Nama & NPM	Topik:	Tanggal:
Zaira ayu wandira **G1F024055**	**FOR dan WHILE**	**28 septemper 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!

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)!

1.3. Cermati kode contoh 3. Apabila ingin menghasilkan luaran berikut:

Luaran:
Masukan Input: 7
****
******
*****
****
***
**
*

Susunlah analisa kode untuk menghasilkan luaran tersebut!

1.4. Analisa diagram flowchart dari Latihan 1.2 dan 1.3!

[No.l] Analisis dan Argumentasi

1.1. Analisis Penggunaan continue dan break di Contoh 1

Kata kunci continue digunakan untuk melewatkan langkah tertentu dalam perulangan tanpa menjalankan pernyataan setelahnya. Dalam contoh 1, jika nilai y adalah ganjil (sesuai kondisi if (y % 2 == 1)), maka continue akan dilewati, dan perulangan langsung dilanjutkan ke langkah berikutnya tanpa mengeksekusi baris System.out.println(y + " ").

Kata kunci break digunakan untuk menghentikan seluruh perulangan. Pada contoh 1, ketika y == 8, pernyataan break menandakan bahwa perulangan akan segera berhenti, dan program keluar dari loop meskipun syarat perulangan masih memungkinkan langkah-langkah lebih lanjut.

Perubahan Kode:

- Ubah 1: Mengubah loop menjadi for (int y = 0; y <= 15; y++)
  - Hasil: Loop sekarang berjalan dari 0 sampai 15, tetapi masih berhenti saat y == 8, jadi hasilnya tetap 0, 2, 4, 6.
- Ubah 2: Mengubah kondisi menjadi if (y % 2 == 0)
  - Hasil: Karena continue aktif saat y genap, hanya angka ganjil (1, 3, 5, 7) yang dicetak hingga program berhenti di y == 8.
- Ubah 3: Mengubah kondisi menjadi else if (y == 9)
  - Hasil: Loop berhenti saat y == 9, sehingga hasil yang dicetak adalah 0, 2, 4, 6, 8, lalu berhenti.

1.2. Perubahan Kode di Contoh 2

a. continue pertama;

- Hasil: Ketika i == 2, program akan melewati semua bagian untuk i == 2, dan langsung lanjut ke i == 3. Jadi, tidak ada output untuk i == 2.

b. break pertama;

- Hasil: Saat i == 2, program langsung keluar dari semua loop, jadi hanya hasil i == 1 yang dicetak Perulangan langsung berhenti.

c. continue kedua;

- Hasil: Ketika i == 2, hanya bagian j == 2 yang dilewati, tapi j == 1 tetap dicetak. Jadi sebagian dari hasil i == 2 masih ada.

1.3 Analisis 1.3

Analisis ini menunjukkan bagaimana penggunaan loop bersarang dapat digunakan untuk menghasilkan pola yang diinginkan berdasarkan input dari pengguna. Struktur logika dalam loop memastikan bahwa jumlah karakter yang dicetak sesuai dengan tinggi yang dimasukkan.

1.4 Analisis Flowchart dari Latihan 1.2 dan 1.3

- Latihan 1.2 Flowchart:
  - Menunjukkan alur eksekusi loop ganda dengan percabangan pada setiap langkah.
  - Menggambarkan kapan continue dan break digunakan untuk melanjutkan atau menghentikan proses.
- Latihan 1.3 Flowchart:
  - Menggambarkan bagaimana input tinggi digunakan untuk menentukan jumlah baris.
  - Mengindikasikan bagaimana loop dalam menghasilkan jumlah asteris yang berkurang setiap baris, sampai mencapai satu asteris di baris terakhir.

[No.1] Kode Program dan flowchart

1.1 perubahan nilai angka pada variabel

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

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

```
java -cp /tmp/d7d8oiwI5r/ContohFor
0
2
4
6

=== Code Execution Successful ===
```

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

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

```
java -cp /tmp/8Jy1OJZ3FH/ContohFor
1
3
5
7
9

=== Code Execution Successful ===
```

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

```
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 == 9)            //ubah 3
            break;              //baris 2
            else
            System.out.println(y + " ");
    }    }    }
```

```
java -cp /tmp/Zbdu9DGTkt/ContohFor
0
2
4
6
8
10

=== Code Execution Successful ===
```

1.2.Buat perubahan kode pada Contoh 2

- Ubah 1 menjadi continue pertama;

```
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)
                    continue pertama; // Ubah 1 (continue pertama)
            }
        }
    }
}
```

```
java -cp /tmp/BP1T2OEQ36/ForBersarang
i = 1; j = 1
i = 1; j = 2
i = 2; j = 1
i = 3; j = 1
i = 3; j = 2
i = 4; j = 1
i = 4; j = 2

=== Code Execution Successful ===
```

- Ubah 1 menjadi break pertama

```
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 pertama; // Ubah 1 (break pertama)
            }
        }
    }
}
```

```
java -cp /tmp/aI8XGxRKjt/ForBersarang
i = 1; j = 1
i = 1; j = 2
i = 2; j = 1

=== Code Execution Successful ===
```

- Ubah 1 menjadi continue kedua;

```
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)
                    continue kedua; // Ubah 1 (continue kedua)
            }
        }
    }
}
```

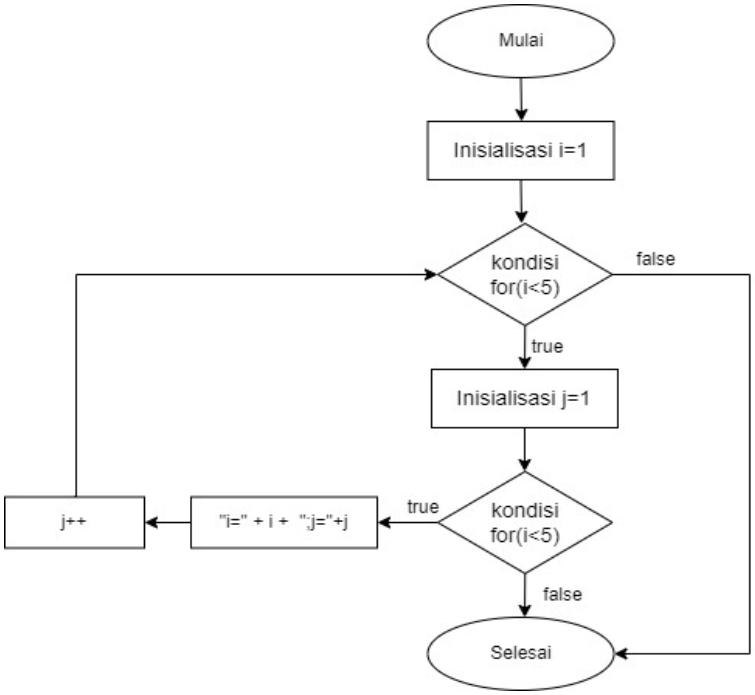
```
java -cp /tmp/YrrjZvEU2Z/ForBersarang
i = 1; j = 1
i = 1; j = 2
i = 2; j = 1
i = 2; j = 2
i = 3; j = 1
i = 3; j = 2
i = 4; j = 1
i = 4; j = 2

=== Code Execution Successful ===
```

1.3. Mengubah Kode pada Contoh 3

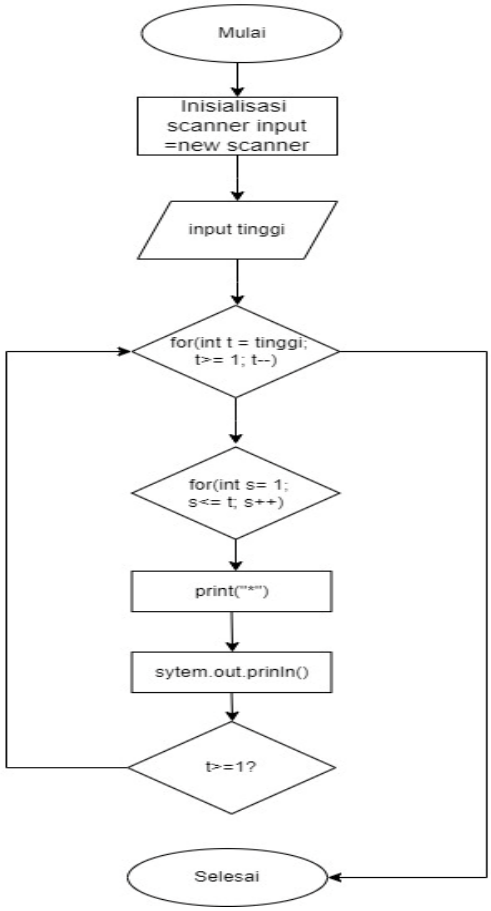
```
import java.util.Scanner;

public class ForBersarang {
    public static void main(String[] args){
        Scanner input = new Scanner(System.in);
        System.out.print("Masukan Input: ");
        int tinggi = input.nextInt();  // Mendapatkan input dari pengguna

        for(int t = tinggi; t >= 1; t--){  // Loop dari tinggi ke 1
            for(int s = 1; s <= t; s++){  // Cetak bintang sebanyak t
                System.out.print("*");
            }
            System.out.println();  // Pindah ke baris baru
        }
    }
}
```

```
java -cp /tmp/GARUy6jLMO/ForBersarang
Masukan Input: 7
*******
******
*****
****
***
**
*

=== Code Execution Successful ===
```

1.4 flowchart dari latihan 1.2



flowchart dari latihan 1.3



[No.1] Kesimpulan

Analisis penggunaan kata kunci continue dan break pada Contoh 1 menunjukkan bahwa continue digunakan untuk melewati langkah dalam perulangan, sehingga angka ganjil tidak dicetak, sementara break menghentikan perulangan saat mencapai angka tertentu. Perubahan pada nilai perulangan dan kondisi menghasilkan output yang berbeda. Di Contoh 2, penerapan continue pertama break pertama, dan continue kedua memberikan hasil yang bervariasi, di mana continue melewatkan langkah, dan break menghentikan perulangan sepenuhnya. Untuk Contoh 3, modifikasi diperlukan untuk mencetak piramida terbalik dengan mengurangi jumlah simbol'* pada setiap baris. Flowchart dari latihan-latihan ini membantu memperjelas alur logika dan perbedaan penggunaan continue dan break.

[No. 2] 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!
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!
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!
2.4. Analisa diagram flowchart dari Latihan 2.1, Contoh 5, dan Latihan 2.3!

[No.2] Analisis dan Argumentasi

2.1 Analisis Dampak Perubahan terhadap Luaran :

- **Sebelum perubahan (break)**:
  Ketika program menemukan angka 4, perulangan **berhenti total**. Jadi, hanya angka 1, 2, dan 3 yang tercetak. Pernyataan break membuat program keluar dari perulangan lebih awal, meskipun seharusnya masih ada angka lain yang bisa dicetak (seperti 4, 5, dan 6).
- **Setelah perubahan (continue)**:
  Ketika program mencapai angka 4, program **melewatkan** pencetakan angka tersebut, tetapi **tidak menghentikan perulangan**. Program melanjutkan prosesnya, sehingga angka 5 dan 6 tetap tercetak.

kegunaan break dan continue:

- break digunakan untuk menghentikan perulangan secara paksa sebelum kondisi perulangan terpenuhi.

- continue digunakan untuk melewati sisa pernyataan dalam satu proses dan melanjutkan ke proses berikutnya.

2.2 Analisis Dampak Perubahan terhadap Luaran :

- Sebelum diubah: Perulangan mencetak nilai count yang merupakan kelipatan 3, seperti 0, 3, 6, 9, 12, 15, 18
- Setelah diubah: Sekarang perulangan akan mencetak nilai count yang merupakan kelipatan 5, seperti 0, 5, 10, 15

kegunaan %:

Operator % (modulus) digunakan untuk mendapatkan sisa dari pembagian dua angka. Jadi, pernyataan if (count % 5 == 0) akan menghasilkan true jika count adalah kelipatan 5, karena sisa pembagian count dengan 5 adalah 0. Mengubah nilai modulus dari 3 ke 5 mempengaruhi kelipatan angka yang tercetak

2.3.Analisis perbandingan hasil luaran antara menggunakan while dan do … while

- while: Mengecek kondisi terlebih dahulu. Jika kondisi salah dari awal, blok kode tidak akan dieksekusi sama sekali. Tidak ada luaran karena kondisi awal count < 0 langsung bernilai false.
- do-while: Menjalankan blok kode setidaknya sekali sebelum memeriksa kondisi. Jika kondisi salah setelah eksekusi pertama, loop akan berhenti. Program mencetak luaran 0 karena perintah dalam do-while selalu dieksekusi setidaknya sekali sebelum memeriksa kondisi.

2.4. Analisa diagram flowchart dari Latihan 2.1, Contoh 5, dan Latihan 2.3!

Analisis Flowchart**:**

1. Flowchart Latihan 2.1 (Perubahan break ke continue):
   - Menggambarkan loop yang terus menerus hingga i mencapai 6, dengan cabang yang menunjukkan penggunaan continue saat i sama dengan 4.
2. Flowchart Contoh 5 (Perubahan modulus):
   - Mengilustrasikan loop yang terus menerus hingga count mencapai 20, dengan cabang untuk memeriksa apakah count merupakan kelipatan dari 5 dan mencetak hasilnya.
3. Flowchart Latihan 2.3 (Perubahan ke do...while):
   - Menunjukkan eksekusi blok kode sekali terlepas dari kondisi, diikuti oleh pemeriksaan kondisi yang tidak memenuhi.

[No.2] Kode Program dan flowchart

2.1 ubah break menjadi continue

```
1  public class ContohWhile{
2  public static void main(String[] args) {
3      int i=1;
4          while(i<=6){
5              System.out.println(i);
6              i++;
7              if(i==4){
8                  continue;        //ubah1
9              }}}}
```

```
java -cp /tmp/4MqQgHqmQc/ContohWhile
1
2
3
4
5
6

=== Code Execution Successful ===
```

2.2 ubah if (count % 3 == 0) menjadi if (count % 5 == 0)

```
1  public class WhileBersarang { //deklarasi kelas
2      public static void main(String[] args) { //method utama
3          int count = 0;  // Inisialisasi  count
4          while (count < 20) {  // Loop akan berjalan selama count kurang dari 20
5              if (count % 5 == 0)  // Jika kelipatan 5
6                  System.out.println(count);  // Mencetak nilai count
7              count++;  // Meningkatkan nilai count sebesar 1
8          }
9      }
10 }
```

```
java -cp /tmp/RZ71oIXJAN/WhileBersarang
0
5
10
15

=== Code Execution Successful ===
```

2.3 Buat perubahan nilai angka pada variable

- Dengan while

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

```
java -cp /tmp/Q3oOXLzIYh/WhileBersarang

=== Code Execution Successful ===
```

- Dengan do-while

```java
public class DoWhileBersarang { //deklarasi kelas
    public static void main(String[] args) { //method main
        int count = 0;
        // Perulangan do...while dengan kondisi count < 0
        do {
            if (count % 3 == 0) {
                System.out.println(count);
            }
            count++;
        } while (count < 0);
    }
}
```

```
java -cp /tmp/grOHn9X8wQ/DoWhileBersarang
0

=== Code Execution Successful ===
```

2.4. diagram flowchart dari Latihan 2.1

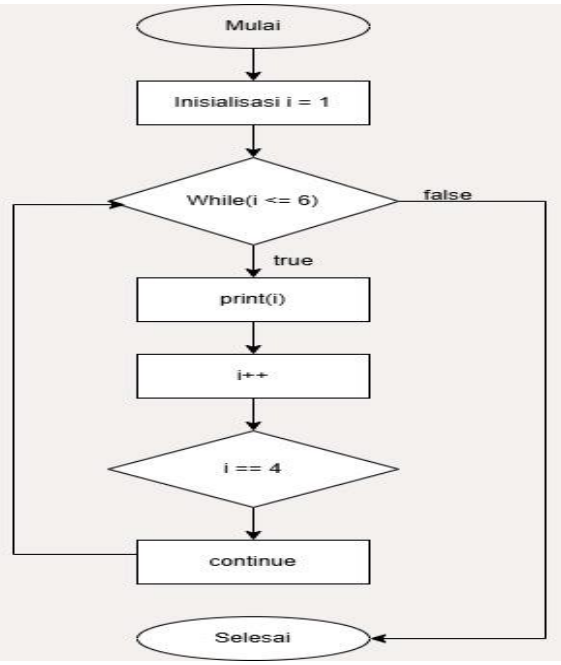
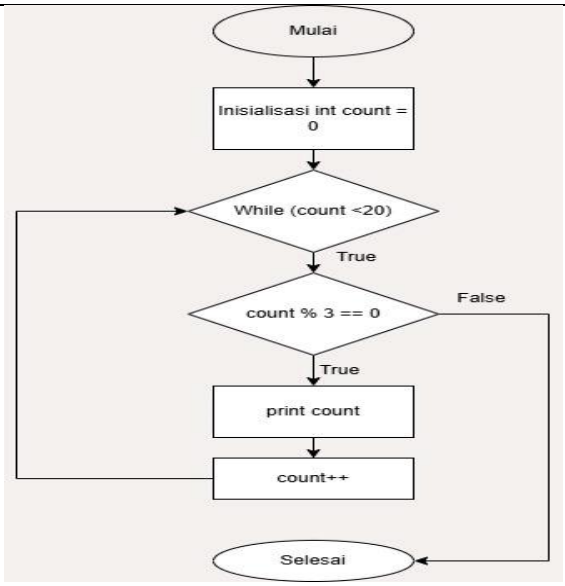
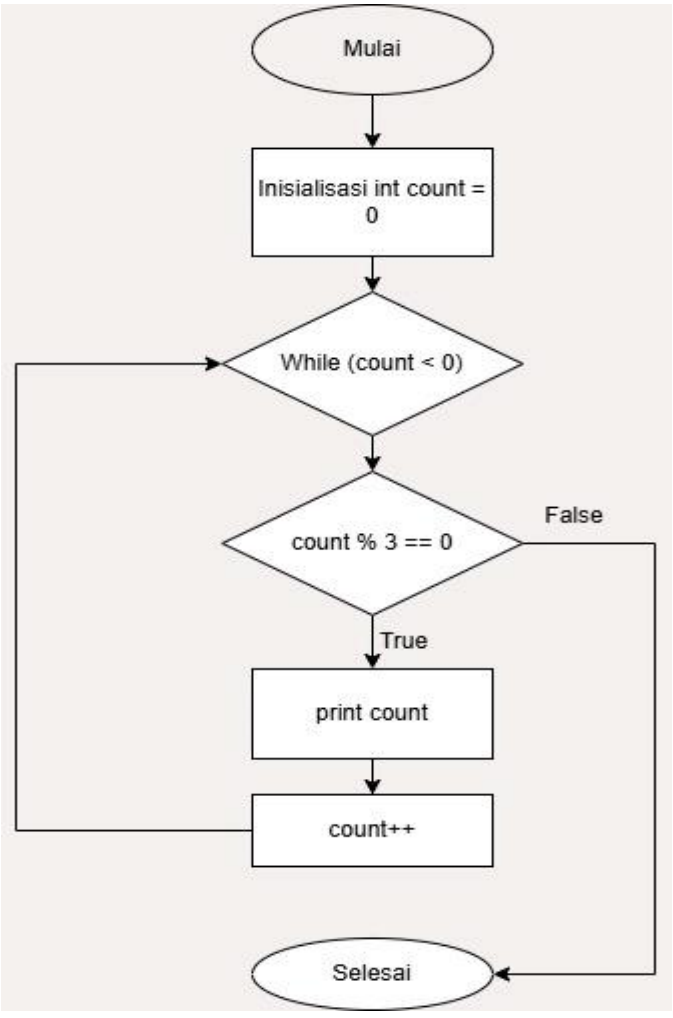


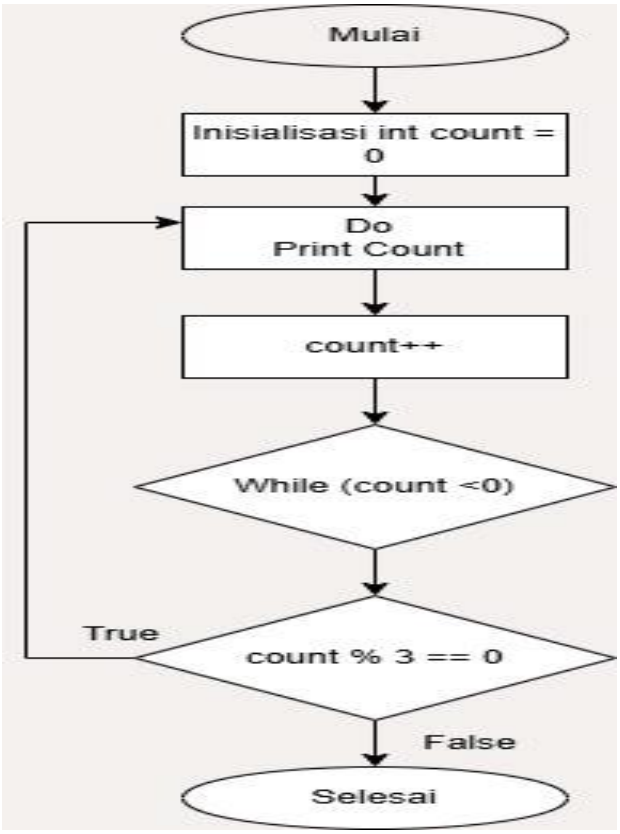
diagram flowchart dari Latihan contoh 5



Latihan 2.3
-dengan while



-dengan do while



[No.2] Kesimpulan

Kesimpulan ini menyoroti pentingnya penggunaan continue dan break dalam perulangan. Pada 2.1, perubahan dari `break` ke continue memungkinkan program melanjutkan pencetakan angka meskipun angka 4 dilewatkan. Di 2.2, perubahan modulus dari 3 ke 5 mengubah kelipatan angka yang dicetak. Pada 2.3, while tidak mengeksekusi kode jika kondisi awal salah, sementara do...while selalu menjalankan blok kode setidaknya sekali. Secara keseluruhan, analisis ini menegaskan pentingnya memahami kontrol alur untuk mencapai output yang diinginkan.

Refleksi:

dari pembelajaran ini menunjukkan bahwa memahami kontrol alur seperti continue, break, dan kondisi dalam perulangan sangat penting dalam pemrograman. Perubahan kecil dapat menghasilkan output yang berbeda, sehingga penting untuk menganalisis logika kode dengan baik. Selain itu, perbandingan antara while dan do-while menunjukkan bagaimana struktur perulangan mempengaruhi eksekusi. Secara keseluruhan, pembelajaran ini mengajarkan bahwa pemahaman mendalam membantu dalam membuat program yang efektif.