Template Lembar Kerja Individu

Nama & NPM	Topik:	Tanggal:
ANGGUN SYAVIRA TRINANDA (G1F024071)	IF dan SWITCH Java	24 September 2024

[No. 1] Identifikasi Masalah:

1) Uraikan permasalahan dan variabel

Contoh 1: Salin dan tempel kode program berikut ke Eclipse.

```
import java.util.Scanner; //memanggil impor package yang membaca masukan pengguna

public class PercabanganIf {
    public static void main(String[] args) {
Scanner masuk = new Scanner(System.in);  // membaca teks yang dimasukkan pengguna
System.out.print("Masukkan Angka Anda : ");  //pengguna memasukkan data
        int nilai =  masuk.nextByte();  //menyimpan masukan pengguna ke tipe data

if (nilai == 10) {  //percabangan yang memeriksa kondisi
System.out.println("Sepuluh");  //baris kode yang dieksekusi bila benar
        }
else  {  //baris kode yang dieksekusi bila kondisi tidak terpenuhi dan salah
    System.out.println("Nilai Bukan Sepuluh");
}    }
}
```

Luaran Contoh 1:

```
Masukkan Angka Anda : 8
Nilai Bukan Sepuluh
```

Contoh 2: Salin dan tempel kode program berikut ke Eclipse.

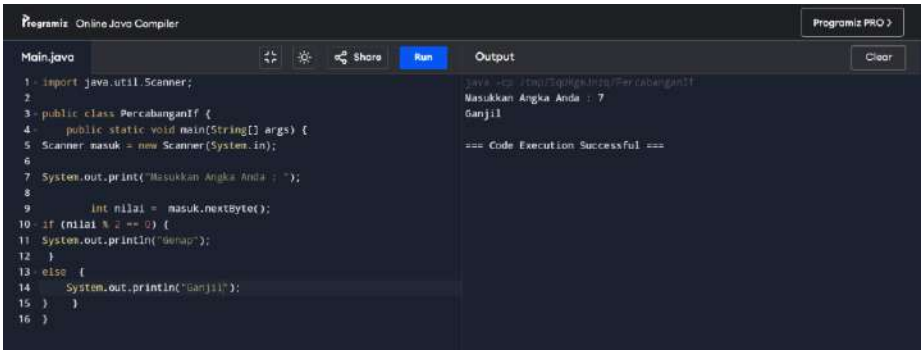
```
import java.util.Scanner;

public class IfBersarang {
    public static void main(String[] args) {
        Scanner varT = new Scanner(System.in);
        System.out.print("Masukkan Angka Tugas Anda : ");
        int nilaiT =  varT.nextByte();

        Scanner varQ = new Scanner(System.in);
        System.out.print("Masukkan Angka Quiz Anda : ");
        int nilaiQ =  varQ.nextByte();

    if (nilaiU >= 80) {
        if(nilaiT >= 80) {
            System.out.println("Anda mendapatkan nilai A");
        }
    }
    else{
        System.out.println("Anda TIDAK mendapatkan nilai A");
    }
    }
}
```

Luaran Contoh 2:

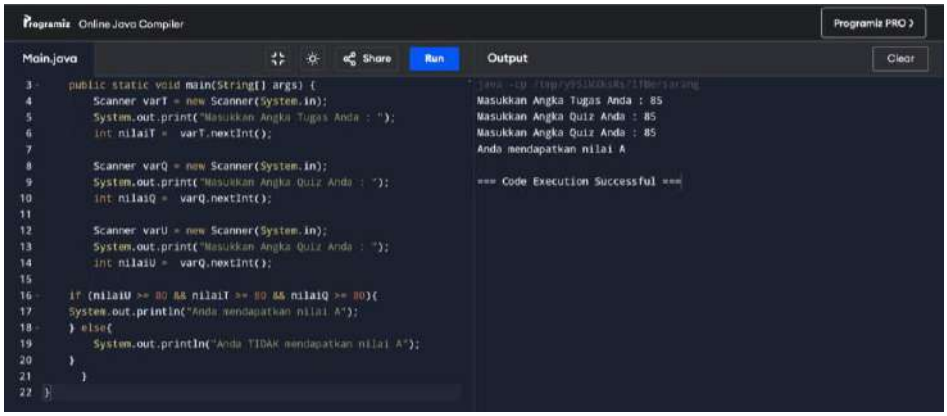
```
Masukkan Angka Tugas Anda : 70
Masukkan Angka Quiz Anda : 70
Masukkan Angka UTS Anda : 70
Anda TIDAK mendapatkan nilai A
```

Latihan 1:

1.1. 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)



1.2. 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.



1.3. 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)`

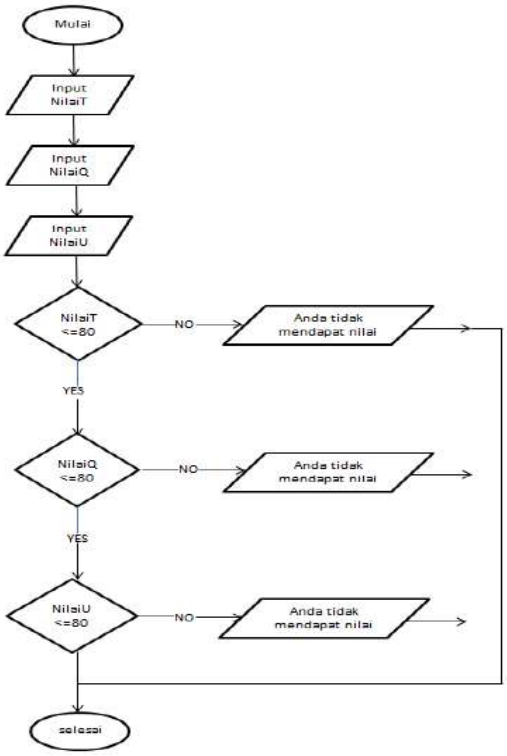
Jawaban yang tepat adalah yang `IF (nilaiU >= 80 && nilaiT >= 80 && nilaiQ >= 80)`

Berikut tampilannya:

```
import java.util.Scanner;

public class IfBersarang {

    public static void main(String[] args) {

        Scanner varT = new Scanner(System.in);

        System.out.print("Masukkan Angka Tugas Anda : ");

        int nilaiT =  varT.nextInt();
```

```
        Scanner varQ = new Scanner(System.in);
        System.out.print("Masukkan Angka Quiz Anda : ");
        int nilaiQ =  varQ.nextInt();

        Scanner varU = new Scanner(System.in);
        System.out.print("Masukkan Angka Quiz Anda : ");
        int nilaiU =  varQ.nextInt();

    if (nilaiU >= 80 && nilaiT >= 80 && nilaiQ >= 80){
    System.out.println("Anda mendapatkan nilai A");
    } else{
        System.out.println("Anda TIDAK mendapatkan nilai A");
    }
      }
}
```

1.4. Uraikan gambar diagram flowchart dari Latihan 1.2!



2) Rincikan sumber informasi yang relevan (buku / webpage)

Video Materi 1 tentang IF – https://www.youtube.com/watch?v=G0dfdAFa9iM

Video Materi 2 tentang SWITCH – https://www.youtube.com/watch?v=RB4nz4xkisM

[N0.1] Analisis dan Argumentasi

1) Uraikan rancangan solusi yang diusulkan.

Contoh 1 : Berikut adalah "program" Java yang menentukan apakah suatu bilangan, yang dimasukkan oleh pengguna, adalah genap atau ganjil dengan menggunakan percabangan if. Langkah pertama adalah mengimpor kelas Scanner untuk membaca input dari konsol. Kemudian, dalam metode main, objek Scanner dibuat dan pengguna diminta untuk tepat sebuah bilangan. Bilangan itu kemudian dibaca menggunakan nextByte() yang dapat dijadikan nextInt() dan sangat dianjurkan karena input dan keluaran yang mampu menangani angka yang lebih besar dari 127. Lalu program check apakah bilangan itu genap atau tidak. Hal ini dilakukan dengan menggunakan operator modulus % untuk memeriksa sisa bagi. Jika nol, maka program mencetak "Bilangan Genap". Jika tidak program mengeluarkan output "Bilangan Ganjil". Ini adalah bagian yang program. Jika kode ini sebenarnya program sebenar, maka harus ditambahkan penanganan kesalahan input untuk meningkatkan robustasitasnya.

Contoh 2: Solusi program ini bertujuan untuk mengevaluasi apakah pengguna mendapatkan nilai A dari tiga input: nilai tugas, nilai quiz, dan nilai ujian. Program dimulai dengan mengimpor kelas Scanner untuk membaca input. Dalam metode main, objek Scanner dibuat untuk menerima nilai tugas, nilai quiz pertama, dan nilai ujian, yang seharusnya mendapatkan nilai ujian dengan menggunakan objek varU (bukan varQ). Setelah menerima semua input, program menggunakan percabangan untuk menerima nilai ujian. Jika semua nilai memenuhi syarat, program mencetak "Anda mendapatkan nilai A", tetapi jika tidak, mencetak "Anda TIDAK mendapatkan nilai A". Desain ini dapat disederhanakan dengan menggunakan satu objek Scanner untuk semua input

2) Analisis solusi, kaitkan dengan permasalahan.

Contoh 1 : Program Java adalah solusi sederhana untuk masalah menentukan apakah angka yang dimasukkan oleh pengguna genap atau ganjil. Menggunakan percabangan `if, program memeriksa selisih dari modulus angka ketika dibagi oleh dua. Jika selisih itu nol, maka angka ganjil; sebaliknya, jika selisih tersebut bukan nol, maka angka ganjil. Program ini mungkin benar-benar berfungsi dalam parameter yang disebutkan, tetapi beberapa masalah potensial tetap ada. Salah satunya adalah penggunaan metode `nextByte();, yang membatasi rentang angka pengguna hingga -128-127. Jika pengguna memutuskan memasukkan angka di luar jangkauan ini, program akan gagal. Selain itu, dengan tidak menangani input yang salah, program akan crash selama pengguna mencoba memasukkan karakter non-numerik. Oleh karena itu, bahkan solusi yang sederhana dan jelas ini perlu ditetapkan agar aman dan efektif.

Contoh 2 : Analisis solusi ini menunjukkan kelebihan dan kekurangan. Kelebihannya adalah program memungkinkan penilaian kinerja pengguna yang mudah dengan input numerik. Meskipun demikian, ada beberapa masalah. Pertama, satu objek scanner cukup untuk membaca semua input, jadi menggunakan tiga objek tidak efisien. Kedua, ada kesalahan dalam pembacaan nilai ujian yang seharusnya menggunakan varU tetapi malah menggunakan varQ. Selain itu, tidak ada penanganan kesalahan input, yang dapat menyebabkan program crash jika pengguna memasukkan data yang tidak valid, seperti huruf daripada angka. Oleh karena itu, meskipun program memenuhi tujuan dasar, perbaikan pada kinerja dan penanganan kesalahan sangat diperlukan untuk meningkatkan kepercayaan pengguna dan kinerja program.

[No.1] Penyusunan Algoritma dan Kode Program

1) Rancang desain solusi atau algoritma

Contoh 1 :

- Mulai
- Impor Scanner
- Deklarasi kelas PercabanganIf

Di dalam main:

- Buat objek Scanner

Cetak "Masukkan Angka Anda: "

- Baca input sebagai integer

Jika (input % 2 == 0) Maka

- Cetak "Bilangan genap"

Jika Tidak

Cetak "Bilangan ganjil"

- Selesai

Contoh 2 :

- Mulai
- Input nilaiT
- Input nilaiQ
- Input nilaiU
- Jika (nilaiT >= 80 DAN nilaiQ >= 80 DAN nilaiU >= 80)

Cetak "Anda mendapatkan nilai A"

- Jika tidak

Cetak "Anda TIDAK mendapatkan nilai A"

- Tutup Scanner
- Akhiri

2) Tuliskan kode program dan luaran
   a) Beri komentar pada kode

Contoh 1 :

Kode program di atas dimulai dengan mengimpor kelas Scanner dari paket java.util. Kelas ini memberi program kemampuan untuk membaca input pengguna. Kelas utama PercabanganIf dibuat, dan objek Scanner dibuat dalam metode main untuk menerima input dari konsol. Pengguna diminta untuk mencetak angka di layar. Setelah input dibaca sebagai tipe data byte dan disimpan dalam variabel nilai, program menggunakan percabangan if untuk mengetahui apakah nilai adalah genap. Dengan menggunakan operator modulus (%), program mencetak "Bilangan genap" jika hasilnya nol, dan jika hasilnya tidak, program mencetak "Bilangan ganjil". Meskipun kode ini berfungsi dengan baik, ada beberapa hal yang dapat diperbaiki. Misalnya, Anda dapat menambahkan mekanisme penanganan kesalahan untuk meningkatkan ketahanan dan keandalan serta mengganti nextByte() dengan nextInt() untuk menangani input yang lebih besar.

Contoh 2 :

Kode program di atas dimulai dengan mengimpor kelas "Scanner" dari paket "java.util". Ini memberi program kemampuan untuk membaca input pengguna. Kelas utama, "IfBersarang", ditetapkan, dan dalam metode "main", objek "Scanner" pertama dibuat untuk menerima input nilai tugas dan kemudian disimpan dalam variabel "nilaiT". Kemudian, objek "Scanner" kedua dibuat untuk membaca nilai quiz pertama dan disimpan dalam variabel "nilaiQ". Namun, terjadi kesalahan karena nilai ujian seharusnya dibaca menggunakan objek " Program menggunakan percabangan "if" untuk mengetahui apakah semua nilai (tugas, quiz, dan ujian) lebih besar atau sama dengan 80 setelah semua nilai dimasukkan. Program mencetak "Anda mendapatkan nilai A" jika semua memenuhi syarat; jika tidak, program mencetak "Anda TIDAK mendapatkan nilai A".

b) Uraikan luaran yang dihasilkan

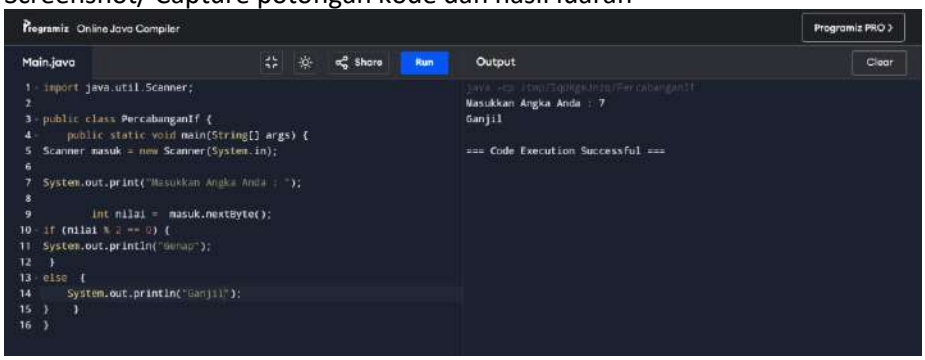
Contoh 1 :

Masukkan Angka Anda : 7
Bilangan ganjil

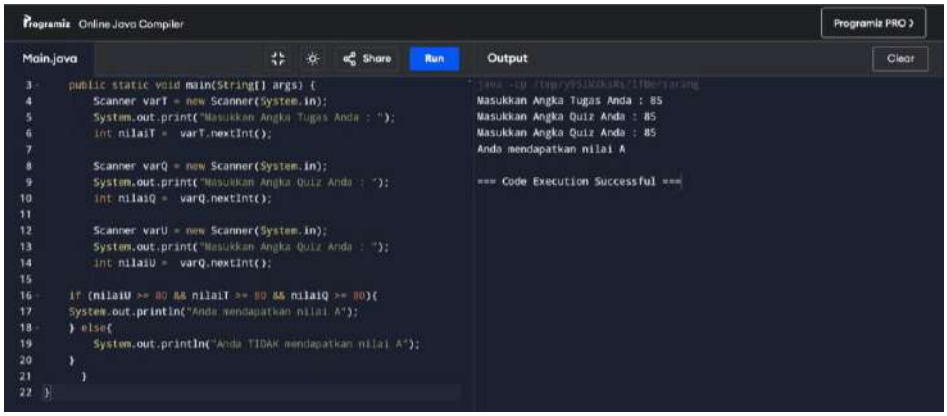
Contoh 2 :

Masukkan Angka Tugas Anda : 85
Masukkan Angka Quiz Anda : 85
Masukkan Angka Quiz Anda : 85
Anda mendapatkan nilai A

c) Screenshot/ Capture potongan kode dan hasil luaran



(gambar contoh 1)



(gambar contoh 2)

[No1] Kesimpulan

1) Evaluasi
   a) Apa konsekuensi dari skenario pemprograman ini?

Contoh 1 :

Konsekuensi dari situasi pemrograman ini mencakup berbagai aspek. Pertama, karena nextByte() membatasi input pengguna di antara -128 dan 127, program dapat gagal jika pengguna memasukkan angka di luar batas ini. Kedua, program tidak memiliki penanganan kesalahan untuk menangani input yang tidak valid, seperti karakter non-numerik, yang dapat menyebabkan crash saat dijalankan. Selain itu, meskipun program memiliki kemampuan untuk memilih bilangan genap atau ganjil, memiliki batasan pada jenis input yang dapat diterima dapat menyebabkan program menjadi kurang fleksibel dan tidak berguna secara keseluruhan. Oleh karena itu, untuk meningkatkan keandalan dan pengalaman pengguna, peningkatan dalam pengelolaan input dan penanganan kesalahan sangat penting.

Contoh 2 :

Situasi pemrograman ini memiliki beberapa akibat penting. Pertama, penggunaan beberapa instance "Scanner" untuk membaca input pengguna tidak perlu; hanya satu instance dapat dioptimalkan, meningkatkan efisiensi dan mengurangi kebingungan. Selain itu, penting untuk menambahkan pengecekan input jika pengguna memasukkan nilai yang tidak sah (seperti karakter bukan angka), karena program dapat mengalami kesalahan jika mereka memasukkan nilai yang tidak valid. Logika penilaian yang hanya menggunakan nilai di atas 80 untuk mendapatkan nilai A mungkin tidak sesuai dengan kebijakan penilaian yang lebih kompleks, jadi kriteria untuk nilai lainnya harus ditambahkan. Selain itu, duplikasi kode saat meminta input membuat kode lebih sulit dibaca dan dipelihara. Program dapat menjadi lebih efisien, lebih mudah dipahami, dan lebih tahan terhadap kesalahan input dengan memperbaiki elemen-elemen

Template Lembar Kerja Individu

Nama & NPM	Topik:	Tanggal:
ANGGUN SYAVIRA TRINANDA (G1F024071)	IF dan SWITCH Java	24 September 2024

[No. 2] Identifikasi Masalah:

1) Uraikan permasalahan dan variabel

Contoh 3: Salin dan tempel kode program berikut ke Eclipse.

```
import java.util.Scanner;

public class SwitchBersarang {
    public static void main(String[] args) {
        Scanner masukData = new Scanner(System.in);
        // mengambil input
        System.out.print("Pilih A atau B :  ");
        char data = masukData.next().charAt(0);
        switch(data) {
        case 'A':
            System.out.print("Anda sudah rajin belajar");
            break; // baris 1
        case 'B':
            System.out.print("  Anda perlu kurangi main game");
            break; // baris 2
        default:
            System.out.print("  Pilihan anda diluar A atau B ");
}      }     }
```

Luaran Contoh 3:

```
Pilih A atau B :  A
Anda sudah rajin belajar
```

Contoh 4: Salin dan tempel kode program berikut ke Eclipse.

```
import java.util.Scanner;

public class SwitchBersarang {

    public static void main(String[] args) {
             byte bulan;
            int tahun = 2022;
            int jumlahHari = 0;
            System.out.print("Masukkan data bulan (dalam angka): ");
            Scanner masukData = new Scanner(System.in);
            bulan = masukData.nextByte();

            switch (bulan) {
                case 1: case 3:   //baris 1
                    jumlahHari = 31;
                    break;
                case 4: //baris 2
                    jumlahHari = 30;
                    break;
                case 2:
                    if (tahun % 4 == 0)
                        jumlahHari = 29;
                    else
                        jumlahHari = 28;
                    break;
                default:
                    System.out.println("Maaf bulan hanya sampai 12.");
                    break;
            }
```

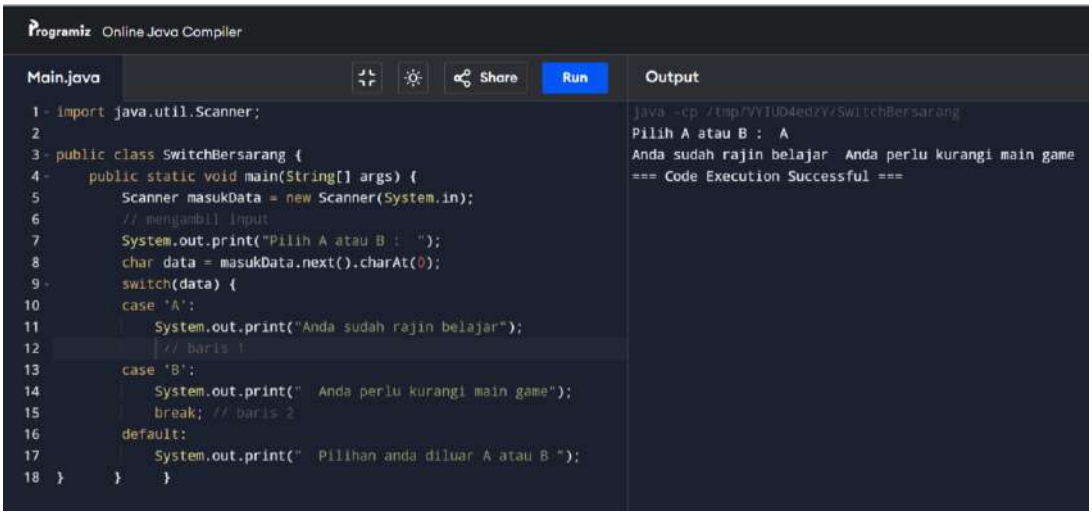
```
            System.out.println("Jumlah hari = " + jumlahHari);
    }
}
```

Luaran Contoh 4:

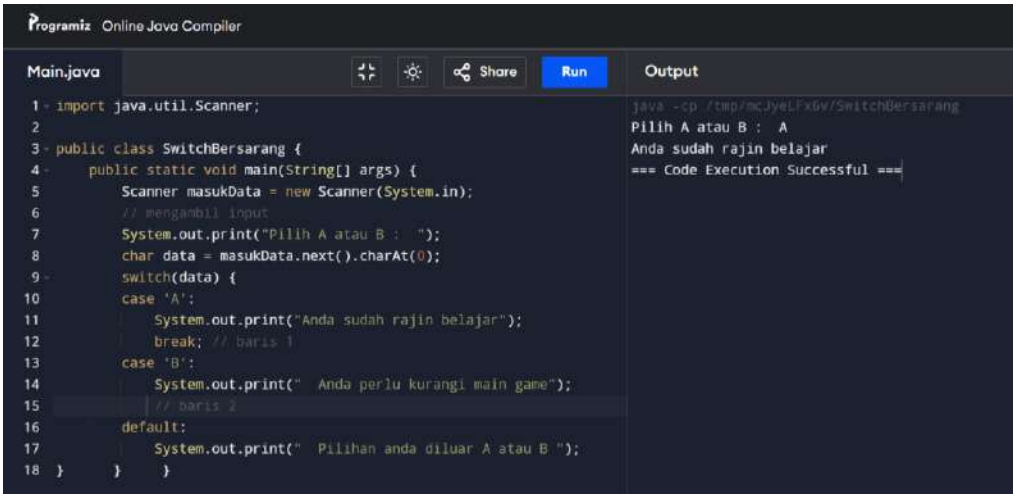
```
Masukkan data bulan (dalam angka): 7
Jumlah hari = 31
```

Latihan 2:

2.1. 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`!



(tampilan kode break; pada// baris 1 dihapus)



tampilan kode break; pada// baris 2 dihapus)

perbedaan hasil luaran ketika kode menggunakan `break`, ketika kode `break` baris 1 dihapus, dan ketika kode `break` baris 2 dihapus terlihat jelas terletak pada hasil luarannya berikut penjelasannya:

jika kode break di baris 1 (pada case 'A') dihapus:

- Hasil: Setelah menampilkan output "Anda sudah rajin belajar", eksekusi akan terus berlanjut ke case 'B' karena tidak ada perintah break untuk menghentikan eksekusi pada case 'A'. Ini berarti pesan dari case 'B' juga akan dicetak, yaitu "Anda perlu kurangi main game", meskipun pengguna

memilih 'A'.

- Penjelasan: Tanpa break, eksekusi akan fall through, artinya akan berlanjut ke case berikutnya tanpa memeriksa nilai case tersebut.

Jika kode break di baris 2 (pada case 'B') dihapus:

- Hasil: Jika pengguna memilih 'B', program akan mencetak "Anda perlu kurangi main game", namun setelah itu, karena tidak ada perintah break, eksekusi akan berlanjut ke blok default, sehingga pesan "Pilihan anda di luar A atau B" juga akan dicetak.

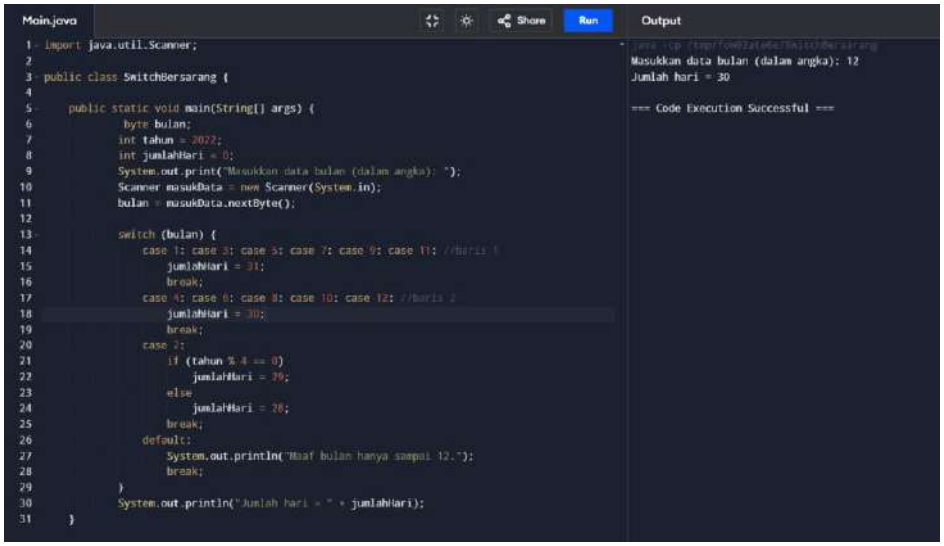
- Penjelasan: Tanpa break, setelah mengeksekusi case 'B', program akan terus mengeksekusi blok default, meskipun case yang cocok sudah ditemukan

kegunaan baris kode dengan `break` dan kata kunci `default`!

- Break: Berfungsi untuk menghentikan eksekusi pada suatu case setelah case tersebut dijalankan. Ini mencegah eksekusi berlanjut ke case berikutnya (fall-through) yang bisa menghasilkan output yang tidak diinginkan.

- Default: Digunakan untuk menangani nilai input yang tidak sesuai dengan case yang didefinisikan. Ini memberikan tanggapan atau tindakan jika tidak ada case yang cocok, mirip dengan "else" dalam struktur if-else.

2.2. 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.

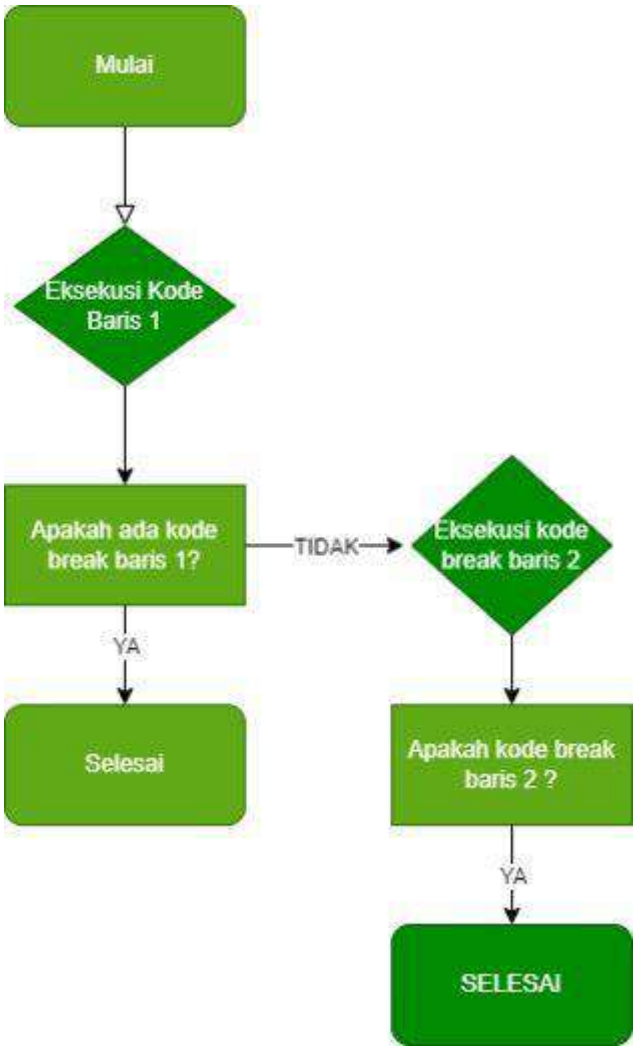


2.3. Cermati permasalahan yang dipecahkan pada Contoh 3.
Apakah masalah ini bisa diubah menjadi perintah `IF`?
Jika bisa, rincikan analisa Anda!
Bandingkan masalah yang dapat diselesaikan percabangan dengan `IF` atau `SWITCH` !

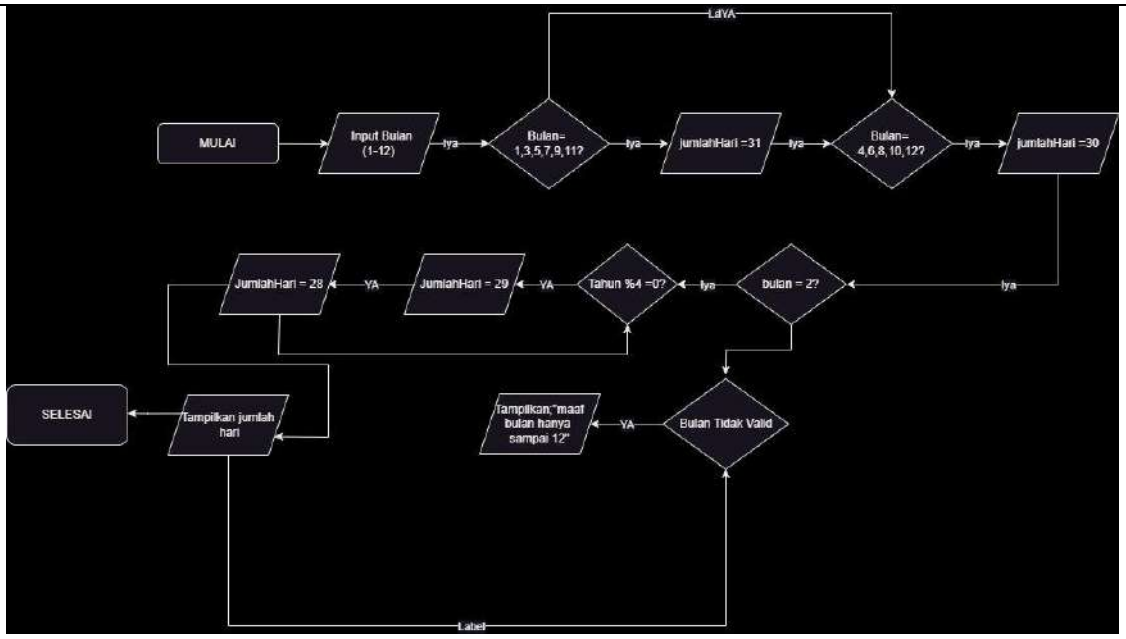
Ya, Masalah ini dapat diubah menjadi perintah `if`, yang memungkinkan kita untuk memeriksa kondisi yang sama seperti yang dilakukan dengan `switch`. Misalnya, dengan menggunakan struktur `if`, kita dapat memeriksa apakah input pengguna adalah 'A' atau 'B', dan memberikan respons yang sesuai. Perbandingan antara penggunaan `if` dan `switch` menunjukkan bahwa `switch` lebih jelas dan terstruktur untuk situasi dengan banyak pilihan tetap, membuatnya lebih mudah dibaca saat ada banyak opsi. Namun, `if` menawarkan fleksibilitas yang lebih besar, karena dapat menangani kondisi kompleks, termasuk rentang nilai dan ekspresi logika. Dalam hal kinerja, meskipun `switch` dapat lebih cepat dalam beberapa kasus karena optimasi, perbedaannya biasanya tidak signifikan untuk jumlah kondisi yang kecil. Dalam praktiknya, `switch` sering

digunakan untuk pilihan sederhana, sedangkan `if` lebih baik untuk kondisi yang lebih rumit. Kesimpulannya, baik `if` maupun `switch` dapat digunakan untuk menyelesaikan masalah ini, dan pilihan antara keduanya bergantung pada konteks dan kompleksitas kondisi yang perlu diperiksa.

2.4. Buatlah dokumentasi gambar flowchart dari Latihan 2.1. dan Latihan 2.2!



(GAMBAR 2.1)



(Gambar 2.2)

2) Rincikan sumber informasi yang relevan (buku / webpage)

Video Materi 1 tentang IF – https://www.youtube.com/watch?v=G0dfdAFa9iM

Video Materi 2 tentang SWITCH – https://www.youtube.com/watch?v=RB4nz4xkisM

[N0.2] Analisis dan Argumentasi

1) Uraikan rancangan solusi yang diusulkan.

Contoh 3 : Tujuan dari rancangan solusi kode ini adalah untuk meminta pengguna memilih antara huruf "A" atau "B" dan kemudian memberikan respons berdasarkan pilihan mereka. Dimulai dengan mengimpor kelas pemindai untuk membaca input pengguna. Kemudian, aplikasi meminta pengguna memasukkan karakter yang diinginkan. Struktur switch kemudian digunakan untuk mengevaluasi input. Setiap pilihan, yang disebut sebagai "A" atau "B", memiliki tanggapan unik. Misalnya, jika pengguna memilih opsi "A", program akan mencetak pesan bahwa mereka telah rajin belajar, dan jika mereka memilih opsi "B", program akan memberi tahu mereka harus mengurangi waktu bermain game. Jika mereka memasukkan karakter lain, program akan menampilkan pesan bahwa pilihan mereka di luar opsi yang tersedia. Oleh karena itu, solusi ini memberikan umpan balik yang jelas berdasarkan input.

Contoh 4:

Tujuan dari rancangan solusi ini adalah untuk menggunakan input bulan pengguna untuk menghitung jumlah hari dalam sebulan. Pertama, variabel bulan bertipe byte dan variabel jumlahHari bertipe int diinisialisasi. Tahun diatur ke 2022. Kemudian, aplikasi meminta pengguna memasukkan nomor bulan dan menggunakan objek Scanner untuk membaca input tersebut. Struktur switch menggunakan nilai bulan untuk menentukan jumlah hari. Bulan dengan 31 hari ditetapkan, bulan dengan 30 hari diatur, dan pada Februari, program memeriksa apakah tahun tersebut kabisat untuk menentukan jumlah hari. Pesan kesalahan ditampilkan oleh program jika input tidak valid. Pengguna melihat jumlah hari dalam bulan yang dipilih setelah evaluasi.

2) Analisis solusi, kaitkan dengan permasalahan.

Contoh 3: Analisis solusi ini menunjukkan bahwa program menangani masalah pemilihan secara

efektif. Dengan struktur yang jelas dan langsung, switch memungkinkan program untuk dengan cepat menentukan respons yang sesuai berdasarkan input pengguna. Jika pengguna memasukkan karakter yang tidak valid, program akan memberikan umpan balik yang sesuai, yang mengurangi kemungkinan kebingungan. Selain itu, penggunaan switch sangat cocok di sini karena ada dua opsi yang jelas dan satu opsi default, yang membuatnya lebih mudah dibandingkan dengan menggunakan beberapa pernyataan if. Namun, metode ini dapat diperluas untuk menangani lebih banyak opsi jika diperlukan. Secara keseluruhan, solusi ini memberikan pengalaman pengguna yang baik dan mengatasi masalah pemilihan dengan baik.

Contoh 4 : Menurut analisis solusi ini, program menyelesaikan masalah penghitungan jumlah hari dengan pendekatan terstruktur dan menggunakan switch untuk memudahkan evaluasi. Namun, program menangani kesalahan input dengan buruk; kesalahan akan terjadi jika pengguna memasukkan nilai yang tidak numerik atau di luar rentang 1-12. Menambah validasi input sebelum penilaian akan meningkatkan pengalaman pengguna dan mengurangi kesalahan. Secara keseluruhan, meskipun program sudah berfungsi dengan baik, memiliki fitur untuk menangani kesalahan input dapat membuatnya lebih kuat dan lebih mudah digunakan.

[No.2] Penyusunan Algoritma dan Kode Program

1) Rancang desain solusi atau algoritma

Contoh 3 :

- Mulai
- Import kelas
- Deklarasi Objek Scanner
- Tampilkan Pesan Permintaan Input
- Baca Input Pengguna
- Evaluasi Input Menggunakan Switch
- Selesai

Contoh 4 :

- Mulai
- Inisialisasi Variabel
- Input Pengguna
- Evaluasi Menggunakan Switch
- Output Hasil
- Akhiri program

2) Tuliskan kode program dan luaran

a) Beri komentar pada kode

Contoh 3 :

Kode ini dimulai dengan mengimpor kelas "Scanner", yang berfungsi untuk membaca informasi yang dimasukkan pengguna. Di metode "main", objek "Scanner" digunakan untuk mengumpulkan input, dan program meminta pengguna untuk memilih antara "A" atau "B". Input pengguna disimpan dalam variabel "data", yang kemudian dievaluasi menggunakan struktur "switch". Dalam "switch", setiap case memberikan tanggapan yang berbeda. Jika pengguna memilih "A", program menampilkan pesan bahwa mereka sudah rajin belajar, dan jika pengguna memilih "B", program menyarankan agar mereka mengurangi waktu bermain game. Jika input tidak sesuai dengan kedua pilihan, pesan default akan ditampilkan, memberi tahu pengguna bahwa pilihan mereka di luar opsi yang tersedia. Untuk menjaga alur program tetap teratur dan jelas, penggunaan pernyataan "break" setelah setiap kasus memastikan bahwa eksekusi program berhenti setelah menerima respons.

Contoh 4:

Program ini dimaksudkan untuk menghitung jumlah hari dalam sebulan dengan menggunakan

input bulan pengguna. Untuk memulai, kode mengimpor kelas "Scanner" untuk memudahkan pembacaan input dari konsol. Untuk menyimpan informasi yang diperlukan, variabel "bulan", "tahun", dan "jumlah hari" dibuat. Setelah pengguna memasukkan nomor bulan, program menggunakan struktur "switch" untuk menghitung nilai "bulan". Setiap kasus dalam "switch" mengatur "jumlahHari" berdasarkan jumlah hari yang sesuai untuk setiap bulan, dengan perhatian khusus pada bulan Februari, yang menentukan apakah tahun itu kabisat atau tidak. Program menampilkan pesan kesalahan yang jelas jika pengguna memasukkan nilai yang tidak valid. Akhirnya, pengguna melihat hasil jumlah hari. Secara keseluruhan, kode ini terstruktur dengan baik dan menyelesaikan masalah penghitungan hari dan bulan dengan mudah, tetapi masih ada ruang untuk menambah validasi input untuk meningkatkan pengalaman pengguna.

b) Uraikan luaran yang dihasilkan

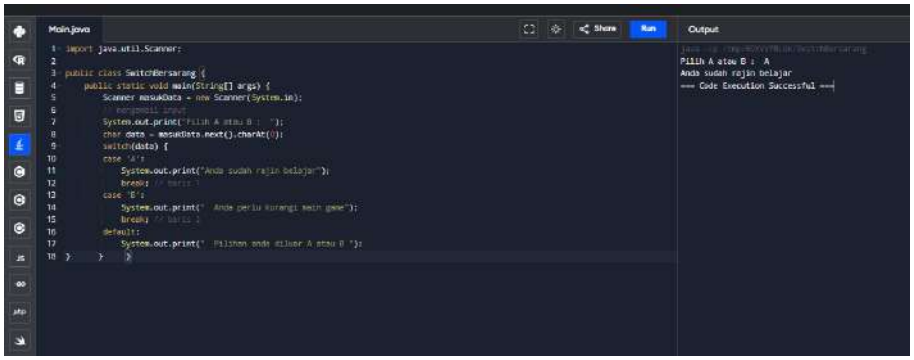
Contoh 3 :

Pilih A atau B : A
Anda sudah rajin belajar
=== Code Execution Successful ===

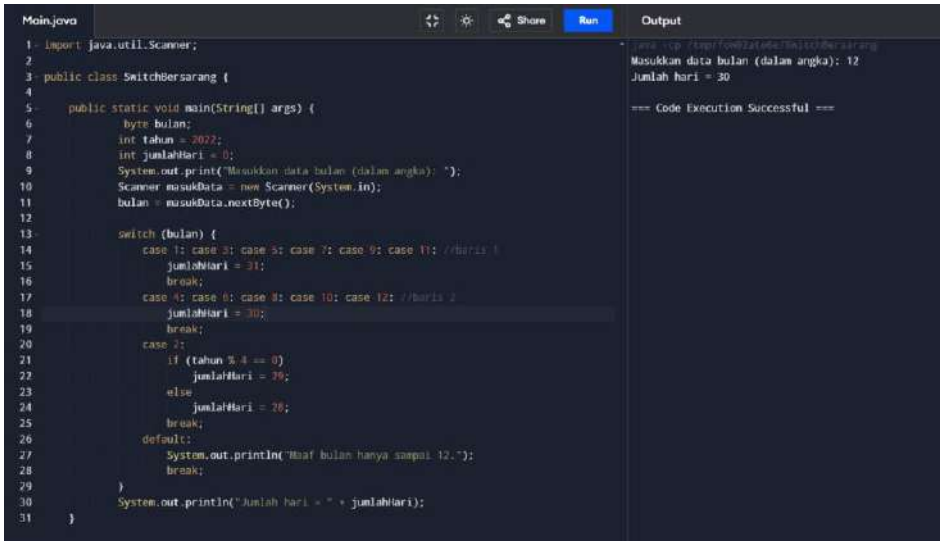
Contoh 4 :

Masukkan data bulan (dalam angka): 12
Jumlah hari = 30
=== Code Execution Successful ===

c) Screenshot/ Capture potongan kode dan hasil luaran



(contoh 3)



(contoh 4)

[No2] Kesimpulan

1) Evaluasi

a) Apa konsekuensi dari skenario pemprograman ini?

Contoh 3 :

Konsekuensi dari skenario pemrograman ini mencakup beberapa aspek penting. Program memberikan umpan balik yang jelas kepada pengguna berdasarkan pilihan mereka, yang meningkatkan interaksi. Namun, jika pengguna memasukkan karakter yang tidak valid, pesan default yang ditampilkan mungkin tidak cukup informatif, berpotensi menyebabkan kebingungan. Penggunaan struktur `switch` membuat evaluasi input lebih mudah dan terorganisir, tetapi menambah kompleksitas jika lebih banyak opsi ingin ditambahkan di masa depan. Program ini juga tidak menangani kesalahan input, seperti karakter tambahan, yang bisa menghasilkan perilaku tidak diinginkan. Secara keseluruhan, meskipun program ini sederhana dan fungsional, ada peluang untuk meningkatkan validasi input dan penanganan opsi agar pengalaman pengguna lebih baik.

Contoh 4 :

Konsekuensi dari skenario pemrograman ini meliputi beberapa aspek penting. Pertama, program ini memberikan fungsionalitas yang jelas dan mudah dipahami bagi pengguna, memungkinkan mereka untuk mengetahui jumlah hari dalam sebulan dengan cepat berdasarkan input yang diberikan. Struktur switch yang digunakan membuat kode terorganisir dan efisien dalam mengelola berbagai kondisi. Namun, program ini juga memiliki keterbatasan; jika pengguna memasukkan nilai yang tidak valid (seperti nilai non-numerik atau angka di luar rentang 1-12), maka hasilnya tidak akan ditangani dengan baik, yang dapat menyebabkan kebingungan. Selain itu, meskipun program dapat menghitung jumlah hari dengan benar, tidak adanya validasi input dapat mengakibatkan kesalahan runtime jika input tidak sesuai dengan yang diharapkan. Hal ini menunjukkan perlunya pengembangan lebih lanjut untuk menambahkan fitur validasi yang lebih baik. Secara keseluruhan, meskipun program berfungsi sesuai tujuannya, peningkatan pada penanganan kesalahan dan validasi input dapat meningkatkan keandalan dan pengalaman pengguna secara signifikan.