

Unit 1 FOR (Latihan 1)

| Nama & NPM | Topik: | Tanggal: |
|----------------------------|---------------|----------------|
| Dini Ramadona G1F024050 | FOR dan WHILE | 5 Oktober 2024 |

[1.1] Evaluasi penyebab kesalahan dan perbaiki kode pada Contoh 1!

Rekomendasikan kata kunci yang tepat diletakkan pada baris kode yang kosong 1 dan 2 untuk dapat menghasilkan luaran berikut:

Luaran contoh 1:

0
2
4
6

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

```
public class ContohFor{  
    public static void main(String[] args) {  
        for (double y <= 15; y = 0; y++) {  
            if (y % 2 == 1) {           //kondisi 1  
                // baris kode kosong 1  
            } else if (y == 8) {       //kondisi 2  
                // baris kode kosong 2  
            } else  
                System.out.println(y + " ");  
        }  
    }  
}
```

Luaran:

```
Exception in thread "main" java.lang.Error: Unresolved compilation problems:  
    Syntax error on token "<=", = expected  
    Type mismatch: cannot convert from double to boolean  
    at ContohFor.main(ContohFor.java:5)  
    }  
}
```

Evaluasi penyebab kesalahan:

1. Tipe data `double` untuk `y`:
Kesalahan: Penggunaan `double` untuk variabel loop yang hanya memerlukan bilangan bulat.
Penyebab: Mungkin kesalahpahaman tentang kebutuhan presisi atau kurangnya pertimbangan tentang tipe data yang sesuai.
Perbaikan: Menggunakan `int` lebih tepat untuk menghitung iterasi bilangan bulat dan lebih efisien.
2. Sintaks `for` loop yang salah:
Kesalahan: `for (double y <= 15; y = 0; y++)` adalah sintaks yang tidak valid.
Penyebab: Kesalahpahaman tentang struktur `for` loop yang benar.
Perbaikan: Memperbaiki menjadi `for (int y = 0; y <= 15; y++)`, yang merupakan struktur yang benar (inisialisasi; kondisi; iterasi).
3. Logika penanganan angka ganjil:
Kesalahan: Tidak ada implementasi untuk melewati angka ganjil.
Penyebab: Kode kosong pada kondisi pertama tidak melakukan apa-apa.
Perbaikan: Menambahkan `continue` untuk melewati iterasi saat angka ganjil ditemui.
4. Logika penghentian loop:
Kesalahan: Tidak ada implementasi untuk menghentikan loop saat `y` mencapai 8.
Penyebab: Kode kosong pada kondisi kedua tidak melakukan apa-apa.

Perbaikan: Menambahkan `break` untuk menghentikan loop saat `y` mencapai 8.

5. Struktur kontrol yang tidak efisien:

Kesalahan: Penggunaan `else if` dan `else` yang tidak optimal.

Penyebab: Mungkin kurangnya pemahaman tentang cara mengorganisir logika kontrol yang efisien.

Perbaikan: Menyusun ulang kondisi untuk menghasilkan output yang diinginkan dengan lebih efisien.

6. Indentasi dan formatting:

Kesalahan: Indentasi dan formatting yang tidak konsisten dan sulit dibaca.

Penyebab: Mungkin kurangnya perhatian terhadap best practices dalam penulisan kode.

Perbaikan: Memperbaiki indentasi dan formatting untuk meningkatkan keterbacaan kode.

Penjelasan perbaikan:

1. Mengubah tipe data `y` dari `double` ke `int` karena kita hanya bekerja dengan bilangan bulat.
2. Memperbaiki sintaks for loop: `for (int y = 0; y <= 15; y++)`.
3. Menggunakan `continue` untuk melewati angka ganjil.
4. Menggunakan `break` untuk menghentikan loop saat `y` mencapai 8

Kode program FOR yang sudah diperbaiki:

```
1 package com.dini;
2
3 public class ContohFor {
4     public static void main(String[] args) {
5         for (int y = 0; y <= 15; y++) {
6             if (y % 2 == 1) {
7                 continue; // Skip angka ganjil
8             } else if (y == 8) {
9                 break; // Hentikan loop saat y mencapai 8
10            } else {
11                System.out.println(y + " ");
12            }
13        }
14    }
15 }
```

Kode program FOR luaran yang sudah diperbaiki:

```
<terminated> ContohFor [Java]
0
2
4
6
```

[1.2] Cermati contoh kode 2 pada kode //baris kode kosong.

Rekomendasikan kode yang tepat menggunakan `break` atau `continue` terhadap pertama atau kedua agar menghasilkan luaran berikut:

Luaran Contoh 2:

```
i = 1; j = 1
i = 1; j = 2
i = 2; j = 1
i = 2; j = 2
```

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) {
                // kode yang hilang
            } } } }
```

Untuk menghasilkan luaran yang diminta, kita perlu memodifikasi kode dengan menambahkan statement `break` yang tepat, seperti contoh dibawah ini:

Penjelasan perubahan:

- Kita menambahkan `break pertama;` di dalam blok `if (i == 2)`.
- `break pertama;` akan menghentikan loop luar yang diberi label "pertama" segera setelah `i` mencapai nilai 2.
- Ini akan menghasilkan output yang diminta, di mana loop berhenti setelah mencetak nilai untuk `i = 2` dan `j = 2`.

Penjelasan output:

- Loop pertama (`i = 1`) berjalan lengkap untuk `j = 1` dan `j = 2`.
- Loop kedua (`i = 2`) juga berjalan lengkap untuk `j = 1` dan `j = 2`.
- Setelah `i = 2` selesai, kondisi `if (i == 2)` terpenuhi, dan `break pertama;` dieksekusi.
- Ini menghentikan loop luar, mencegah iterasi untuk `i = 3` dan `i = 4`.

Alternatif lain:

Kita bisa juga menggunakan `continue` untuk mencapai hasil yang sama dalam situasi tertentu, tetapi dalam kasus ini, `break` adalah pilihan yang lebih tepat karena kita ingin menghentikan loop luar sepenuhnya setelah kondisi tertentu terpenuhi.

Penggunaan label (`pertama:` dan `kedua:`) dalam kode ini memungkinkan kita untuk menentukan secara spesifik loop mana yang ingin kita hentikan. Tanpa label, `break` hanya akan menghentikan loop dalam (loop `j`) yang terdekat.

Praktik ini menunjukkan bagaimana kontrol alur yang tepat dapat digunakan untuk mencapai perilaku program yang diinginkan dalam struktur loop bersarang.

Kode program ForBersarang yang sudah diperbaiki:

```
1 package com.dini;
2
3 public class ForBersarang {
4     public static void main(String[] args) {
5         pertama:
6         for (int i = 1; i < 5; i++) {
7             kedua:
8             for (int j = 1; j < 3; j++) {
9                 System.out.println("i = " + i + "; j = " + j);
10            }
11            if (i == 2) {
12                // Kode yang ditambahkan: menghentikan loop pertama setelah i = 2
13                break pertama;
14            }
15        }
16    }
17 }
18 }
```

Kode program luaran ForBersarang yang sudah diperbaiki:

```
<terminated> ForBersarang [Java Applicat
i = 1; j = 1
i = 1; j = 2
i = 2; j = 1
i = 2; j = 2
```

[No 1.3] Cermati kode contoh 3. Apabila ingin menghasilkan luaran berikut:

Luaran berbentuk piramida

Masukan Input: 7

```
*
***
*****
*****
*****
*****
*****
*****
```

Rekomendasikan 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
        }
    }
}
```

Luaran:

Masukan Input: 7
*

```
**
***
****
*****
*****
*****
```

Penjelasan perubahan:

1. Loop utama (for (int t = 1; t <= tinggi; t++)) sekarang dimulai dari 1 dan berjalan hingga tinggi piramida. Ini membalik urutan dari kode asli untuk membuat piramida tumbuh dari atas ke bawah.
2. Loop pertama di dalam (for (int s = 1; s <= tinggi - t; s++)) mencetak spasi di awal setiap baris. Jumlah spasi berkurang seiring bertambahnya tinggi baris.
3. Loop kedua di dalam (for (int b = 1; b <= 2 * t - 1; b++)) mencetak bintang. Jumlah bintang dihitung dengan rumus $2 * t - 1$, yang menghasilkan jumlah ganjil bintang yang meningkat untuk setiap baris.
4. Komentar ditambahkan untuk menjelaskan fungsi setiap bagian kode.

Hasil output dari kode yang dimodifikasi untuk input 7 akan menjadi:

Penjelasan output yang akan dihasilkan:

- Baris pertama memiliki 6 spasi dan 1 bintang.
- Setiap baris berikutnya memiliki 1 spasi kurang dan 2 bintang lebih dari baris sebelumnya.
- Baris terakhir (ke-7) memiliki 0 spasi dan 13 bintang.

Perubahan ini menghasilkan piramida simetris yang sesuai dengan permintaan. Piramida tumbuh dari atas ke bawah, dengan jumlah bintang yang meningkat secara ganjil (1, 3, 5, 7, ...) dan jumlah spasi yang menurun di setiap baris.

Kode ini mendemonstrasikan penggunaan nested loops untuk mengontrol posisi dan jumlah karakter yang dicetak, menghasilkan pola geometris yang kompleks dari input sederhana.

Kode program Piramida Bintang:

```
module-info... Tugas_Opera... TugasOperat... Javadoc Console Tugas_Kelom...
package com.dini;

import java.util.Scanner;

public class PiramidaBintang {
    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 = 1; t <= tinggi; t++) { // Loop untuk setiap baris
            // Mencetak spasi di awal baris
            for (int s = 1; s <= tinggi - t; s++) {
                System.out.print(" ");
            }

            // Mencetak bintang
            for (int b = 1; b <= 2 * t - 1; b++) {
                System.out.print("*");
            }

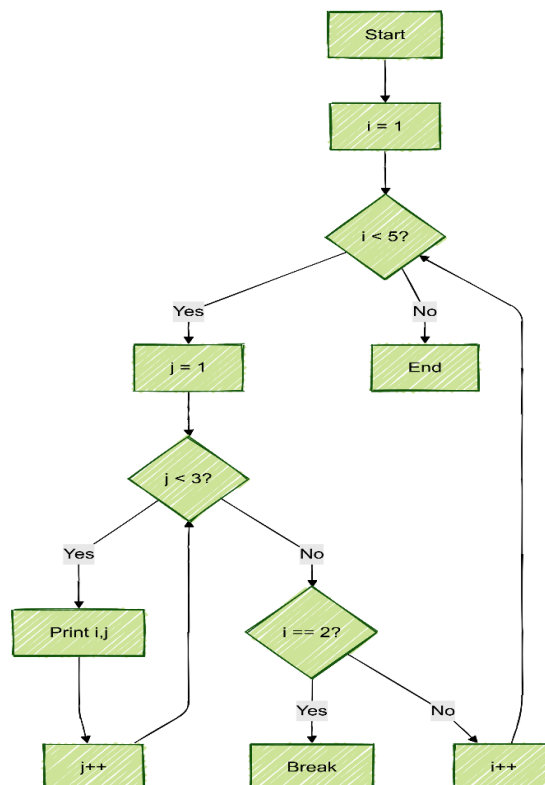
            System.out.println(); // Membuat Baris Baru
        }
    }
}
```

Kode program luaran Piramida Bintang:

```
<terminated> PiramidaBintang [Java Application] <
Masukan Input: 7
*
***
*****
*****
*****
*****
*****
```

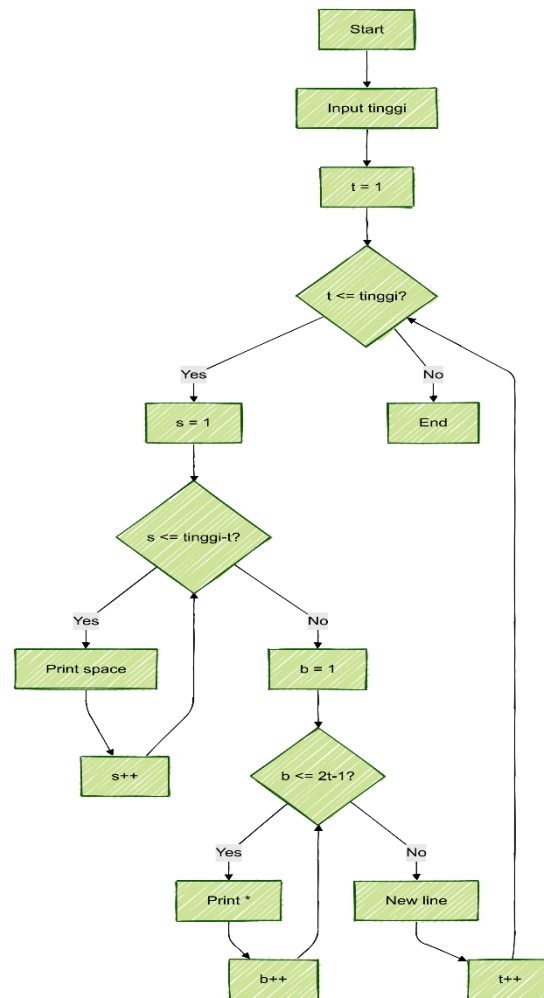
[No 1.4] Analisa diagram flowchart dari Latihan 1.2 dan 1.3!

Flowchart Latihan 1.2



- Start: Memulai program.
- Inisialisasi Loop Pertama: Mulai loop dengan variabel i.
- Inisialisasi Loop Kedua: Mulai loop dengan variabel j.
- Output: Cetak nilai i dan j.
- Cek Kondisi: Jika i == 2, gunakan break untuk keluar dari loop pertama.
- End: Akhiri program.

Flowchart Latihan 1.3



- Start: Memulai program.
- Input Pengguna: Menerima input tinggi piramida dari pengguna.
- Loop Tinggi Piramida: Untuk setiap tingkat, hitung jumlah spasi dan bintang.
- Output Bintang: Cetak bintang sesuai dengan tinggi piramida.
- End: Akhiri program.

Kesimpulan

1) Analisa

a) Kesimpulan Berdasarkan Permasalahan, Algoritma, dan Kode Program:

- Permasalahan:

Dalam latihan ini, terdapat beberapa contoh kode program yang menggunakan pernyataan for untuk melakukan pengulangan. Beberapa kode mengalami kesalahan sintaksis dan logika yang perlu diperbaiki agar dapat berfungsi dengan baik sesuai dengan luaran yang diharapkan.

- Algoritma:

Algoritma yang digunakan dalam setiap contoh program mengikuti struktur pengulangan for, di mana:

- Inisialisasi: Variabel kontrol diinisialisasi untuk memulai pengulangan.

- Kondisi: Ditetapkan untuk menentukan apakah pengulangan harus dilanjutkan atau dihentikan.
- Iterasi: Mengatur bagaimana variabel kontrol akan berubah pada setiap iterasi.

Contoh 1 menunjukkan penggunaan kondisi untuk memeriksa bilangan genap dan ganjil, sedangkan Contoh 2 menggunakan pengulangan bersarang untuk mencetak kombinasi nilai dari dua variabel. Contoh 3 menunjukkan bagaimana membangun piramida bintang berdasarkan input pengguna.

Kode Program:

Pada Contoh 1, kesalahan sintaksis diperbaiki dengan mengubah inisialisasi dan kondisi loop.

Pada Contoh 2, penggunaan break memungkinkan penghentian loop pertama saat kondisi tertentu terpenuhi.

Pada Contoh 3, penambahan logika untuk menghitung jumlah spasi dan bintang menghasilkan bentuk piramida yang diinginkan.

b) Dasar Alasan Pengambilan Keputusan:

Keputusan untuk memperbaiki kesalahan dalam kode program didasarkan pada pemahaman tentang struktur dasar pernyataan for dan pentingnya sintaksis yang benar dalam pemrograman. Kesalahan seperti penggunaan operator yang salah atau kondisi yang tidak tepat dapat menyebabkan program tidak berjalan sebagaimana mestinya. Pemilihan algoritma untuk menyelesaikan masalah didasarkan pada kebutuhan untuk mencapai hasil yang spesifik (misalnya, mencetak bilangan genap, mencetak kombinasi nilai, atau membangun piramida). Dengan menggunakan pernyataan for, kita dapat mengontrol jumlah iterasi dengan tepat sesuai dengan kebutuhan. Keputusan untuk menggunakan break atau continue dalam loop bersarang didasarkan pada kebutuhan untuk mengontrol alur program secara efisien. Ini membantu dalam mencapai hasil yang diinginkan tanpa menjalankan iterasi yang tidak perlu.

Refleksi

Pada minggu ini saya telah mempelajari konsep loop dalam Java melalui latihan-latihan ini memberikan pengalaman yang komprehensif dan praktis. Kita telah mengeksplorasi berbagai jenis loop (for, while, do-while) dan bagaimana mereka dapat diaplikasikan dalam situasi yang berbeda. Latihan membuat piramida bintang dengan for loop bersarang memberikan wawasan mendalam tentang bagaimana mengontrol alur program dengan presisi. Sementara itu, latihan menghitung rata-rata nilai menggunakan while loop menunjukkan fleksibilitas loop dalam menangani input yang jumlahnya tidak diketahui sebelumnya. Adapun tantangan yang saya hadapi dalam pembelajaran for while sebagai berikut:

- Merancang algoritma untuk pola piramida membutuhkan pemikiran spatial yang menantang.
- Menentukan kondisi yang tepat untuk break dan continue memerlukan pemahaman mendalam tentang alur program.
- Memahami perbedaan sintaks antara berbagai jenis loop memerlukan latihan dan pembiasaan.
- Penggunaan variabel kontrol loop (seperti i, j) dalam nested loops dapat membingungkan pada awalnya.
- Menemukan kesalahan dalam loop, terutama infinite loop, dapat menjadi tantangan tersendiri.
- Memahami alur eksekusi dalam nested loops memerlukan pemikiran yang cermat.

Unit 2 WHILE (Latihan 2)

| Nama & NPM | Topik: | Tanggal: |
|----------------------------|---------------|----------------|
| Dini Ramadona G1F024050 | FOR dan WHILE | 5 Oktober 2024 |

[2.1] Ubahlah baris kode pada Contoh 4

//Ubah1 menjadi `if(i % 3 == 0){ ◇ running, periksa hasilnya`

//Ubah2 menjadi `continue; ◇ running, periksa hasilnya`

Evaluasi perbandingan luaran sebelum dan setelah diubah! Simpulkan maksud dari perubahan tersebut!

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  
            }  
        }  
    }  
}
```

Luaran:

1
2
3

Evaluasi perbandingan:

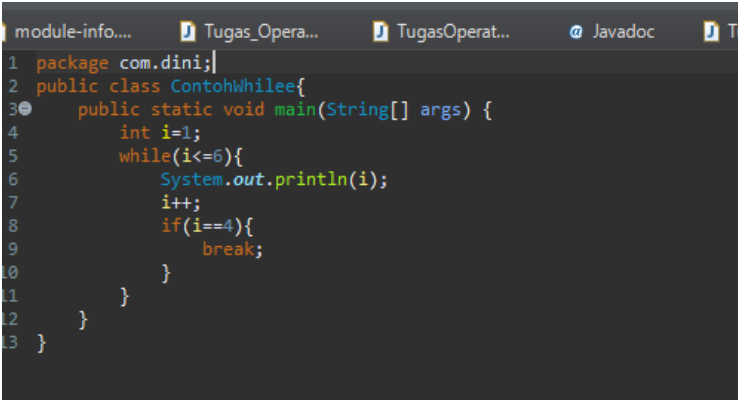
- Versi Awal: Mencetak 1, 2, 3 lalu berhenti karena break saat `i==4`.
- Versi Ubah1: Mencetak 1, 2, 4, 5 (melewati kelipatan 3).
- Versi Ubah2: Sama dengan Ubah1 karena continue sudah digunakan.

Simpulan:

- break menghentikan loop sepenuhnya.
- continue melewati iterasi saat ini dan melanjutkan ke iterasi berikutnya.

Beriku adalah kode program while sebelumnya dan yang sudah diubah 1 dan diubah yg 2:

Kode program WHILE sebelum diubah:



```
1 package com.dini;  
2 public class ContohWhilee{  
3     public static void main(String[] args) {  
4         int i=1;  
5         while(i<=6){  
6             System.out.println(i);  
7             i++;  
8             if(i==4){  
9                 break;  
10            }  
11        }  
12    }  
13 }
```

Kode program luaran WHILE sebelum diubah:

```
<terminated> ContohWhilee [Java Applica  
1  
2  
3
```

Kode program WHILE diubah 1:

```
module-info.... Tugas_Opera... TugasOperat... Javadoc Console  
1 package com.dini;  
2 public class ContohWhilee {  
3     public static void main(String[] args) {  
4         int i=1;  
5         while(i<=6){  
6             if(i % 3 == 0){  
7                 i++;  
8                 continue;  
9             }  
10            System.out.println(i);  
11            i++;  
12        }  
13    }  
14 }
```

Kode program luaran WHILE diubah 1:

```
<terminated> ContohWhilee [Java App  
1  
2  
4  
5
```

Kode program WHILE diubah 2:

```
module-info.... Tugas_Opera... TugasOperat... Javadoc  
1 package com.dini;  
2 public class ContohWhilee{  
3     public static void main(String[] args) {  
4         int i=1;  
5         while(i<=6){  
6             if(i % 3 == 0){  
7                 i++;  
8                 continue;  
9             }  
10            System.out.println(i);  
11            i++;  
12        }  
13    }  
14 }
```

Kode program luaran WHILE diubah 2:

```
<terminated> ContohWhilee [  
1  
2  
4  
5
```

[No 2.2] Cermati Contoh 5. Periksa luaran, bila ketika di eksekusi, jumlah yang diulang = 0!
Evaluasi luaran, bila kode diubah menjadi do ... while dengan masukan sama jumlah yang diulang = 0.
Simpulkan perbedaan while dan do ... while!

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

```
import java.util.Scanner;

public class ForBersarang {
    public static void main(String[] args) {
        Scanner dataKata = new Scanner(System.in);
        System.out.print("Masukkan Kata yang ingin diulang : ");
        String kata = dataKata.nextLine();

        Scanner dataJumlah = new Scanner(System.in);
        System.out.print("Masukkan Jumlah ingin diulang : ");
        int jumlah = dataJumlah.nextInt();

        int i = 0; //Inisialisasi batas dasar
        while(i < jumlah){
            System.out.println(kata);
            i++; //Faktor pengulang Increment
        }
    }
}
```

Luaran Contoh 5:

```
Masukkan Kata yang ingin diulang : Fakultas Teknik
Masukkan Jumlah ingin diulang : 5
Fakultas Teknik
Fakultas Teknik
Fakultas Teknik
Fakultas Teknik
Fakultas Teknik
```

Evaluasi Luaran Ketika Jumlah yang Diulang = 0:

- Dengan While: Jika pengguna memasukkan 0, tidak ada output yang dihasilkan.
- Dengan do...while: Meskipun pengguna memasukkan 0, kode dalam blok do tetap dieksekusi setidaknya sekali. Namun, dalam hal ini, tidak ada output karena kondisi di akhir loop tidak terpenuhi.

Kesimpulan Perbedaan:

- While Loop: Mengecek kondisi sebelum menjalankan blok kode. Jika kondisi salah sejak awal (misalnya, jika jumlah adalah 0), blok kode tidak akan pernah dijalankan.
- do...while Loop: Menjalankan blok kode setidaknya sekali sebelum memeriksa kondisi. Ini berguna ketika kita ingin memastikan bahwa setidaknya satu iterasi dilakukan..

Beriku adalah kode program yang dirubah menjadi do...while:

Kode program yang dirubah menjadi do...while:

```

package com.dini;
import java.util.Scanner;

public class DoWhileExample {
    public static void main(String[] args) {
        Scanner dataKata = new Scanner(System.in);
        System.out.print("Masukkan Kata yang ingin diulang : ");
        String kata = dataKata.nextLine();

        System.out.print("Masukkan Jumlah ingin diulang : ");
        int jumlah = dataKata.nextInt();

        int i = 0; // Inisialisasi batas dasar
        do {
            System.out.println(kata);
            i++; // Faktor pengulangan Increment
        } while(i < jumlah);
    }
}

```

Kode program luaran yang dirubah menjadi do...while:

```

<terminated> DoWhileExample [Java Application] C:\Users\Dini Ramadana
Masukkan Kata yang ingin diulang : 0
Masukkan Jumlah ingin diulang : 0
0

```

[No 2.3] Bila diketahui pernyataan pseudocode berikut:

- [1] inisiasi idPelajaran
- [2] inisiasi nilai pelajaran
- [3] inisiasi nilai rata-rata
- [4] Minta pengguna untuk menuliskan jumlah pelajaran
- [5] Ketika idPelajaran lebih kecil dari jumlah pelajaran
- [6] Minta pengguna untuk menuliskan nilai pelajaran
- [7] Hitung nilai rata-rata = (nilai pelajaran + nilai rata-rata) / 2
- [8] Tambah satu ke idPelajaran
- [9] Tampilkan nilai rata-rata

Rekomendasikan kode untuk menyelesaikan Pseudocode tersebut!

Berikut adalah rekomendasi kode Java berdasarkan pseudocode yang diberikan:

```

package com.dini;
import java.util.Scanner;

public class NilaiRataRata {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        int idPelajaran = 1;
        double nilaiPelajaran = 0;
        double nilaiRataRata = 0;

        System.out.print("Masukkan jumlah pelajaran: ");
        int jumlahPelajaran = input.nextInt();

        while (idPelajaran <= jumlahPelajaran) {
            System.out.print("Masukkan nilai pelajaran " + idPelajaran + ": ");
            nilaiPelajaran = input.nextDouble();

            nilaiRataRata = (nilaiPelajaran + nilaiRataRata) / 2;

            idPelajaran++;
        }
    }
}

```

```
System.out.println("Nilai rata-rata: " + nilaiRataRata);
```

Kode program Pseudocode :

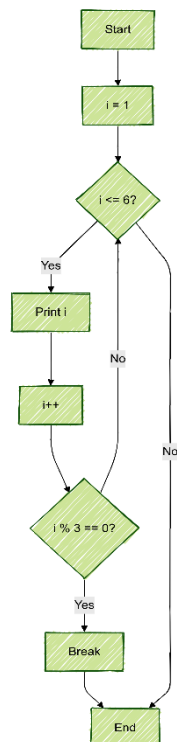
```
1 package com.dini;
2 import java.util.Scanner;
3
4 public class NilaiRataRata {
5     public static void main(String[] args) {
6         Scanner input = new Scanner(System.in);
7
8         int idPelajaran = 1;
9         double nilaiPelajaran = 0;
10        double nilaiRataRata = 0;
11
12        System.out.print("Masukkan jumlah pelajaran: ");
13        int jumlahPelajaran = input.nextInt();
14
15        while (idPelajaran <= jumlahPelajaran) {
16            System.out.print("Masukkan nilai pelajaran " + idPelajaran + ": ");
17            nilaiPelajaran = input.nextDouble();
18
19            nilaiRataRata = (nilaiPelajaran + nilaiRataRata) / 2;
20
21            idPelajaran++;
22        }
23
24        System.out.println("Nilai rata-rata: " + nilaiRataRata);
25    }
26 }
```

Kode program luaran Pseudocode :

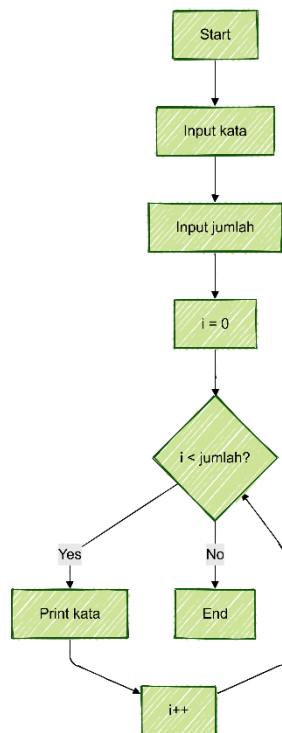
```
<terminated> NilaiRataRata [Java Application] C:\Users\Dini Kamadona\AppData\Local\Temp\plugins\
Masukkan jumlah pelajaran: 1
Masukkan nilai pelajaran 1: 100
Nilai rata-rata: 50.0
```

[No 2.4] Rancang diagram flowchart dari Latihan 2.1, Latihan 2.2, dan Latihan 2.3!

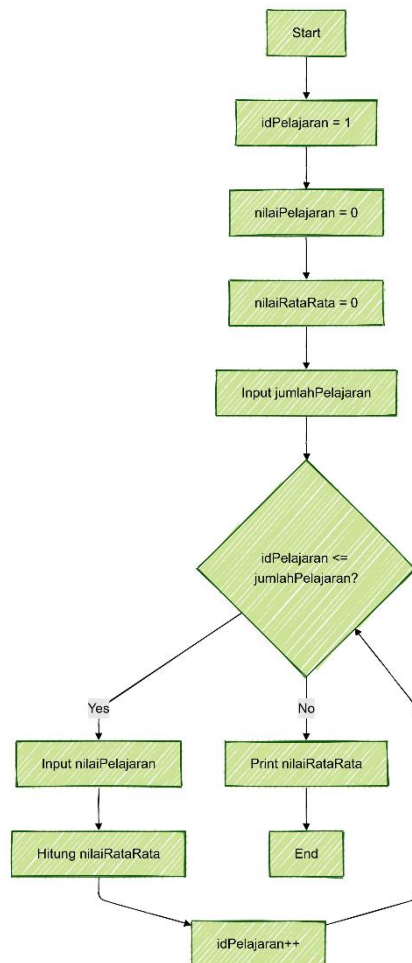
Flowchart 2.1



Flowchart 2.2



Flowchart 2.2



Kesimpulan

2) Evaluasi

a) Konsekuensi/Konsekuensi dari Kode Program yang Dibuat:

- Program Kontrol Pengulangan while:

Konseksi Utama: Penggunaan pernyataan while memungkinkan program untuk melakukan proses yang berulang-ulang selama kondisi tertentu masih benar.

Contoh 4:

Kondisi: while($i \leq 6$)

Aksi: Cetak nilai i dan increment i .

Break Statement: Digunakan untuk menghentikan loop ketika i mencapai nilai tertentu (contohnya, $i == 4$).

Dampak: Program akan mencetak angka dari 1 hingga 3 saja karena pernyataan break menghentikan loop sebelum mencapai angka 4.

Contoh 5:

Kondisi: while($i < \text{jumlah}$)

Aksi: Cetak kata yang diinputkan dan increment counter.

Dampak: Program akan mencetak kata yang diinputkan sebanyak jumlah yang diinputkan oleh user.

b) Evaluasi Input Program, Proses Perhitungan, dan Luaran yang Dihasilkan:

Contoh 4:

Input: Angka integer dari 1 hingga 6.

Proses Perhitungan:

Cetak nilai i .

Increment nilai i .

Jalankan pernyataan if($i == 4$) dan jalankan statement break jika kondisi terpenuhi.

Luaran:

Angka dari 1 hingga 3 saja karena pernyataan break menghentikan loop sebelum mencapai angka 4.

Contoh 5:

Input: Kata string dan jumlah ulangan.

Proses Perhitungan:

Masukkan kata yang diinputkan oleh user.

Masukkan jumlah ulangan yang diinputkan oleh user.

Jalankan pernyataan while($i < \text{jumlah}$) dan cetak kata yang diinputkan sebanyak jumlah yang diinputkan oleh user.

Luaran:

Kata yang diinputkan sebanyak jumlah yang diinputkan oleh user.

Kesimpulan Evaluasi

Dalam evaluasi kode program yang dibuat menggunakan pernyataan while, kita dapat melihat bahwa penggunaan kondisional dan control-flow statements seperti break sangat penting dalam mengatur alur program. Pada Contoh 4, penggunaan break menghentikan loop sebelum mencapai angka 4, sedangkan pada Contoh 5, penggunaan while memungkinkan program untuk mencetak kata yang diinputkan sebanyak jumlah yang diinputkan oleh user.

Refleksi

Pada minggu ini saya telah mempelajari konsep loop dalam Java melalui latihan-latihan ini memberikan pengalaman yang komprehensif dan praktis. Kita telah mengeksplorasi berbagai jenis loop (for, while, do-while) dan bagaimana mereka dapat diaplikasikan dalam situasi yang berbeda. Latihan membuat piramida bintang dengan for loop bersarang memberikan wawasan mendalam tentang bagaimana mengontrol alur program dengan presisi. Sementara itu, latihan menghitung rata-rata nilai menggunakan while loop menunjukkan fleksibilitas loop dalam menangani input yang jumlahnya tidak diketahui sebelumnya. Adapun tantangan yang saya hadapi dalam pembelajaran for while sebagai berikut:

- Merancang algoritma untuk pola piramida membutuhkan pemikiran spatial yang menantang.
- Menentukan kondisi yang tepat untuk break dan continue memerlukan pemahaman mendalam tentang alur program.
- Memahami perbedaan sintaks antara berbagai jenis loop memerlukan latihan dan pembiasaan.
- Penggunaan variabel kontrol loop (seperti i, j) dalam nested loops dapat membingungkan pada awalnya.
- Menemukan kesalahan dalam loop, terutama infinite loop, dapat menjadi tantangan tersendiri.
- Memahami alur eksekusi dalam nested loops memerlukan pemikiran yang cermat.