MODUL 8

PERULANGAN 2

KOMPETENSI

- Mahasiswa memahami **konsep** perulangan bersarang
- Mahasiswa dapat menjelaskan format penulisan perulangan bersarang (*nested loop*)
- Mahasiswa dapat mengimplementasikan *flowchart* perulangan bersarang menggunakan bahasa pemrograman Java

MATERI DASAR

Pengertian Perulangan Bersarang (Nested Loop)

Pada bahasan sebelumnya, telah dibahas tentang konsep dasar perulangan. Pada bahasan tersebut disebutkan bahwa **logika perulangan** digunakan untuk melakukan beberapa proses atau statement program **secara berulang-ulang**, dengan suatu pola tertentu. Pada **perulangan**, proses atau satetement akan terus dilakukan **secara berulang-ulang**, selama **kondisi perulangan** bernilai **benar/true**. Dan sebaliknya, **perulangan akan berhenti** dan proses atau statement tidak akan dieksekusi lagi ketika **kondisi perulangan** bernilai **salah/false**. Jadi, dalam logika perulangan, suatu kondisi perulangan diperlukan untuk menentukan apakah suatu perulangan masih akan akan berlangsung lagi atau harus berhenti.

Perulangan bersarang (nested loop) adalah struktur perulangan yang berada di dalam perulangan lainnya. Pada umumnya, struktur perulangan yang berada di dalam perulangan lainnya tersebut memiliki hubungan yang saling terkait dalam menyesuaikan sebuah kasus. Pada dasarnya tidak ada batasan dalam jumlah perulangan bersarang. Tetapi sebaiknya tidak terlalu dalam, untuk menghindari kompleksitas yang tinggi serta alur program menjadi lebih sukar untuk dipahami.

Perulangan bersarang minimal terdapat 2 tingkat/level, akan tetapi *nested loop* dapat memiliki lebih dari 2 tingkat. Secara umum, *nested loop* dapat dituliskan dalam bentuk *pseudocode* yang sangat sederhana. Perulangan bersarang dengan *pseudocode* bisa dituliskan sebagai berikut:

```
loop-level-1 {
    loop-level-2 {
        .....
        loop-level-n {
            // statement
        }
        .....
    }
}
```

Ilustrasi Perulangan Bersarang

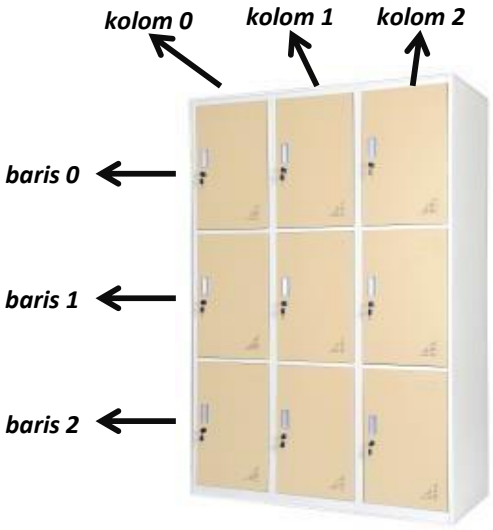
Sebagai ilustrasi sederhana tentang cara kerja perulangan bersarang, misalkan terdapat loker (rak penyimpanan) yang memiliki 9 pintu penyimpanan. Terdapat aktivitas dimana kita ingin mengetahui isi di setiap pintu loker. Maka kita akan membuka pintu satu persatu pada loker tersebut.

Kita mengunjungi baris pertama, dan membuka setiap pintu loker satu persatu (pintu 1, 2, dan 3). Kemudian kita mengunjungi baris selanjutnya yaitu baris kedua dan membuka pintu loker pada baris kedua tersebut satu persatu (pintu 4, 5, dan 6). Terakhir kita mengunjungi baris ketiga dan membuka pintu loker satu persatu (pintu 7, 8, dan 9).

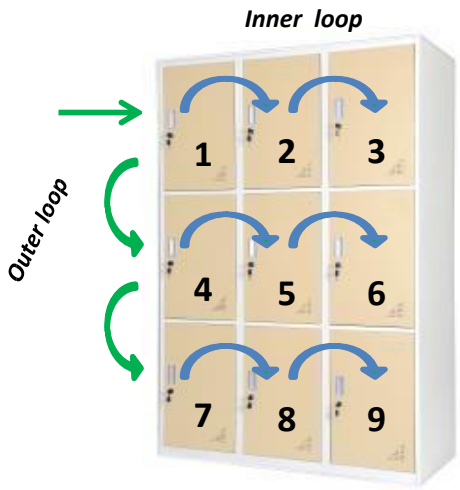
Gambar 1. Loker Penyimpanan

Proses kita mengunjungi baris pertama, kemudian baris kedua dan ketiga merupakan sebuah perulangan dan disebut perulangan luar (***outer loop***). Disebut *outer loop* karena aktivitas perulangan tersebut kita lakukan pertama kali.

Selanjutnya, saat kita membuka pintu loket satu persatu pada baris pertama, kemudian membuka lagi pintu loker satu persatu pada baris kedua dan ketiga. Merupakan sebuah perulangan juga dan disebut dengan perulangan dalam (***inner loop***).



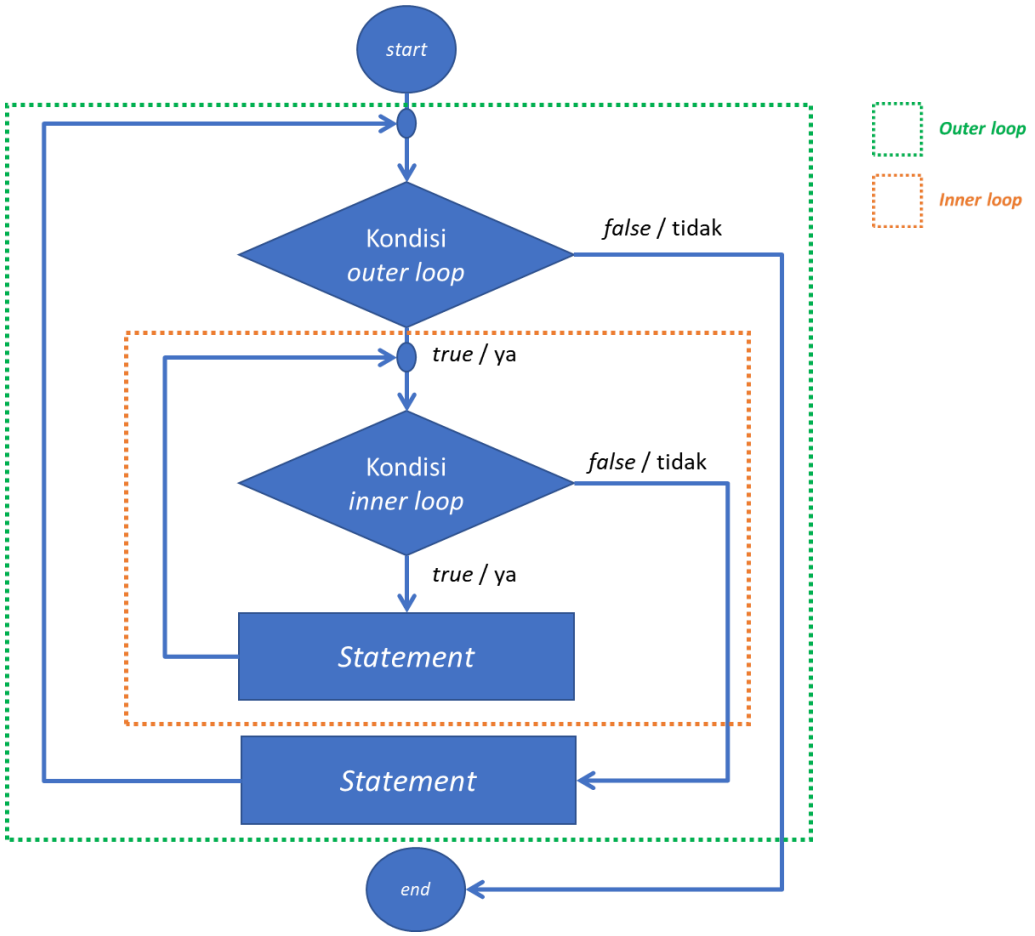
Gambar 2. Baris Kolom pada loker



Gambar 3. Ilustrasi Outer loop dan Inner Loop

Flowchart Nested Loop

Secara umum dan sederhana, flowchart perulangan bersarang ditunjukan pada Gambar 4. Pada gambar tersebut terlihat terdapat perulangan dalam yang berada di dalam perulangan luar. Tiap perulangan memiliki kondisi sendiri-sendiri dan statement masing-masing.



Gambar 4. *Flowchart Nested Loop*

Sintaks Nested Loop

Semua sintaks perulangan yang telah dibahas sebelumnya, seperti *for*, *while* dan *do-while*, semuanya bisa digunakan untuk kasus perulangan bersarang. Dan tidak ada aturan yang mengharuskan menggunakan sintaks yang sama antara perulangan luar dan perulangan yang ada di dalamnya.

Berdasarkan pada Gambar 2 dan Gambar 3, secara kode program ***outer loop*** kita identifikasi sebagai penunjuk **baris** dan ***inner loop*** kita identifikasi sebagai penunjuk **kolom**.

```
for(int baris = 0; baris < 3; baris++) {
    for(int kolom = 0; kolom < 3; kolom++){
        // statement
    }
}
```

Nested loop tidak hanya berupa satu jenis *loop/*perulangan yang bertingkat, akan tetapi bisa kombinasi *loop* yang bertingkat.

```
/* Kombinasi for dan do-while loop */
for(int i = 0; i < 10; i++){
    int j = 0;
    do {
        // statement
        j++;
    } while(j < 10);
}
```

```
/* Kombinasi while dan do-while loop */
int i = 0;
while(i < 10) {
    int j = 0;
    do {
        // statement
        j++;
    } while(j < 10);

    i++;
}
```

```
/* Kombinasi while dan for loop */
int i = 0;
while(i < 10) {
    for(int j = 0; j < 10; j++) {
        // statement
    }
    i++;
}
```

```
/* Kombinasi do-while dan for loop */
int i = 0;
do {
    for(int j = 0; j < 10; j++) {
        // statement
    }
    i++;
} while(i < 10);
```

Percobaan 1

1. Percobaan ini ditujukan me-review kembali perulangan yang sudah dibahas pada pertemuan sebelumnya. Pada percobaan 1 akan dibuat program untuk membuat tampilan * sebanyak N kali ke arah **samping**.
2. Buat project baru dengan nama **Star** dan simpan dalam file **Star.java**
3. Karena program membutuhkan input dari keyboard, maka perlu import class Scanner. Jadi tambahkan sintaks import di baris atas sendiri program.

```
import java.util.Scanner;
```

4. Di dalam fungsi **main()** yang telah dibuat, deklarasikan objek **Scanner** dengan nama **sc**.

```
Scanner sc = new Scanner(System.in);
```

5. Pada baris selanjutnya, tampilkan instruksi untuk memasukan nilai yang akan disimpan ke variabel **N**.

```
System.out.print("Masukkan nilai N : ");
int N = sc.nextInt();
```

6. Pada baris selanjutnya, buat sintaks perulangan dengan for seperti di bawah ini.

```
for(int i=1; i<=N; i++){
    System.out.print("*");
}
```

Catatan: perlu diperhatikan,bahwa yang digunakan adalah perintah `print`, bukan `println` karena kita ingin menampilkan tanpa ada baris baru.

7. Compile dan jalan program!
8. Amati hasilnya, maka hasilnya harusnya akan serupa dengan tampilan di bawah ini.

```
run:
Masukkan nilai N : 50
**************************************************
BUILD SUCCESSFUL (total time: 4 seconds)
```

Percobaan 2

1. Pada percobaan ke-2 akan dilakukan percobaan tentang nested loop. Kasus yang akan diselesaikan adalah untuk membuat tampilan bujursangkar *, dengan panjang sisi sebanyak N. Misalkan N dimasukan 10, maka hasilnya adalah:

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

2. Kalau diamati lebih lanjut, sebenarnya mirip dengan kasus percobaan 1 bukan? Jika di percobaan 1, misal input N bernilai 5, maka yang akan dihasilkan adalah ***** (kita bisa anggap ini sebagai inner loop yang mencetak 5 bintang *****), maka untuk kasus percobaan 2 ini

bukankah hasil dari percobaan 1 tersebut hanya perlu diulang lagi sebanyak N kali? (dengan menambahkan outer loop untuk mengulangi proses inner loop sebanyak N kali.)

3. Buat project baru dan simpan dengan nama file **Square.java**.
4. Karena program membutuhkan input dari keyboard, maka perlu import class Scanner. Jadi tambahkan sintaks import di baris atas sendiri program.

```
import java.util.Scanner;
```

4. Buat method **main()**, dan isikan kode program yang sama dengan isi method **main()** di percobaan 1. Perhatikan sintaks perulangan yang digunakan untuk mencetak * sebanyak N kali ke arah samping. Di step-6 di atas kode **for** (kotak merah) kita jadikan sebagai ***inner loop***.
5. Kita looping lagi inner loop sebanyak N kali untuk menghasilkan output seperti tahap 1. Maka perlu ditambahkan perulangan luar (***outer loop***, menjadi:

```
public static void main (String [] args){
    Scanner sc = new Scanner(System.in);

    System.out.print("Masukkan nilai N : ");
    int N = sc.nextInt();

    for(int iOuter=1; iOuter<=N; iOuter++){
        for(int i=1; i<=N; i++){
            System.out.print("*");
        }
        System.out.println("");
    }
}
```

6. Amati hasilnya, maka hasilnya harusnya akan serupa dengan tampilan di bawah ini.

```
run:
Masukkan nilai N : 10
**********
**********
**********
**********
**********
**********
**********
**********
**********
**********
BUILD SUCCESSFUL (total time: 3 seconds)
```

Percobaan 3

1. Pada percobaan ke-3 akan dilakukan percobaan segitiga *, dengan sama siku dengan tinggi sebesar N. Misalkan N dimasukan 10, maka hasilnya adalah:

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

2. Buat class **Triangle** dan simpan dengan nama file **Triangle.java**
3. Karena program membutuhkan input dari keyboard, maka perlu import class Scanner.
4. Salin kode program berikut kedalam method **main()**.

```
public static void main (String [] args){
    Scanner sc = new Scanner(System.in);

    System.out.print("Masukkan nilai N : ");
    int N = sc.nextInt();

    int i = 0;
    while(i <= N){
        int j = 0;
        while(j < i){
            System.out.print("*");
            j++;
        }
        System.out.println("");
        i++;
    }
}
```

5. Compile dan jalankan program! Amati hasil running program berikut:

```
run:
Masukkan nilai N : 10

*
**
***
****
*****
******
*******
********
*********
**********
BUILD SUCCESSFUL (total time: 3 seconds)
```