|  |  |  |
| --- | --- | --- |
| **Nama & NPM** | **Topik:** | **Tanggal:** |
| **Ariansyah Putra Ramadhan**  **G1F024080** | **If dan switch** | **10 Oktober 2024** |

**[No. 1] Identifikasi Masalah:**

1.1. Analisa tujuan penulisan kata kunci continue dan break pada Contoh 1!

Buat perubahan nilai angka pada variabel di

//Ubah 1 menjadi for (int y = 0; y <= 15; y++) { lalu running, periksa hasilnya

//Ubah 2 menjadi if (y % 2 == 0) lalu running, periksa hasilnya

//Ubah 3 menjadi else if (y == 9) lalu running, periksa hasilnya

Analisa dampaknya perubahan ini terhadap luaran setelah running!

Contoh 1:

public class ContohFor{   
public static void main(String[] args) {  
        for (int y = 0; y <= 10; ++y) {        //ubah 1  
            if (y % 2 == 1)             //ubah 2  
            continue;      //baris 1  
            else if (y == 8)           //ubah 3  
            break;            //baris 2  
            else  
            System.out.println(y + " ");  
    }    }    }

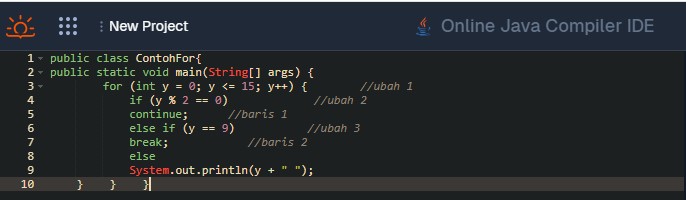
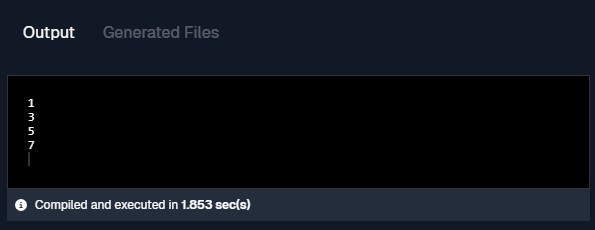
**[No.1] Analisis dan Argumentasi**

Tujuan continue digunakan untuk melewati iterasi saat ini dari loop dan melanjutkan ke iterasi berikutnya. Dalam konteks ini, jika y adalah angka ganjil (y % 2 == 1), maka pernyataan continue akan mengeksekusi, dan program akan langsung melanjutkan ke langkah berikutnya dalam loop for, tanpa mengeksekusi pernyataan yang ada setelahnya (yaitu System.out.println(y + " ");).

Tujuan: break digunakan untuk menghentikan eksekusi loop sepenuhnya. Dalam konteks ini, jika nilai y mencapai 8, pernyataan break akan dieksekusi, dan program akan keluar dari loop for.

**[No.1 ] Penyusunan Algoritma dan Kode Program**

1. Algoritma
2. Inisialisasi
3. Looping
4. Pemeriksaan Genap atau Ganjil
5. Pemeriksaan Nilai Khusus
6. Menampilkan Angka
7. Akhiri Program
8. Kode program dan luaran



1. Screenshot/ Capture potongan kode dan hasil luaran

Beri komentar pada kode yang di Screenshot

1. Analisa luaran yang dihasilkan   
   Rentang Iterasi Lebar: Dengan mengubah batas atas loop menjadi 15, program memeriksa lebih banyak angka.

Melewatkan Angka Genap: Program sekarang mencetak semua angka ganjil dalam rentang yang lebih luas hingga angka 9.

Penghentian Lebih Awal: Dengan menghentikan program di 9, hasil akhirnya hanya menampilkan angka ganjil hingga 7, karena iterasi akan berhenti sebelum mencetak 9.

**[No.1] Kesimpulan**

1. **Evaluasi**
2. Kelebihan:

Struktur program jelas dan mudah dipahami.

Penggunaan kontrol alur (continue dan break) efektif dalam mencapai tujuan.

1. Kekurangan:

Jika tujuan adalah mencetak semua angka ganjil hingga 15, perluasan batas loop menjadi tidak relevan karena akan menghentikan eksekusi di 9.

Program ini bisa lebih jelas jika ditambahkan komentar untuk menjelaskan logika yang digunakan.

Secara keseluruhan, program ini berfungsi sesuai dengan yang dimaksud, tetapi ada ruang untuk perbaikan dalam hal keterbacaan dan tujuan akhir output.

|  |  |  |
| --- | --- | --- |
| **Nama & NPM** | **Topik:** | **Tanggal:** |
| **Ariansyah Putra Ramadhan**  **G1F024080** | **If dan switch** | **10 Oktober 2024** |

**[No. 1] Identifikasi Masalah:**

1.2. Buat perubahan kode pada Contoh 2 di baris //Ubah1 menjadi

a. continue pertama; lalu running, periksa hasilnya

b. break pertama; lalu running, periksa hasilnya

c. continue kedua; lalu running, periksa hasilnya

Analisa perbedaan perubahan kode pada Ubah 1 untuk setiap poin (a), (b), dan (c)!

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

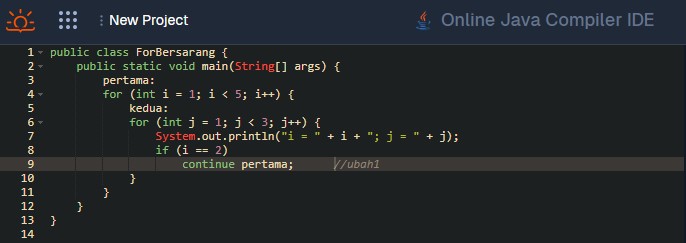
public class ForBersarang {   
        public static void main(String[] args) {  
             pertama:  
                    for( int i = 1; i < 5; i++) {      
                                   
                        kedua:  
                        for(int j = 1; j < 3; j ++ ) {  
                            System.out.println("i = " + i + "; j = " +j);  
                         if ( i == 2)  
                                       break kedua;      //ubah1  
                      }    }    }   }

**[No.1] Analisis dan Argumentasi**

1. Ketika i bernilai 2, perulangan dalam label pertama akan dilanjutkan ke iterasi berikutnya, sehingga j tidak akan dieksekusi untuk i = 2.
2. Ketika i bernilai 2, perulangan dalam label pertama akan dihentikan sepenuhnya, dan program akan keluar dari kedua loop.
3. Ketika i bernilai 2, perulangan dalam label kedua akan dilanjutkan ke iterasi berikutnya. Namun, karena j tidak pernah lebih dari 2, loop kedua untuk i = 2 tidak akan menghasilkan output.

**[No.1 ] Penyusunan Algoritma dan Kode Program**

1. Algoritma
2. Inisialisasi Program
3. Loop Pertama (i)
4. Loop Kedua (j)
5. Akhiri Program
6. Kode program dan luaran





1. Screenshot/ Capture potongan kode dan hasil luaran

Beri komentar pada kode yang di Screenshot

1. Analisa luaran yang dihasilkan   
   Program ini menunjukkan penggunaan loop bersarang dengan penggunaan label dan pernyataan break yang spesifik.

Dalam program ini, penggunaan label menjadi penting karena memungkinkan pengontrolan alur program saat i mencapai 2.

**[No.1] Kesimpulan**

1. **Evaluasi**

Kelebihan :

Struktur Jelas:

Program memiliki struktur yang jelas dengan penggunaan label, sehingga mudah dipahami alur eksekusinya.

Penggunaan Loop Bersarang:

Contoh penggunaan loop bersarang menunjukkan cara iterasi pada dua variabel secara bersamaan.

Pengendalian Alur yang Efektif:

Penggunaan break dengan label memungkinkan pengendalian alur yang lebih spesifik, menghentikan loop kedua saat kondisi tertentu terpenuhi.

Output yang Informatif:

Output memberikan informasi yang jelas tentang nilai i dan j pada setiap iterasi, memudahkan pemahaman logika program.

Kekurangan :

Penggunaan Label yang Tidak Perlu:

Dalam kasus ini, penggunaan label mungkin tidak diperlukan. Cukup menggunakan break; tanpa label dapat memperjelas kode, mengingat break hanya menghentikan loop terdekat.

Keterbatasan Output:

Program hanya mencetak output saat i = 1 dan i = 2. Nilai j tidak dicetak untuk i = 2 pada iterasi selanjutnya, yang mungkin membingungkan bagi pembaca yang berharap melihat lebih banyak output.

Kekakuan Logika:

Penggunaan kondisi yang mengharuskan loop kedua berhenti pada i = 2 membatasi fleksibilitas program. Jika logika yang lebih kompleks diperlukan, perlu penyesuaian lebih lanjut.

Kurangnya Komentar:

Program kurang memiliki komentar yang menjelaskan logika di balik setiap bagian, yang dapat membantu pembaca yang tidak familiar dengan kode.

|  |  |  |
| --- | --- | --- |
| **Nama & NPM** | **Topik:** | **Tanggal:** |
| **Ariansyah Putra Ramadhan**  **G1F024080** | **If dan switch** | **10 Oktober 2024** |

**[No. 1] Identifikasi Masalah:**

1.3.    Cermati kode contoh 3. Apabila ingin menghasilkan luaran berikut:  
Luaran:  
Masukan Input: 7  
\*\*\*\*\*\*\*  
\*\*\*\*\*\*  
\*\*\*\*\*  
\*\*\*\*  
\*\*\*  
\*\*  
\*  
           Susunlah analisa kode untuk menghasilkan luaran tersebut!

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

import java.util.Scanner;

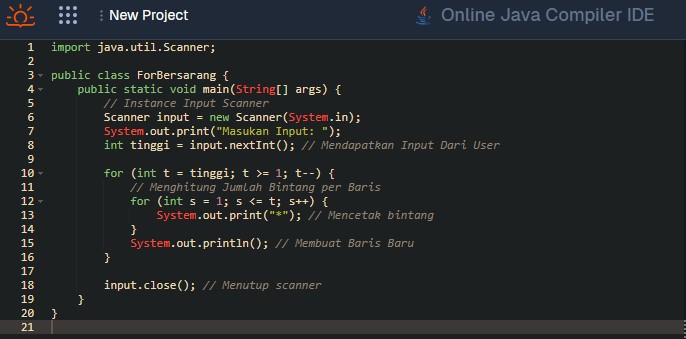
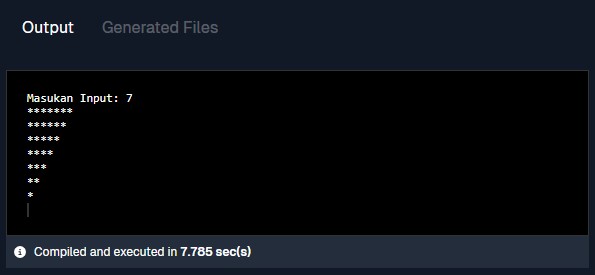
public class ForBersarang {  
        public static void main(String[] args){  
            //Instance Input Scanner  
            Scanner input = new Scanner(System.in);  
            System.out.print("Masukan Input: ");  
            int tinggi = input.nextInt(); //Mendapatkan Input Dari User  
            for(int t=tinggi; t>=1; t--){  
                //Menghitung Jumlah Tinggi Piramida  
                for(int s=tinggi; s>=t; s--){  
                    //Menghitung Jumlah Spasi per Baris  
                    System.out.print("\*");  
                }  
                 System.out.println(); //Membuat Baris Baru  
            }  
        }  
    }

**[No.1] Analisis dan Argumentasi**

Untuk menghasilkan luaran yang diinginkan, kita perlu melakukan beberapa penyesuaian pada kode yang sudah ada. Kode yang diberikan saat ini mencetak bintang dari tinggi yang ditentukan, tetapi tidak mengatur jumlah bintang dengan benar.

**[No.1 ] Penyusunan Algoritma dan Kode Program**

1. Algoritma
2. Inisialisasi
3. Input
4. Loop untuk Baris
5. Tutup Scanner
6. Kode program dan luaran



1. Screenshot/ Capture potongan kode dan hasil luaran

Beri komentar pada kode yang di Screenshot

1. Analisa luaran yang dihasilkan   
   Program ini mencetak pola bintang berbentuk segitiga terbalik. Jumlah baris dan jumlah bintang di setiap baris ditentukan oleh input pengguna. Setiap baris akan memiliki bintang yang lebih sedikit dibandingkan baris di atasnya..

**[No.1] Kesimpulan**

**Evaluasi**

Program Java di atas adalah program yang mencetak pola bintang dalam bentuk segitiga terbalik. Mari kita evaluasi beberapa aspek dari program tersebut:

Fungsi Program :

Input dari Pengguna: Program meminta pengguna untuk memasukkan tinggi segitiga.

Pengulangan Bersarang: Menggunakan dua loop for untuk mencetak bintang:

Loop luar mengatur jumlah baris, dimulai dari tinggi dan berkurang hingga 1.

Loop dalam mencetak bintang pada setiap baris sesuai dengan nilai dari loop luar.

Output: Menghasilkan pola bintang yang berkurang per baris.

Kelebihan

Sederhana dan Jelas: Kode mudah dipahami dengan logika yang jelas.

Menggunakan Scanner: Mengizinkan interaksi pengguna untuk menentukan tinggi segitiga.

Menutup Scanner: Praktik yang baik dengan menutup Scanner setelah digunakan untuk mencegah kebocoran sumber daya.

Kekurangan

Validasi Input: Tidak ada pemeriksaan untuk memastikan bahwa input adalah bilangan bulat positif. Jika pengguna memasukkan nilai negatif atau non-numeric, program akan mengakibatkan kesalahan.

Desain Pola: Pola yang dihasilkan hanya bintang, mungkin bisa ditambahkan spasi atau variasi lain untuk membuat lebih menarik.

|  |  |  |
| --- | --- | --- |
| **Nama & NPM** | **Topik:** | **Tanggal:** |
| **Ariansyah Putra Ramadhan**  **G1F024080** | **If dan switch** | **10 Oktober 2024** |

**[No. 1] Identifikasi Masalah:**

1.4.      Analisa diagram flowchart dari Latihan 1.2 dan 1.3!

**[No.1] Analisis dan Argumentasi**

Inisialisasi Flowchart 1.2:

Program dimulai dengan menginisialisasi variabel i dengan nilai 1. Ini menandakan bahwa kita akan mulai iterasi pada loop pertama.

Loop Pertama (Outer Loop):

Kondisi i < 5: Program memeriksa apakah nilai i kurang dari 5. Jika ya, program melanjutkan ke loop kedua. Jika tidak, program selesai.

Rentang Nilai i: Loop ini akan berjalan untuk nilai i 1, 2, 3, dan 4.

Loop Kedua (Inner Loop):

Inisialisasi j: Di dalam loop pertama, variabel j diinisialisasi dengan 1 pada setiap iterasi i.

Kondisi j < 3: Program memeriksa apakah nilai j kurang dari 3. Jika ya, cetak nilai i dan j, kemudian periksa apakah i sama dengan 2.

Cetak Nilai:

Saat kondisi j < 3 terpenuhi, program mencetak nilai dari i dan j ke konsol.

Pemeriksaan i == 2:

Jika nilai i adalah 2, program akan menghentikan loop kedua (inner loop) menggunakan pernyataan break. Ini berarti bahwa ketika i mencapai 2, program tidak akan melanjutkan untuk j yang lebih besar dan akan langsung kembali ke loop pertama.

Increment j:

Jika i tidak sama dengan 2, program akan mengincrement j dan kembali ke kondisi j < 3. Jika j mencapai 3, loop kedua akan berhenti.

Increment i:

Setelah loop kedua selesai, program akan mengincrement i dan kembali ke kondisi i < 5 untuk iterasi berikutnya.

Selesai:

Ketika semua iterasi selesai (setelah i mencapai 5), program berakhir.

Berikut adalah analisis flowchart dari latihan 1.3:

1. Inisialisasi dan Input

Proses Inisialisasi: Program dimulai dengan menginisialisasi objek Scanner untuk menerima input dari pengguna. Ini penting untuk interaksi pengguna.

Pengambilan Input: Pengguna diminta untuk memasukkan nilai tinggi, yang menentukan jumlah baris bintang yang akan dicetak.

2. Looping Eksternal

Looping dari tinggi ke 1: Program menggunakan loop for untuk iterasi dari nilai tinggi hingga 1. Ini memungkinkan program mencetak bintang dalam bentuk terbalik (banyak bintang di atas, sedikit di bawah).

Kontrol Pengulangan: Variabel t digunakan untuk mengontrol berapa banyak bintang yang akan dicetak di setiap baris.

3. Looping Internal

Looping dari 1 ke t: Di dalam loop eksternal, ada loop lain yang mencetak bintang. Loop ini berjalan sebanyak nilai t, sehingga baris pertama (tinggi) memiliki bintang sebanyak tinggi, dan setiap baris berikutnya memiliki satu bintang lebih sedikit.

Cetak Bintang: Proses ini bertanggung jawab untuk mencetak karakter bintang ke layar. Setiap iterasi mencetak satu bintang hingga mencapai t.

4. Baris Baru

Pindah ke Baris Baru: Setelah mencetak bintang di satu baris, program memanggil System.out.println() untuk memindahkan kursor ke baris berikutnya. Ini memastikan bahwa bintang-bintang di baris berikutnya dicetak di baris yang terpisah.

5. Penyelesaian

Mengakhiri Loop: Setelah loop eksternal selesai (ketika t menjadi kurang dari 1), program berakhir. Tidak ada output lebih lanjut, dan program akan kembali ke sistem operasi.

|  |  |  |
| --- | --- | --- |
| **Nama & NPM** | **Topik:** | **Tanggal:** |
| **Ariansyah Putra Ramadhan**  **G1F024080** | **If dan switch** | **10 Oktober 2024** |

**[No. 1] Identifikasi Masalah:**

2.1.  Buat perubahan nilai angka pada variabel di Contoh 4  
        //Ubah 1 menjadi continue;  lalu running, periksa hasilnya  
        Analisa dampaknya perubahan terhadap luaran setelah running dan uraikan kegunaan break dan continue!

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

public class ContohWhile{   
public static void main(String[] args) {  
    int i=1;  
        while(i<=6){  
            System.out.println(i);  
            i++;  
            if(i==4){               
                break;        //ubah1   
            }}}

**[No.1] Analisis dan Argumentasi**

Setelah Perubahan (dengan continue):

Program akan mencetak angka 1, 2, dan 3.

Ketika i menjadi 4, continue akan mengabaikan sisa iterasi saat itu dan langsung kembali ke awal loop.

Karena i sudah ditambahkan menjadi 4, loop akan melanjutkan ke iterasi berikutnya.

Pada iterasi selanjutnya, i menjadi 5, dan program akan mencetak angka 5, lalu menjadi 6 dan mencetak angka 6.

Akhirnya, loop berakhir ketika i menjadi 7.

Kegunaan break dan continue

break:

Digunakan untuk menghentikan eksekusi loop sepenuhnya. Ketika break dijalankan, kontrol langsung keluar dari loop, dan program melanjutkan eksekusi dari baris kode setelah loop.

Berguna ketika kita ingin menghentikan loop berdasarkan suatu kondisi tertentu.

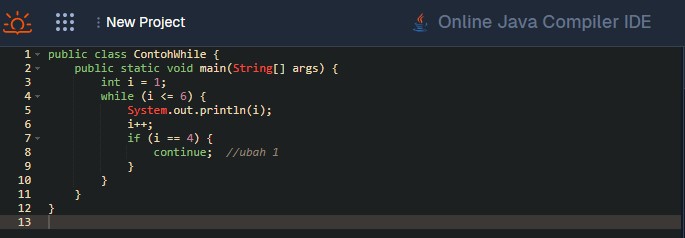
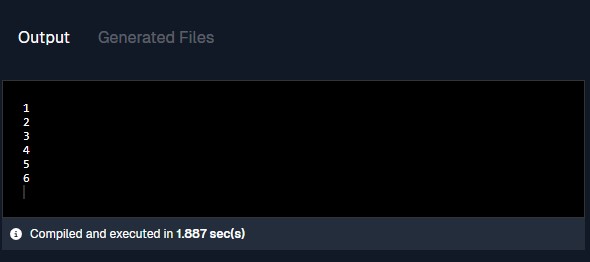
continue:

Digunakan untuk melewatkan sisa iterasi saat ini dan langsung melanjutkan ke iterasi berikutnya dari loop.

Berguna ketika kita ingin mengabaikan beberapa langkah dalam loop berdasarkan suatu kondisi tetapi masih ingin melanjutkan eksekusi loop.

**[No.1 ] Penyusunan Algoritma dan Kode Program**

1. Algoritma
2. Inisialisasi
3. Mulai Loop
4. Tampilkan Nilai
5. Increment
6. Cek Kondisi
7. Akhiri Loop
8. Program Selesai
9. Kode program dan luaran



1. Screenshot/ Capture potongan kode dan hasil luaran

Beri komentar pada kode yang di Screenshot

1. Analisa luaran yang dihasilkan   
   Program hanya mencetak angka 1, 2, dan 3 sebelum keluar dari loop ketika i mencapai 4.

Dengan menggunakan perintah break, program menghentikan eksekusi loop berdasarkan kondisi yang ditentukan.

Ini menunjukkan bagaimana kontrol alur dalam loop dapat memengaruhi luaran program, di mana tidak semua iterasi dicetak jika suatu kondisi dipenuhi..

**[No.1] Kesimpulan**

1. **Evaluasi**

Kelebihan:

Sederhana dan Mudah Dipahami: Struktur program jelas dan mudah diikuti, cocok untuk pemula.

Penggunaan break: Memperlihatkan cara menghentikan loop berdasarkan kondisi tertentu, berguna dalam banyak situasi.

Kekurangan:

Tidak Mencetak Semua Nilai: Program tidak mencetak angka 4, 5, atau 6. Jika tujuannya adalah mencetak semua angka dalam rentang tersebut, maka penggunaan break tidak sesuai.

Kurangnya Komentar: Meskipun kode sederhana, penjelasan lebih lanjut (komentar) mengenai fungsionalitas dapat membantu pemahaman, terutama untuk pemula.

|  |  |  |
| --- | --- | --- |
| **Nama & NPM** | **Topik:** | **Tanggal:** |
| **Ariansyah Putra Ramadhan**  **G1F024080** | **If dan switch** | **10 Oktober 2024** |

**[No. 1] Identifikasi Masalah:**

2.2. Buat perubahan nilai angka pada variabel di Contoh 5  
       //Ubah2 menjadi if (count % 5 == 0)  lalu running, periksa hasilnya  
       Analisa dampaknya perubahan terhadap luaran setelah running dan uraikan kegunaan % untuk angka yang berbeda         pada perintah tersebut!

**Contoh 5:** Salin dan tempel kode program berikut ke Eclipse.

public class WhileBersarang {  
    public static void main(String[] args) {  
        int count = 0;  //ubah1  
        while (count < 20) {  
        if (count % 3 == 0)  //ubah2  
        System.out.println(count);  
        count++;  
        }  
        }  
    }

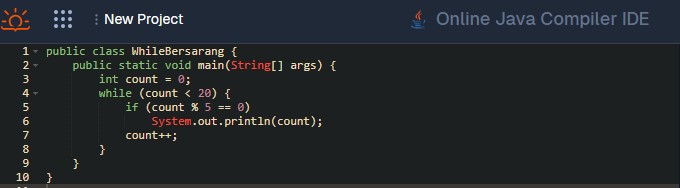
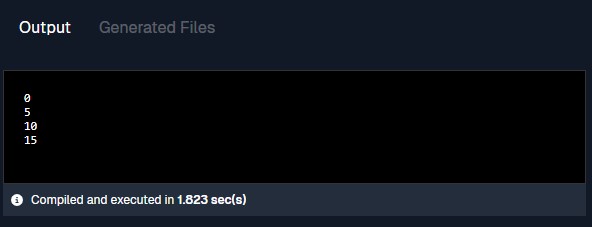
**[No.1] Analisis dan Argumentasi**

Setelah melakukan perubahan pada kode dengan mengubah if (count % 3 == 0) menjadi if (count % 5 == 0), program tersebut akan mencetak angka-angka dari 0 hingga 19 yang merupakan kelipatan dari 5.

Operator modulus (%) digunakan untuk mendapatkan sisa hasil bagi dari suatu angka. Dalam konteks ini, count % 5 akan menghasilkan 0 ketika count adalah kelipatan dari 5.

**[No.1 ] Penyusunan Algoritma dan Kode Program**

1. Algoritma
2. Inisialisasi
3. Loop
4. Selesai
5. Kode program dan luaran



1. Screenshot/ Capture potongan kode dan hasil luaran

Beri komentar pada kode yang di Screenshot

1. Analisa luaran yang dihasilkan   
   Program yang di atas akan mencetak angka dari 0 hingga 19 yang merupakan kelipatan 3.

**[No.1] Kesimpulan**

1. **Evaluasi**

Program WhileBersarang berfungsi dengan baik untuk tujuan yang ditetapkan, tetapi ada ruang untuk perbaikan dalam hal fleksibilitas dan penanganan input. Dengan beberapa perubahan, program dapat menjadi lebih umum dan lebih mudah digunakan.

|  |  |  |
| --- | --- | --- |
| **Nama & NPM** | **Topik:** | **Tanggal:** |
| **Ariansyah Putra Ramadhan**  **G1F024080** | **If dan switch** | **10 Oktober 2024** |

**[No. 1] Identifikasi Masalah:**

2.3. Buat perubahan nilai angka pada variabel di   
       //Ubah1 menjadi while (count < 0) {    lalu running, periksa hasilnya  
       Ubahlah baris kode while pada Contoh 5 menjadi do … while dengan persyaratan yang sama  while (count < 0).   Bandingkan hasil luaran antara menggunakan while dan do … while!

**Contoh 5:** Salin dan tempel kode program berikut ke Eclipse.

public class WhileBersarang {  
    public static void main(String[] args) {  
        int count = 0;  //ubah1  
        while (count < 20) {  
        if (count % 3 == 0)  //ubah2  
        System.out.println(count);  
        count++;  
        }  
        }  
    }

**[No.1] Analisis dan Argumentasi**

1. Mengubah Kode while dengan while (count < 0)

Jika kode dijalankan, tidak ada output yang dihasilkan. Ini karena kondisi count < 0 tidak terpenuhi (nilai awal count adalah 0).

2. Mengubah Kode menjadi do...while

Dengan menggunakan do...while, program akan tetap menjalankan blok kode setidaknya satu kali, meskipun kondisi count < 0 tidak terpenuhi. Karena nilai count adalah 0, outputnya juga tetap tidak ada, tetapi ada perbedaan dalam cara eksekusi.

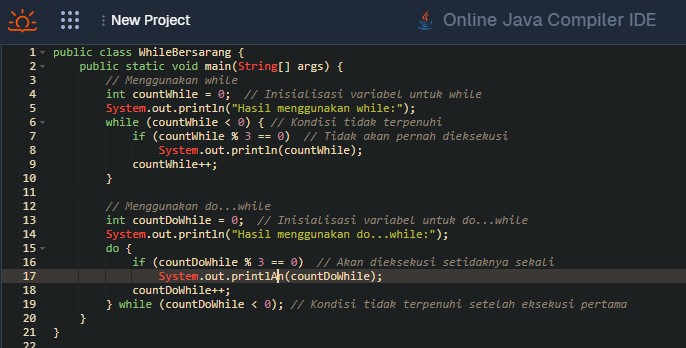
Perbandingan:

while: Memeriksa kondisi sebelum menjalankan blok kode. Jika kondisi tidak terpenuhi (seperti count < 0), tidak ada yang dijalankan.

do...while: Selalu menjalankan blok kode setidaknya satu kali, kemudian memeriksa kondisi. Dalam hal ini, meskipun tidak ada output, pendekatannya berbeda dalam cara eksekusi.

**[No.1 ] Penyusunan Algoritma dan Kode Program**

1. Algoritma
2. Mulai
3. Inisialisasi Variabel
4. Output
5. While Loop
6. Output
7. Do...While Loop
8. Selesai
9. Kode program dan luaran



1. Screenshot/ Capture potongan kode dan hasil luaran

Beri komentar pada kode yang di Screenshot

1. Analisa luaran yang dihasilkan   
   hanya angka 0 yang dicetak sebagai hasil dari loop do...while, sementara loop while tidak menghasilkan output sama sekali.

**[No.1] Kesimpulan**

**Evaluasi**

Kelebihan

Program ini secara jelas menunjukkan perbedaan antara while dan do...while. Ini merupakan contoh yang baik untuk pemahaman dasar tentang kedua jenis loop.

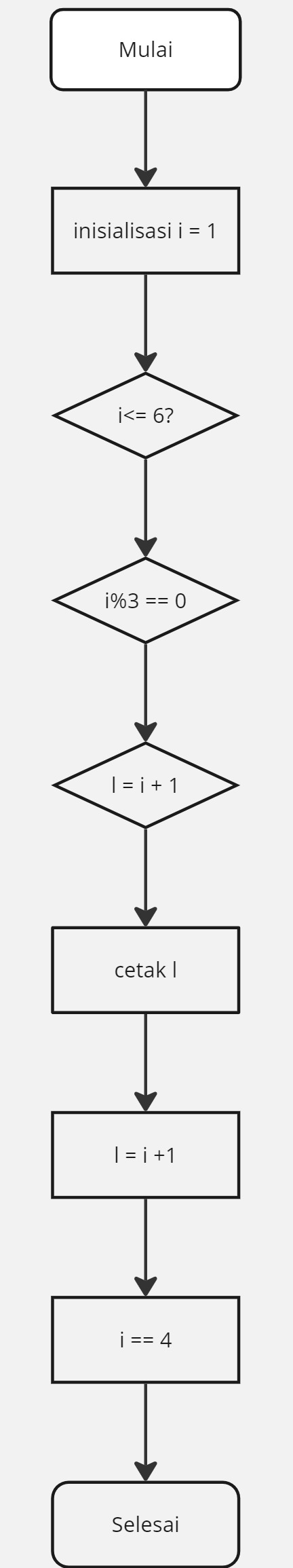
Kekurangan

Bagian while tidak berguna karena tidak menghasilkan output dan tidak melakukan operasi yang berarti. Ini mungkin membingungkan bagi pembaca yang tidak memahami bahwa kondisi tersebut tidak terpenuhi.

Program ini bisa ditingkatkan dengan menggunakan rentang yang lebih luas untuk menunjukkan fungsi looping yang lebih kompleks, seperti menghitung hingga nilai tertentu.

Nama kelas WhileBersarang mungkin sedikit menyesatkan karena tidak ada loop bersarang (nested loop) di dalam kode.

Gambar flowchart 2.1 :



Gambar Flowchart Latihan 2.3:

