Nama & NPM	Topik:	Tanggal:
Aditya Bagas Setiawan(G1F024051)	Perulangan: FOR dan WHILE	09/10/2024

[Nomor 1] Identifikasi Masalah:

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!

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)!

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

4) Analisa diagram flowchart dari Latihan 1.2 dan 1.3!

[Nomor 1] Analisis dan Argumentasi

1.1

A.Tujuan continue adalah untuk menghentikan eksekusi dan melanjutkan ke berikutnya. Pada contoh 1, jika nilai y adalah bilangan ganjil, eksekusi melompat ke bagian berikutnya tanpa menjalankan perintah System.out.println(y + " ");.

Tujuan break adalah untuk Menghentikan eksekusi seluruh loop. Pada contoh ini, ketika nilai y mencapai 8, loop dihentikan dan tidak ada eksekusi lebih lanjut.

B. Ubah 1 menjadi for (int y = 0; y <= 15; y++)

```
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         }  }  }
```

```
0
2
4
6
```

Ini memperpanjang rentang loop dari 0 hingga 15. Dengan rentang ini, kita akan melihat lebih banyak nilai, dan break akan tetap menghentikan loop ketika y mencapai 8.

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

```
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 == 0)                 //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         }  }  }
```

```
1
3
5
7
9
11
13
15
```

Perubahan ini membuat loop hanya mencetak angka ganjil (karena sekarang continue dieksekusi jika y adalah bilangan genap). Hasilnya, hanya bilangan ganjil di bawah 8 yang akan dicetak.

D. Ubah 3 menjadi else if (y == 9)

```
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 == 0)                  //ubah 2
5              continue;       //baris 1
6              else if (y == 9)             //ubah 3
7              break;             //baris 2
8              else
9              System.out.println(y + " ");
10         }  }  }
```

```
1
3
5
7
```

Perubahan ini menyebabkan loop berhenti saat y mencapai 9. Hasilnya, angka 9 menjadi batas untuk loop.

1.2

A. continue pertama

```
1  public class ForBersarang {
2      public static void main(String[] args) {
3          pertama:
4              for( int i = 1; i < 5; i++) {
5
6
7                  kedua:
8                  for(int j = 1; j < 3; j ++ ) {
9                      System.out.println("i = " + i + "; j = " +j);
10                     if ( i == 2)
11                         continue pertama;       //ubah1
12
13                 }
14             }
15     }
16 }
17
```

```
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
```

Perintah continue akan melompat ke berikutnya dari loop luar (pertama). Ini berarti jika kondisi if (i == 2) terpenuhi, loop i akan langsung melanjutkan ke nilai berikutnya tanpa menyelesaikan loop j yang ada.

B. break pertama

```
1  public class ForBersarang {
2          public static void main(String[] args) {
3               pertama:
4                     for( int i = 1; i < 5; i++) {
5
6                         kedua:
7                         for(int j = 1; j < 3; j ++ ) {
8                             System.out.println("i = " + i + "; j = " +j);
9                          if ( i == 2)
10                                                break pertama;      //ubah1
11                      }    }    }   }
```

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

break akan menghentikan loop pertama sepenuhnya jika i == 2, sehingga setelah loop mencapai nilai i == 2, seluruh program akan berhenti.

C. continue kedua

```
1  public class ForBersarang {
2          public static void main(String[] args) {
3               pertama:
4                     for( int i = 1; i < 5; i++) {
5
6                         kedua:
7                         for(int j = 1; j < 3; j ++ ) {
8                             System.out.println("i = " + i + "; j = " +j);
9                          if ( i == 2)
10                                                continue kedua;      //ubah1
11                      }    }    }   }
```

```
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
```

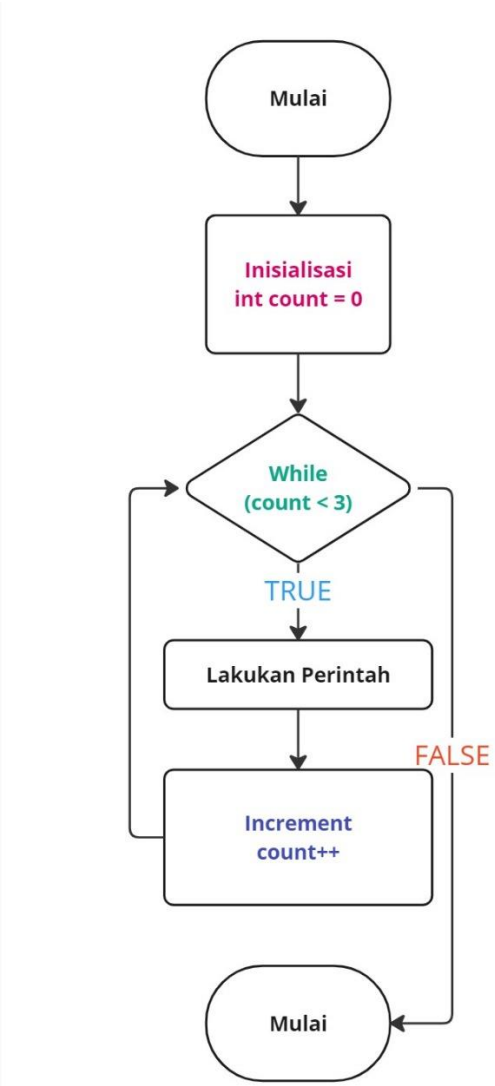
continue pada loop kedua akan melompat ke berikutnya dari j ketika kondisi i == 2 terpenuhi, sehingga nilai j tidak akan dicetak sepenuhnya.

1.3

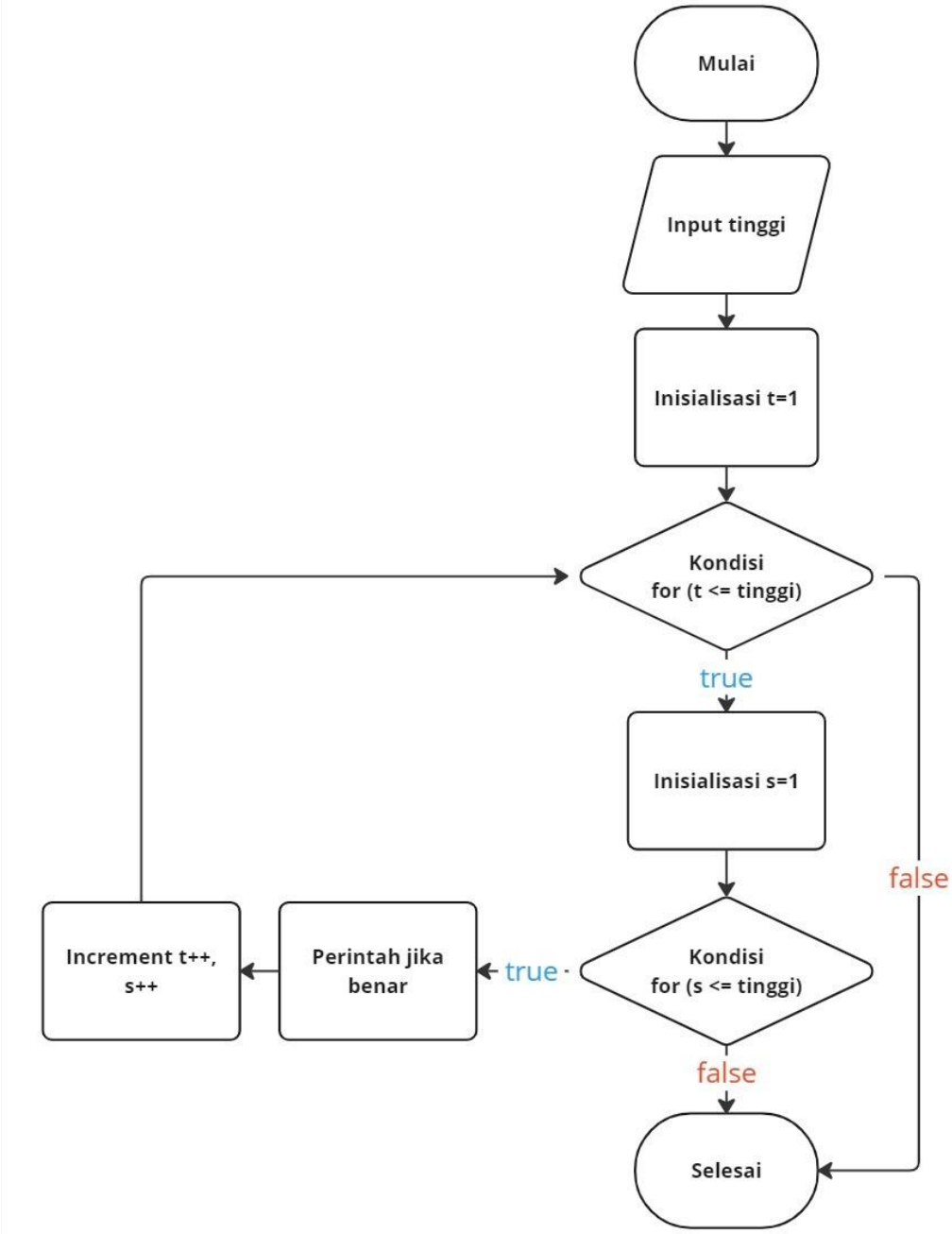
```java
import java.util.Scanner;

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 = 1; t <= tinggi; t++){    // Mengubah loop t untuk mulai dari 1 hingga tinggi
            for(int s = tinggi; s >= t; s--){ // Mengubah spasi, mencetak lebih sedikit bintang setiap baris
                System.out.print("*");
            }
            System.out.println(); // Membuat baris baru
        }
    }
}

```

```
Masukan Input: 7
*******
******
*****
****
***
**
*
```

1.4 Flowchart 1.2



Flowchart 1.3



[Nomor 1] Kesimpulan

Evaluasi dari ketiga contoh program menunjukkan pentingnya memahami dan menggunakan kontrol alur dalam loop secara efektif, terutama dalam penggunaan kata kunci continue dan break. Perubahan pada struktur loop dapat menghasilkan luaran yang sangat berbeda, baik dalam hal angka yang dicetak pada output (contoh 1 dan 2) maupun pola visual yang dihasilkan (contoh 3).

Secara umum, hasil ini menunjukkan bahwa:

continue digunakan untuk melompati eksekusi iterasi tertentu dan langsung melanjutkan ke

iterasi berikutnya.

break digunakan untuk menghentikan seluruh eksekusi loop secara paksa ketika kondisi tertentu terpenuhi.

Struktur dan urutan loop sangat menentukan hasil akhir yang diinginkan, terutama ketika bekerja dengan pola atau pengulangan bersarang.

Dengan memahami logika ini, kita dapat mengontrol eksekusi program secara lebih efisien, menyesuaikan hasil sesuai kebutuhan, baik dalam skenario matematis maupun visual.

Template Lembar Kerja Individu dan Kelompok

Nama & NPM	Topik:	Tanggal:
Aditya Bagas Setiawan (G1F024051)	Perulangan : FOR dan WHILE	09/10/2024

[Nomor Soal] 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!

[Nomor 2] Analisis dan Argumentasi

2.1.

```
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         } } } }
```

```
1
2
3
4
5
6
```

Setelah menggunakan continue, ketika nilai i menjadi 4, perintah continue menyebabkan perulangan langsung lompat ke berikutnya tanpa menjalankan perintah di bawahnya. Ini artinya angka 4 dilewati dari output.

2.2.

```
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  }
```

```
0
5
10
15
```

Operator modulus % digunakan untuk mendapatkan sisa hasil bagi. Jika count % 3 == 0, maka count adalah kelipatan dari 3. Ketika diubah menjadi count % 5 == 0, maka program hanya mencetak angka yang merupakan kelipatan dari 5.

2.3.

```
1  public class WhileBersarang {
2      public static void main(String[] args) {
3          int count = 0;  //ubah1
4          do {
5      //...
6          } while (count < 0); {
7          if (count % 3 == 0)  //ubah2
8          System.out.println(count);
9          count++;
10         }
11         }
12     }
```

```
0
```

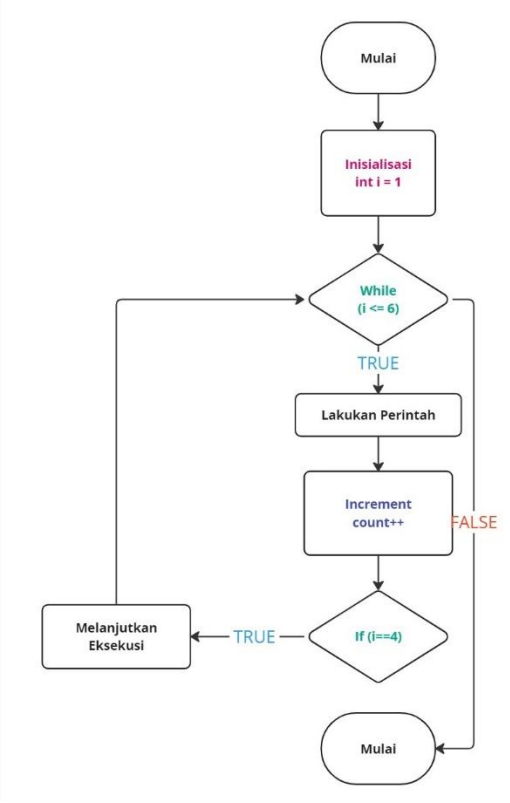
Perbedaan antara while dan do-while:

while : Kondisi diperiksa sebelum menjalankan blok kode. Jika kondisi tidak terpenuhi, blok tidak akan dijalankan.

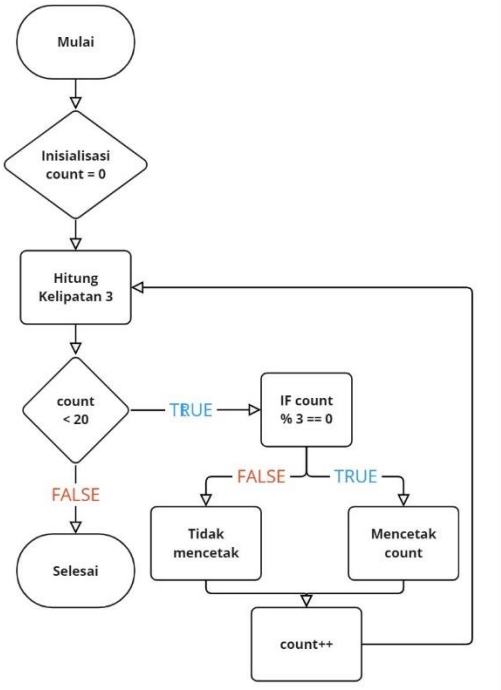
do-while : Blok kode dijalankan terlebih dahulu, baru kemudian kondisi diperiksa. Oleh karena itu, blok akan dijalankan setidaknya sekali, meskipun kondisi tidak terpenuhi.

2.4.

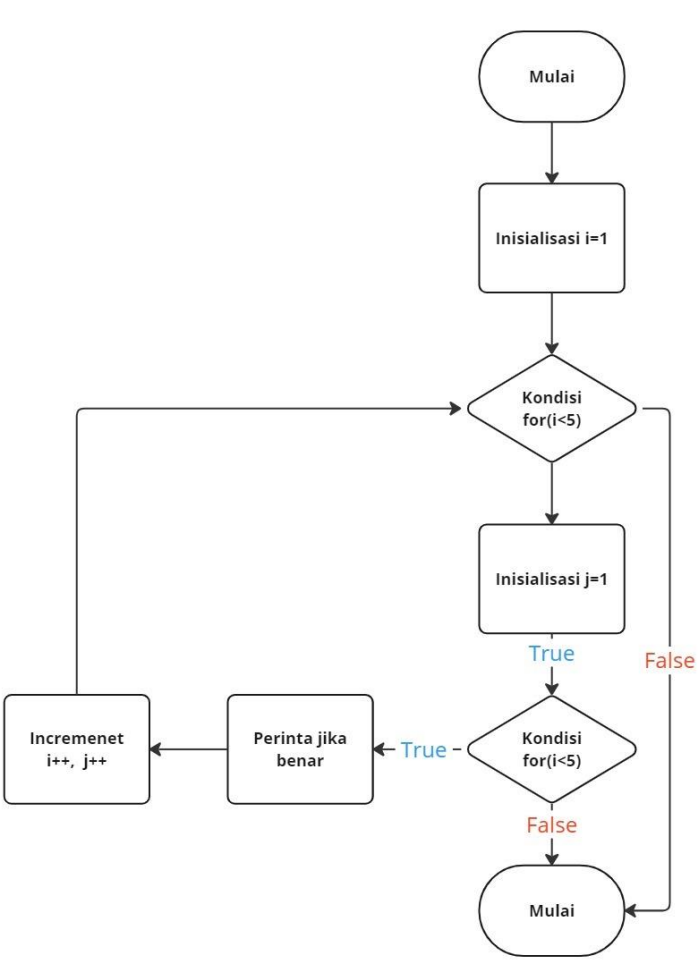
Flowchart 2.1



Flowchart contoh 5



Flowchart 2.3



[Nomor 2] Kesimpulan

perintah break digunakan untuk menghentikan perulangan secara keseluruhan ketika suatu kondisi terpenuhi, sedangkan continue hanya melewatkan iterasi saat kondisi tersebut terjadi tanpa menghentikan perulangan. Operator modulus (%) berfungsi untuk menentukan kelipatan angka, sehingga perubahan angka pada modulus (misalnya dari 3 menjadi 5) akan mengubah output dengan mencetak kelipatan yang berbeda. Selain itu, terdapat perbedaan antara perulangan while dan do-while, di mana while memeriksa kondisi terlebih dahulu sebelum menjalankan perulangan, sedangkan do-while akan menjalankan perulangan setidaknya sekali sebelum mengecek kondisi. Diagram flowchart menunjukkan bagaimana percabangan terjadi dalam perulangan ketika menggunakan break atau continue, serta perbedaan alur antara while dan do-while dalam memeriksa kondisi dan menjalankan perulangan.