda dengan konsep yang anda ketahui! (jika ada)   **Analisis Kode dan Kesimpulan**  **Kode di atas merupakan contoh sederhana penggunaan struktur switch-case dalam bahasa Java untuk membuat keputusan berdasarkan suatu nilai.**  **Bagaimana Kode Bekerja?**   1. **Meminta Input:** Program meminta pengguna untuk memilih antara huruf 'A' atau 'B'. 2. **Membaca Input:** Pilihan pengguna disimpan dalam variabel data bertipe karakter. 3. **Percabangan:**    * **Case 'A':** Jika pengguna memilih 'A', program akan menampilkan pesan bahwa pengguna rajin belajar.    * **Case 'B':** Jika pengguna memilih 'B', program akan menampilkan pesan agar pengguna mengurangi bermain game.    * **Default:** Jika pengguna memilih selain 'A' atau 'B', program akan menampilkan pesan bahwa pilihannya tidak valid.   **Keunggulan switch-case:**   * **Mudah Dibaca:** Struktur switch-case membuat kode lebih mudah dipahami dibandingkan dengan menggunakan banyak if-else. * **Efisien:** Untuk banyak kemungkinan nilai, switch-case seringkali lebih efisien daripada if-else. * **Fleksibel:** Dapat digunakan dengan berbagai tipe data, tidak hanya karakter.   **Kesimpulan**  Kode ini menunjukkan dasar penggunaan switch-case dalam Java. Ini adalah alat yang berguna untuk membuat keputusan berdasarkan nilai suatu variabel.  **Konsep kunci yang perlu diingat:**   * **switch:** Memulai blok percabangan. * **case:** Mendefinisikan nilai yang akan dibandingkan. * **break:** Mengakhiri blok case saat kondisi terpenuhi. * **default:** Digunakan jika tidak ada case yang cocok.   **Potensi Pengembangan:**   * **Lebih Banyak Kasus:** Tambahkan lebih banyak pilihan dengan menambahkan lebih banyak case. * **Nested Switch:** Buat switch bersarang untuk percabangan yang lebih kompleks. * **Validasi Input:** Pastikan input pengguna valid (misalnya, hanya karakter tertentu). * **Fungsi:** Buat fungsi terpisah untuk setiap kasus untuk meningkatkan modularitas.   **Analisis Kode dan Kesimpulan**  **Kode di atas dirancang untuk menghitung jumlah hari dalam suatu bulan, dengan mempertimbangkan tahun kabisat untuk bulan Februari.**  **Bagaimana Kode Bekerja:**   1. **Meminta Input:** Program meminta pengguna untuk memasukkan nomor bulan. 2. **Menentukan Jumlah Hari:**    * **Bulan dengan 31 hari:** Januari, Maret, Mei, Juli, Agustus, Oktober, Desember.    * **Bulan dengan 30 hari:** April, Juni, September, November.    * **Februari:**      + Jika tahun adalah tahun kabisat (habis dibagi 4), maka Februari memiliki 29 hari.      + Jika tidak, Februari memiliki 28 hari. 3. **Menampilkan Hasil:** Program akan menampilkan jumlah hari yang sesuai dengan bulan yang dimasukkan.   **Kelebihan Kode:**   * **Efisien:** Menggunakan switch-case untuk percabangan membuat kode lebih efisien daripada menggunakan banyak if-else. * **Jelas:** Logika program mudah dipahami karena struktur switch-case yang jelas. * **Fleksibel:** Dapat dengan mudah dimodifikasi untuk menghitung jumlah hari pada tahun yang berbeda.   **Potensi Pengembangan:**   * **Tahun yang Dinamis:** Bisa menambahkan input untuk tahun sehingga program lebih fleksibel untuk menghitung jumlah hari pada tahun yang berbeda. * **Validasi Input:** Bisa menambahkan validasi untuk memastikan input bulan berada dalam rentang 1-12. * **Fungsi:** Bisa membuat fungsi terpisah untuk menghitung jumlah hari agar kode lebih modular. * **Array:** Bisa menggunakan array untuk menyimpan jumlah hari setiap bulan sehingga tidak perlu menulis banyak case.   **Kesimpulan:**  Kode ini adalah contoh yang baik dari penggunaan switch-case untuk menyelesaikan masalah sederhana dalam pemrograman. Dengan sedikit modifikasi, kode ini dapat menjadi lebih fleksibel dan dapat digunakan untuk berbagai keperluan.  **Konsep Penting yang Digunakan:**   * **switch-case:** Digunakan untuk membuat keputusan berdasarkan nilai suatu variabel. * **Tahun Kabisat:** Tahun yang habis dibagi 4. * **Operator Modulus (%):** Digunakan untuk mencari sisa pembagian.   **Contoh Penggunaan dalam Kehidupan Nyata:**   * **Aplikasi kalender:** Untuk menghitung jumlah hari dalam suatu bulan. * **Perencanaan acara:** Untuk menentukan tanggal jatuh tempo suatu proyek atau acara.   **Pertanyaan:**   * Apakah Anda ingin melihat contoh kode yang lebih kompleks menggunakan switch-case? * Apakah Anda ingin mempelajari cara membuat fungsi untuk menghitung jumlah hari? * Apakah Anda ingin mengetahui bagaimana cara membuat program ini lebih interaktif dengan pengguna?   **Dalam Bahasa Indonesia yang Lebih Sederhana:**  Kode ini menghitung jumlah hari dalam satu bulan. Kita bisa memasukkan nomor bulan, lalu program akan memberitahu kita berapa hari dalam bulan itu. Program ini sudah memperhitungkan tahun kabisat (tahun yang bisa dibagi 4) untuk bulan Februari. | | |
|  | | |

hadapi pada minggu tersebut. Ringkasan singkat dari semua soal, bukan per soal)