Nama & NPM	Topik:	Tanggal:
Sindi Putri Utami G1F024053	**FOR dan WHILE Java**	**26 September 2024**

[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!

[1] Analisis dan Argumentasi

1.1. Analisa kode pada contoh 1

1. Analisa tujuan penulisan kata kunci continue dan break pada Contoh 1
   Tujuan continue dan break:
   - continue: Kata kunci ini digunakan untuk melewatkan iterasi saat ini dari loop dan melanjutkan ke iterasi berikutnya. Dalam kode ini, jika y adalah bilangan ganjil (y % 2 == 1), maka continue akan membuat program melewati cetakan angka dan langsung melanjutkan ke nilai berikutnya.
   - break: Kata kunci ini digunakan untuk menghentikan loop sepenuhnya. Dalam kode ini, jika y mencapai 8, maka loop akan dihentikan.

2. Analisa dampak perubahan kode terhadap luaran setelah running
   a) Analisa kode asli
      Di kode ini, loop yang berjalan dari angka 0 sampai 10. Dalam setiap putaran:
      - Jika angka tersebut adalah angka ganjil (contohnya 1, 3, 5), program akan "lewat" dan tidak mencetaknya.
      - Jika angka itu mencapai 8, program akan "berhenti" dan tidak melanjutkan.
      - Jadi, hasil yang dicetak hanyalah angka genap dari 0 sampai 6.

   b) Analisa dampak perubahan setelah merubah nilai angka pada variabel di **//Ubah 1 menjadi for (int y = 0; y <= 15; y++)**

- Sebelum kode diubah, program memeriksa nilai dari 0 sampai 10, setelah diubah kode akan memeriksa nilai dari 0 sampai 15
- Karena kode ini masih memiliki aturan untuk melewati angka ganjil, maka yang dicetak hanya: 0,2,4,6
- Program memeriksa apakah nilai itu ganjil menggunakan **y % 2 == 1**. Program akan membagi nilai dengan 2, jika nilai tersebut memiliki sisa bagi, berarti nilai tersebut ganjil dan nilai tersebut akan dilewati dan tidak akan ditampilkan.
- Program tidak mencetak angka genap sampai 15, karena ada perintah **break** pada angka 8. Jadi program hanya akan mencetak angka-angka genap sebelum 8.
- Angka 8 tidak dicetak karena ada perintah **break** yang menghentikan proragram ketika mencapai angka 8. Jadi saat program mencapai angka 8 perintah menghentikan loop, dan angka 8 tidak ditampilkan.
- Jadi output yang dihasilkan hanya angka-angka genap sebelum 8, yaitu: 0,2,4, dan 6

c) Analisa dampak perubahan setelah merubah nilai angka pada variabel di **//Ubah 2 menjadi if (y % 2 == 0)**
- Setelah kode diubah, program akan melewatkan angka-angka genap dan hanya menampilkan angka-angka ganjil seperti: 1,3,5 dan seterusnya sampai mencapai angka 15
- Program memeriksa apakah nilai itu genap menggunakan **y % 2 == 0**. Program akan membagi nilai dengan 2, jika nilai tersebut habis dibagi dua berarti nilai tersebut genap, dan nilai tersebut akan dilewati dan tidak akan ditampilkan.
- Program terus mencetak angka-angka ganjil sampai 15 dan tidak berhenti di angka 8, karena angka 8 merupakan angka genap dan sebelumnya sudah dilewatkan oleh program
- Jadi output yang dihasilkan hanya angka-angka ganjil rentang angkanya dari 0 sampai 15, yaitu: 1,3,5,7,9,11,13, dan 15

d) Analisa dampak perubahan setelah merubah nilai angka pada variabel di **//Ubah 3 menjadi else if (y == 9)**
- Setelah kode diubah, program akan memeriksa nilai dan berhenti ketika mencapai nilai 9.
- Pada kode sebelumnya, program sudah diatur untuk melewatkan angka-angka genap, jadi program hanya akan mencetak nilai ganjil.
- Program tidak mencetak angka-angka ganjil sampai 15, karena ada perintah **break** pada angka 9. Jadi program hanya akan mencetak angka-angka ganjil sebelum 9.
- Angka 9 tidak dicetak karena ada perintah **break** yang menghentikan proragram ketika mencapai angka 9. Jadi saat program mencapai angka 9 perintah menghentikan loop, dan angka 9 tidak ditampilkan.
- Jadi output yang dihasilkan hanya angka-angka ganjil sebelum 9, yaitu: 1,3,5, dan 7.

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

a. Mengganti **break kedua;** dengan **continue pertama;**

- Kata kunci continue dalam pemrograman berfungsi untuk melewatkan bagian yang tersisa dari putaran saat ini dalam loop dan melanjutkan ke putaran berikutnya.
- Karena continue pertama; dieksekusi saat i mencapai 2, j = 1 dijalankan, dan program mencetak i = 2; j = 1. j = 2 tidak dijalankan, karena continue pertama; mengarahkan program untuk melanjutkan kembali ke loop luar.

b. Mengganti **break kedua;** dengan **break pertama;**
- Pernyataan break digunakan untuk menghentikan eksekusi dari loop. Ketika break dieksekusi, program akan keluar dari loop yang bersangkutan dan melanjutkan eksekusi setelah loop tersebut.
- Dengan menggunakan break pertama;, saat i mencapai 2, program tidak hanya menghentikan eksekusi loop kedua, tetapi juga keluar dari loop pertama sepenuhnya. Hal ini mengakibatkan tidak ada lagi iterasi untuk i yang lebih tinggi, dan hanya hasil untuk i = 1 dan sebagian dari i = 2 yang ditampilkan.

c. Mengganti **break kedua;** dengan **continue kedua;**
- continue kedua; digunakan untuk memengaruhi alur eksekusi dalam loop kedua yang memiliki label kedua.
- Pernyataan continue kedua; dalam kode ini membuat program melompati sisa bagian dari putaran saat ini dalam loop kedua (yang diberi label kedua) ketika nilai i sudah mencapai 2.
- hanya kombinasi i = 2; j = 1 yang dicetak, dan program tidak mencetak kombinasi untuk j = 2 saat i adalah 2.

1.3. Analisa kode pada contoh 3, setelah kode diubah.
- Kode **for(int s=tinggi; s>=t; s--){** di ubah menjadi **for(int s=1; s<=t; s++){**
- Loop Luar **(for(t=tinggi; t>=1; t--))**: Loop ini berjalan dari tinggi hingga 1. Ini berarti program akan mencetak baris-baris bintang dari baris tertinggi ke terendah.
- Sebelum kode diubah luaran akan menghasilkan piramida dari 1 bintang hingga tinggi, tapi dicetak dari bawah ke atas.
- Loop Luar (for(t=tinggi; t>=1; t--)): Sama dengan kode asli, berjalan dari tinggi hingga 1.
- Loop Dalam (for(s=1; s<=t; s++)): Perubahan utama terjadi di sini. Pada setiap iterasi t, loop ini mencetak bintang dari 1 hingga t, sehingga jumlah bintang yang dicetak berkurang sesuai dengan nilai t.
- Output: Hasilnya adalah piramida terbalik yang dimulai dari tinggi bintang hingga 1 bintang.

1.4. Analisa diagram flowchart dari Latihan 1.2 dan 1.3
- Flowchart 1.2: Menunjukkan kontrol alur yang kompleks dengan penggunaan break dan continue, serta bagaimana kedua perintah ini mengubah jalannya program.
- Flowchart 1.3: menunjukkan proses sederhana dari input pengguna hingga pengulangan untuk mencetak piramida terbalik, dengan fokus pada decrement untuk mencetak bintang.

[1] Penyusunan Algoritma dan Kode Program

1.1 kode pada contoh 1

1. Kode asli
   a. Algoritma
      1. Mulai.
      2. Inisialisasi y dengan 0.

3. Ulangi langkah :
    - Jika y adalah bilangan ganjil (y % 2 == 1):
      Lanjutkan ke langkah 5.
    - Jika y sama dengan 8:
      Keluar dari loop.
4. Cetak y.
5. Tambah 1 pada y (y = y + 1).
6. Kembali ke langkah 3.
7. Selesai.

Kode program dan luaran

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

```
java -cp /tmp/xQ1I5VoHA3/ContohFor
0
2
4
6

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

Luaran sudah sesuai dengan program yang disusun.

2. Kode ubah 1
    a. Algoritma
        1. Mulai.
        2. Inisialisasi y dengan 0.
        3. Ulangi langkah berikut hingga y lebih besar dari 15:
            - Jika y adalah bilangan ganjil (y % 2 == 1):
              Lanjutkan ke langkah 5.
            - Jika y sama dengan 8:
              Keluar dari loop.
            - Cetak y.
        4. Tambah 1 pada y (y = y + 1).
        5. Kembali ke langkah 3.
        6. Selesai.

Kode program dan luaran

```
1  public class ContohFor{
2  public static void main(String[] args) {
3          for (int y = 0; y <= 15; y++) {          //ubah 1
4              if (y % 2 == 1)              //ubah 2
5              continue;       //baris 1
6              else if (y == 8)             //ubah 3
7              break;          //baris 2
8              else
9              System.out.println(y + " ");
10     }    }    }
```

```
java -cp /tmp/blOWPn1iuK/ContohFor
0
2
4
6

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

Luaran sudah sesuai dengan program yang disusun.

3. Kode ubah 2
    a. Algoritma
        1. Mulai.
        2. Inisialisasi y dengan 0.
        3. Ulangi langkah berikut hingga y lebih besar dari 10:
            - Jika y adalah bilangan genap (y % 2 == 0):
              Lanjutkan ke langkah 5.
            - Jika y sama dengan 8:
              Keluar dari loop.
            - Cetak y.

4. Tambah 1 pada y (y = y + 1).
5. Kembali ke langkah 3.
6. Selesai.

Kode program dan luaran

```java
public class ContohFor{
public static void main(String[] args) {
        for (int y = 0; y <= 15; y++) {        //ubah 1
            if (y % 2 == 0)             //ubah 2
            continue;       //baris 1
            else if (y == 8)            //ubah 3
            break;              //baris 2
            else
            System.out.println(y + " ");
    }   }   }
```

```
java -cp /tmp/c8Q1QIhmMs/ContohFor
1
3
5
7
9
11
13
15

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

Luaran sudah sesuai dengan program yang disusun.

4. Kode ubah 3
   a. Algoritma
      1. Mulai.
      2. Inisialisasi y dengan 0.
      3. Ulangi langkah berikut hingga y lebih besar dari 10:
         - Jika y adalah bilangan ganjil (y % 2 == 1):
           Lanjutkan ke langkah 5.
         - Jika y sama dengan 9:
           Keluar dari loop.
         - Cetak y.
      4. Tambah 1 pada y (y = y + 1).
      5. Kembali ke langkah 3.
      6. Selesai.

Kode program dan luaran

```java
public class ContohFor{
public static void main(String[] args) {
        for (int y = 0; y <= 15; y++) {        //ubah 1
            if (y % 2 == 0)             //ubah 2
            continue;       //baris 1
            else if (y == 9)            //ubah 3
            break;              //baris 2
            else
            System.out.println(y + " ");
    }   }   }
```

```
java -cp /tmp/g8JhkI1xdI/ContohFor
1
3
5
7

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

Luaran sudah sesuai dengan program yang disusun.

1.2 perubahan kode pada Contoh 2 di baris //Ubah1

1. kode asli
   a. Algoritma
      1. Mulai.
      2. Inisialisasi i dengan 1.
      3. Ulangi langkah berikut hingga i kurang dari 5:
         - Inisialisasi j dengan 1.
         - Ulangi langkah berikut hingga j kurang dari 3:
           1. Cetak "i = i; j = j".
           2. Jika i sama dengan 2:
         - Keluar dari loop kedua (gunakan break kedua).
           3. Tingkatkan j dengan 1.
      4. Tingkatkan i dengan 1.
      7. Kembali ke langkah 3.
      6. . Selesai.

Kode program dan luaran

```
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
                   }    }    }   }
```

```
java -cp /tmp/wzIdiTlkVx/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 ===
```

Luaran sudah sesuai dengan program yang disusun.

2 Ubah kode pada //baris ubah 1 menjadi **continue pertama;**

a. Algoritma

1. Mulai.
2. Inisialisasi i dengan 1.
3. Ulangi langkah berikut hingga i kurang dari 5:
   - Inisialisasi j dengan 1.
   - Ulangi langkah berikut hingga j kurang dari 3:
4. Cetak "i = i; j = j".
5. Jika i sama dengan 2:
- Lanjutkan ke iterasi berikutnya dari loop pertama (lanjutkan dengan increment i).
   3. Tingkatkan j dengan 1.
4. Tingkatkan i dengan 1.
5. Kembali ke langkah 3.
6. Selesai.

Kode program dan luaran

```
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;      //ubah1
                   }    }    }   }
```

```
java -cp /tmp/jkUNIE7anL/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 ===
```

Luaran sudah sesuai dengan program yang disusun.

3 Ubah kode pada //baris ubah 1 menjadi **break pertama;**

a. Algoritma

1. Mulai.
2. Inisialisasi i dengan 1.
3. Ulangi langkah berikut hingga i kurang dari 5:
   - Inisialisasi j dengan 1.
   - Ulangi langkah berikut hingga j kurang dari 3:
     1. Cetak "i = i; j = j".
     2. Jika i sama dengan 2:
   - Keluar dari loop pertama (gunakan break pertama).
     3. Tingkatkan j dengan 1.
4. Tingkatkan i dengan 1.
5. Kembali ke langkah 3.
6. Selesai.

Kode program dan luaran

```java
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;      //ubah1
                }   }   }   }

```

```
java -cp /tmp/W5ewLrcAzH/ForBersarang
i = 1; j = 1
i = 1; j = 2
i = 2; j = 1

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

Luaran sudah sesuai dengan program yang disusun.

4 Ubah kode pada //baris ubah 1 menjadi **continue kedua;**

a. Algoritma

1. Mulai.
2. Inisialisasi i dengan 1.
3. Ulangi langkah berikut hingga i kurang dari 5:
   - Inisialisasi j dengan 1.
   - Ulangi langkah berikut hingga j kurang dari 3:
     1. Cetak "i = i; j = j".
     2. Jika i sama dengan 2:
        - Lanjutkan ke iterasi berikutnya dari loop kedua (gunakan continue kedua).
     3. Tingkatkan j dengan 1.
4. Tingkatkan i dengan 1.
5. Kembali ke langkah 3.
6. Selesai.

Kode program dan luaran

```java
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;      //ubah1
                }   }   }   }

```

```
java -cp /tmp/Orr8dKj2ia/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 ===
```

Luaran sudah sesuai dengan program yang disusun.

1.3 Kode contoh 3

a. Algoritma

1. Mulai.
2. Buat instance Scanner untuk input.
3. Cetak "Masukan Input: ".
4. Baca nilai tinggi dari input pengguna.
5. Ulangi langkah berikut dari tinggi hingga 1:
   - Inisialisasi t dengan nilai tinggi.
   - Ulangi langkah berikut dari 1 hingga t:
     Cetak karakter *.
   - Cetak baris baru.
6. Selesai.

Kode program dan luaran

```java
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=1; s<=t; s++){
                    //Menghitung Jumlah Spasi per Baris
                    System.out.print("*");
                }
                 System.out.println(); //Membuat Baris Baru
            }
        }
    }
```

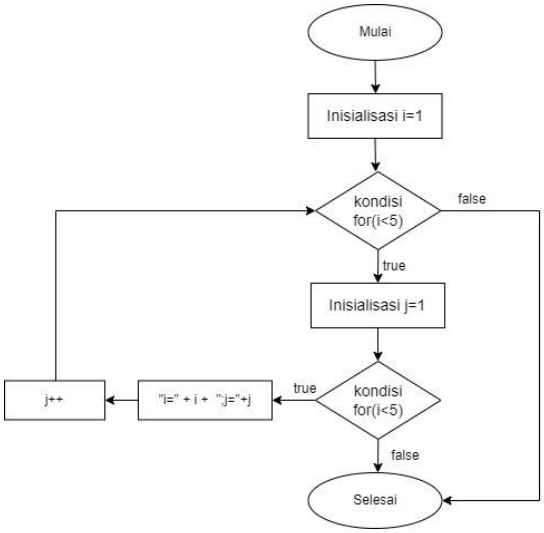
```
java -cp /tmp/lWjKOeMx9T/ForBersarang
Masukan Input: 7
*******
******
*****
****
***
**
*

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

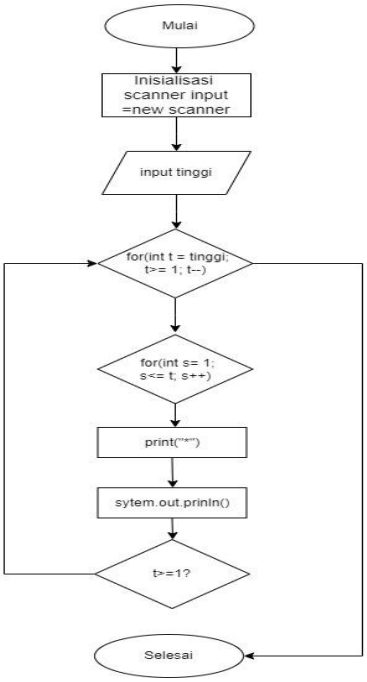
Luaran sudah sesuai dengan program yang disusun.

1.4. flowchart dari Latihan 1.2 dan 1.3

1. **Flowchart 1.2**



2. **Flowchart 1.3**



[1] Kesimpulan

kata kunci continue dan break sangat penting untuk mengontrol alur eksekusi. Di Contoh 1, continue melewatkan bilangan ganjil, sedangkan break menghentikan loop saat mencapai angka 8. Perubahan kode di Contoh 2 menunjukkan bahwa menggunakan continue pertama; akan melompati iterasi untuk i = 2, sementara break pertama; menghentikan seluruh loop. Di Contoh 3, untuk mencetak piramida terbalik, perlu mengubah logika loop agar jumlah bintang menurun per baris.
Penggunaan continue dan break dalam loop sangat memengaruhi cara program berjalan. Di latihan 1.2, continue membuat program melewatkan langkah tertentu, sedangkan break menghentikan loop sepenuhnya, mengubah hasil yang muncul. Sementara di latihan 1.3, mengatur loop sangat penting untuk mendapatkan pola yang diinginkan. Dengan memahami perbedaan antara continue dan break, dan cara mengatur loop dengan benar, bisa lebih mudah mengontrol alur program dan mendapatkan output yang tepat.

[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!

[2] Analisis dan Argumentasi

2.1. Analisa dampak perubahan terhadap luaran setelah kode break pada //ubah satu menjadi continue

- Sebelum kode diubah, program menjalankan perintah break di dalam loop, ketika nilai i mencapai 4 kondisi if terpenuhi dan perintah break dijalankan. kondisi ini membuat loop berhenti sebelum mencapai angka 4. Jadi saat program mencapai angka 4 perintah menghentikan loop, dan dan hanya menampilkan angka 1,2, dan 3, angka 4,5, dan 6 tidak ditampilkan.
- Ketika kode break diganti dengan kode continue program akan mencetak semua angka, yaitu angka 1 sampai 6.

2.2. Analisa dampaknya perubahan terhadap luaran setelah perubahan nilai angka pada variabel di Contoh 5 dan uraikan kegunaan % untuk angka yang berbeda pada perintah.

- Sebelum kode diubah, program akan mencetak angka angka yang bisa dibagi 3 (kelipatan 3). Dan luarannya adalah: 0,3,6,9,12,15, dan 18
- Setelah kode diubah, program akan mencetak angka angka yang bisa dibagi 5 (kelipatan 5). Dan luarannya adalah: 0,5,10,15.
- count % 3 == 0:

ini memeriksa apakah angka bisa dibagi 3 tanpa ada sisa.

- count % 5 == 0:ini memeriksa apakah angka bisa dibagi 5 tanpa ada sisa.
- Perubahan dari 3 ke 5 membuat kode mencetak angka yang berbeda. Dengan menggunakan tanda %.

2.3. Membandingkan hasil luaran antara menggunakan while dan do … while pada contoh 5

3. while (count < 0)
    - Loop ini hanya akan berjalan jika kondisi count < 0 benar.
    - Karena count dimulai dari 0, kondisi ini tidak terpenuhi (0 tidak kurang dari 0).
    - Karena kondisinya salah dari awal, kode di dalam loop tidak dijalankan sama sekali. Jadi, tidak ada yang ditampilkan.
4. do...while (count < 0)
    - Loop ini akan menjalankan kode di dalamnya setidaknya sekali, terlepas dari kondisi yang diberikan.
    - Saat program dijalankan, kode di dalam do akan dieksekusi pertama kali. Di sini, count diperiksa untuk kelipatan 3 dan kemudian ditambah 1.
    - Setelah eksekusi pertama, count menjadi 1.
    - Setelah itu, kondisi count < 0 diperiksa lagi. Karena count sekarang adalah 1 (yang tidak kurang dari 0), kondisi ini tidak terpenuhi.
    - Meskipun kode di dalam do telah dijalankan, tidak ada yang ditampilkan karena tidak ada kelipatan 3 yang dicetak.

- while (count < 0): Tidak ada yang dijalankan, sehingga tidak ada output.
- do...while (count < 0): Kode dijalankan sekali, tetapi tidak ada yang ditampilkan karena count tidak memenuhi syarat untuk mencetak.

2.4. Analisa diagram flowchart dari Latihan 2.1, Contoh 5, dan Latihan 2.3
- Flowchart 2.1: Mengilustrasikan bagaimana penggunaan continue memengaruhi eksekusi, dengan melompati langkah tertentu.
- Flowchart Contoh 5: Menunjukkan penggunaan loop while dengan kondisi untuk mencetak kelipatan 3.
- Flowchart 2.3: Menggambarkan bagaimana do...while memastikan setidaknya satu langkah terjadi, tanpa memperhatikan kondisi awal.

[2] Penyusunan Algoritma dan Kode Program

2.1. perubahan nilai angka pada variabel di Contoh 4

5. kode asli

a. Algoritma

1. Mulai.
2. Inisialisasi variabel count dengan 0.
3. Selama count kurang dari 20, lakukan langkah berikut:
    - Jika count dibagi 3 menghasilkan sisa 0:

    Cetak nilai count.
    - Tingkatkan count dengan 1.
4. Selesai.

Kode program dan luaran

```
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                  break;          //ubah1
9  }}}}
```

```
java -cp /tmp/dWL3W1O1sY/ContohWhile
1
2
3

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

Luaran sudah sesuai dengan program yang disusun.

2. kode setelah diubah

a. Algoritma

1. Mulai Program:
   - Buat sebuah angka i dan mulai dari 1.
2. Mulai Loop:
   - Selama angka i kurang dari atau sama dengan 6, lakukan langkah-langkah berikut:
3. Tampilkan Angka:
   - Tampilkan angka i.
4. Naikkan Angka:
   - Tambahkan 1 pada angka i.
5. Cek Angka:
   - Jika angka i sudah menjadi 4:
   - Lewati langkah selanjutnya dan kembali ke langkah 2.
6. Akhiri Loop:
   Ulangi langkah 2 sampai angka i lebih dari 6.

Kode program dan luaran

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

```
java -cp /tmp/uaKNtWqKl8/ContohWhile
1
2
3
4
5
6

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

Luaran sudah sesuai dengan program yang disusun.

2.2. perubahan nilai angka pada variabel di Contoh 5

1. kode asli

a. Algoritma

1. Mulai Program:
   - Buat sebuah variabel count dan set nilainya ke 0.
2. Mulai Loop:
   - Selama count kurang dari 20, lakukan langkah-langkah berikut:
3. Cek Kelipatan:
   - Jika count dapat dibagi 3 tanpa sisa (yaitu, count % 3 == 0):
   - Tampilkan nilai count.
4. Naikkan Angka:
   - Tambahkan 1 ke nilai count.
5. Kembali ke Loop:
   - Ulangi langkah 2 hingga nilai count mencapai 20.
6. Akhiri Program:
   - Program selesai setelah loop berakhir.

Kode program dan luaran

```
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++;
        }
        }
    }
```

```
java -cp /tmp/UgfdCnvoDC/WhileBersarang
0
3
6
9
12
15
18

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

Luaran sudah sesuai dengan program yang disusun.

3. kode setelah diubah
   a. algoritma
      1. Mulai Program:
         - Buat sebuah variabel count dan set nilainya ke 0.
      2. Mulai Loop:
         - Selama count kurang dari 20, lakukan langkah-langkah berikut:
      3. Cek Kelipatan:
         - Jika count dapat dibagi 5 tanpa sisa (yaitu, count % 5 == 0):
         - Tampilkan nilai count.
      4. Naikkan Angka:
         - Tambahkan 1 ke nilai count.
      5. Kembali ke Loop:
         - Ulangi langkah 2 hingga nilai count mencapai 20.
      6. Akhiri Program:
         - Program selesai setelah loop berakhir.

Kode program dan luaran

```
1  public class WhileBersarang {
2      public static void main(String[] args) {
3          int count = 0;  //ubah1
4          while (count < 20) {
5          if (count % 5 == 0)  //ubah2
6          System.out.println(count);
7          count++;
8          }
9          }
10     }
```

```
java -cp /tmp/A4lotjig9p/WhileBersarang
0
5
10
15

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

Luaran sudah sesuai dengan program yang disusun.

2.3. perubahan nilai angka pada variable kode while pada Contoh 5

1. kode asli

a. algoritma
   1. Mulai Program:
      - Buat sebuah variabel count dan set nilainya ke 0.
   2. Mulai Loop:
      - Selama count kurang dari 20, lakukan langkah-langkah berikut:
   3. Cek Kelipatan:
      - Jika count dapat dibagi 3 tanpa sisa (yaitu, count % 3 == 0):
      - Tampilkan nilai count.
   4. Naikkan Angka:
      - Tambahkan 1 ke nilai count.
   5. Kembali ke Loop:
      - Ulangi langkah 2 hingga nilai count mencapai 20.
   6. Akhiri Program:
      - Program selesai setelah loop berakhir.

Kode program dan luaran

```
1  public class WhileBersarang {
2      public static void main(String[] args) {
3          int count = 0;  //ubah1
4          while (count < 20) {
5          if (count % 3 == 0)  //ubah2
6          System.out.println(count);
7          count++;
8          }
9          }
10     }
```

```
java -cp /tmp/UgfdCnvoDC/WhileBersarang
0
3
6
9
12
15
18

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

Luaran sudah sesuai dengan program yang disusun.

2. kode setelah diubah while

a. algoritma

1. Mulai Program:
   - Buat sebuah variabel count dan set nilainya ke 0.
2. Mulai Loop:
   - Selama count kurang dari 0, lakukan langkah-langkah berikut:
3. Cek Kelipatan:
   - Jika count dapat dibagi 3 tanpa sisa (yaitu, count % 3 == 0):
   - Tampilkan nilai count.
4. Naikkan Angka:

- Tambahkan 1 ke nilai count.

5. Kembali ke Loop:

- Ulangi langkah 2 hingga nilai count tidak kurang dari 0.

6. Akhiri Program:

- Program selesai setelah loop berakhir.

Kode program dan luaran

```
1  public class WhileBersarang {
2      public static void main(String[] args) {
3          int count = 0;  //ubah1
4          while (count < 0) {
5          if (count % 3 == 0)  //ubah2
6          System.out.println(count);
7          count++;
8          }
9          }
10     }
```

```
java -cp /tmp/RCtHumumCO/WhileBersarang

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

Luaran sudah sesuai dengan program yang disusun.

3. baris kode while pada Contoh 5 menjadi do … while dengan persyaratan yang sama while (count < 0).

a. algoritma

1. Inisialisasi:
   - Set count ke 0.
2. Loop (do-while):
   - Lakukan:
   - Jika count % 3 == 0, tampilkan count.
   - Tambah count dengan 1.
   - Ulangi selama count < 0.
3. Akhiri Program:
   - Program selesai.

Kode program dan luaran

```
1  public class WhileBersarang {
2      public static void main(String[] args) {
3          int count = 0;  //ubah1
4          do {
5              if (count % 3 == 0){
6                  System.out.println(count);
7              }
8              count++;
9          } while (count < 0);
10         }
11     }
```

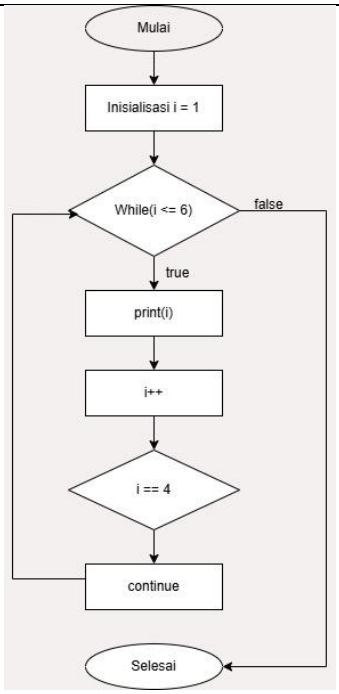
```
java -cp /tmp/BX41cSGWCj/WhileBersarang
0

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

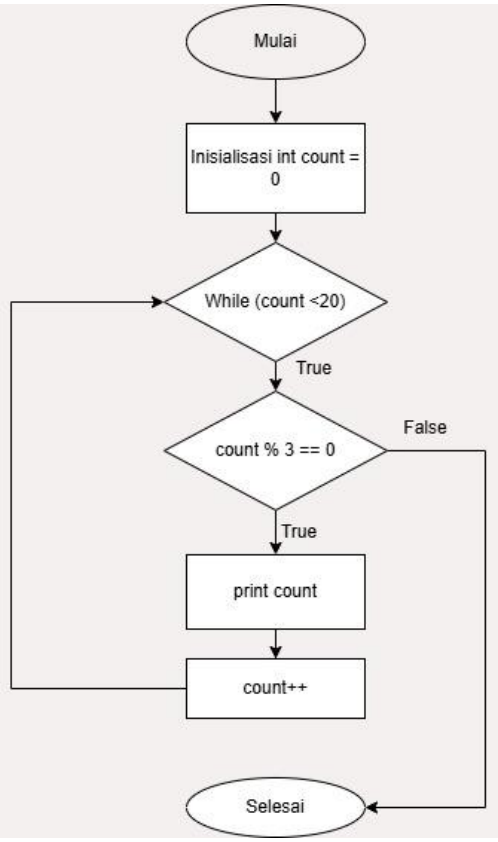
Luaran sudah sesuai dengan program yang disusun.

2.4. flowchart dari Latihan 2.1, Contoh 5, dan Latihan 2.3

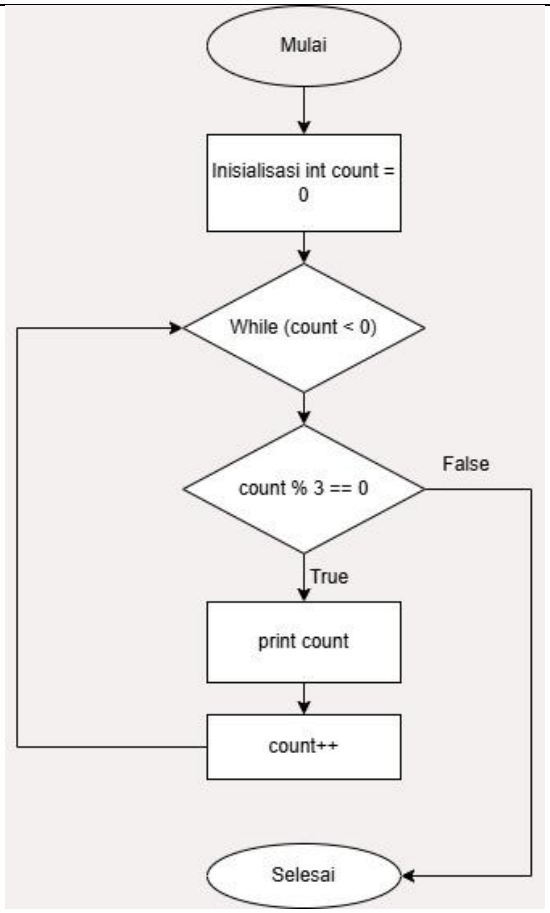
1. flowchart 2.1



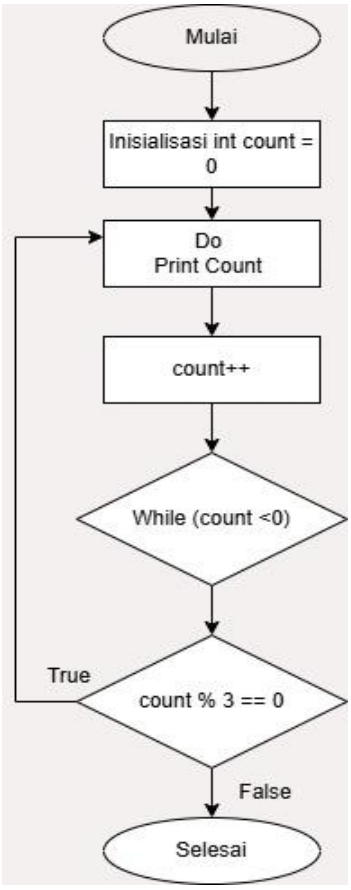
2. flowchart contoh 5



3. flowchart 2.3
   1.while



2. Do While



[2] Kesimpulan

perubahan pada program menggunakan break dan continue menunjukkan dampak yang jelas terhadap keluaran. Pada latihan 2.1, mengganti break dengan continue mengubah output dari 1, 2, dan 3 menjadi 1, 2, 3, 5, dan 6. Pada latihan 2.2, mengganti kelipatan 3 dengan 5 mengubah output menjadi 0, 5, 10, dan 15, yang menunjukkan kegunaan operator modulus (%). Pada latihan 2.3, mengubah while (count < 20) menjadi while (count < 0) dan menggunakan do ... while menghasilkan tidak ada output, karena kondisi tidak terpenuhi.

Refleksi

Minggu ini belajar tentang penggunaan pernyataan break dan continue. Tantangannya adalah menyadari perubahan kecil pada kode bisa mengubah hasil dari keseluruhan.