JOBSHEET 7
Perulangan 2

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

2. Alat dan Bahan
   - PC/Laptop
   - JDK
   - Java IDE

3. Ulasan Teori

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

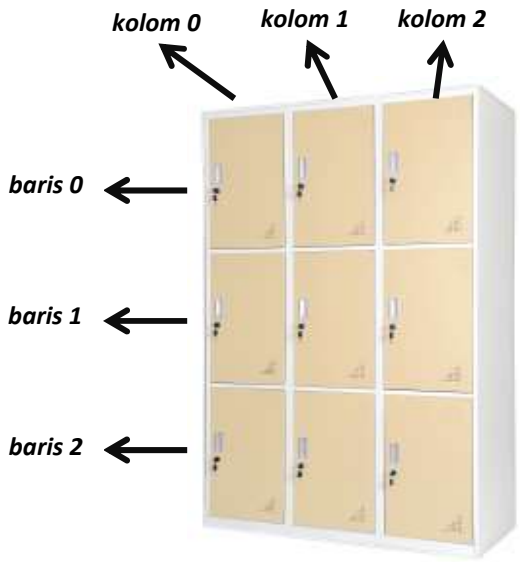
Gambar 1. Loker Penyimpanan

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.

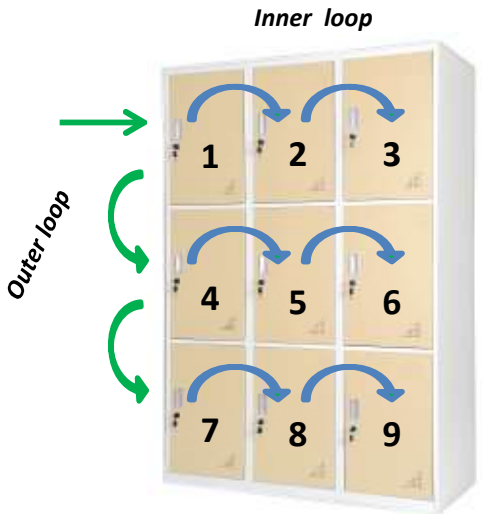
Kita mengunjungi baris pertama, dan membuka setiap pintu loker satu persatu (pintu 1, 2, dan 3). Kemudian kita mengunjungi baris selanjutnya yaitu baris kedua dan membuka pintu loker pada baris kedua tersebut satu persatu (pintu 4, 5, dan 6). Terakhir kita mengunjungi baris ketiga dan membuka pintu loker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

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***).



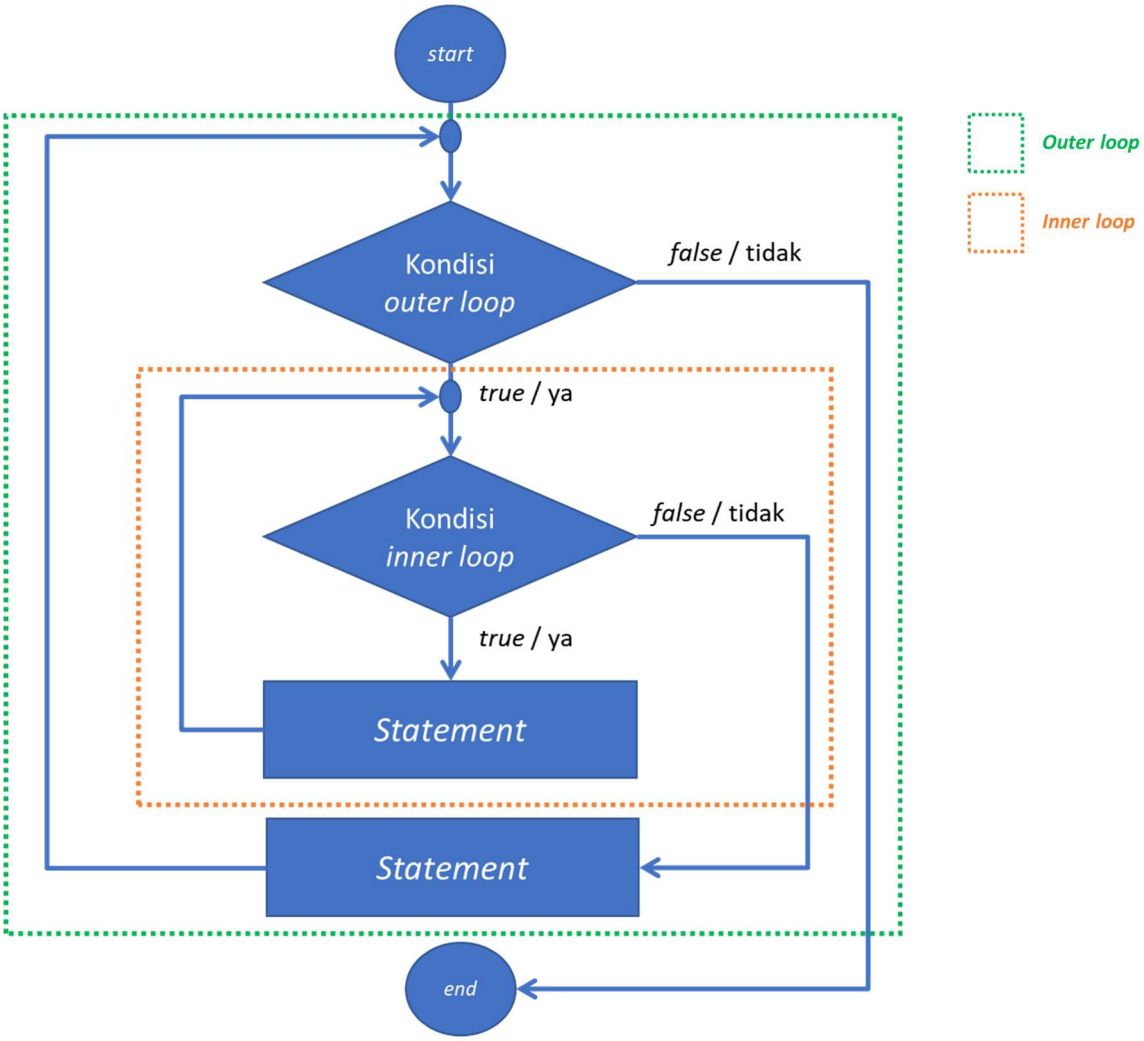
Gambar 2. Baris Kolom pada loker



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);
```

4. Praktikum

4.1 Percobaan 1: review perulangan yang lalu

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 class baru dengan nama **`Star`** dan simpan dalam file **`Star.java`**
3. Buat fungsi/method **`main()`** di dalamnya.
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;
```

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

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

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

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

7. 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

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

```
Masukkan Nilai N = 5
*****
```

Pertanyaan

1. Jika pada perulangan for, inisialisasi **i=1** diubah menjadi **i=0**, apa yang akibatnya? Mengapa bisa demikian?
2. Jika pada perulangan for, kondisi **i <= N** diubah menjadi **i > N**, apa akibatnya? Mengapa bisa demikian?
3. Jika pada perulangan for, kondisi step **i++** diubah menjadi **i--** apa akibatnya? Mengapa bisa demikian?

4.2 Percobaan 2 : Bintang Persegi

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 5, 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.)
4. Buat class Square dan simpan dengan nama file Square.java
5. Karena program membutuhkan input dari keyboard, maka perlu import class Scanner. Jadi tambahkan sintaks import di baris atas sendiri program.

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

6. Buat method main(), dan isikan kode program yang sama dengan isi method main() di percobaan 1.

```
Scanner sc = new Scanner(System.in);
System.out.print("Masukkan nilai N = ");
int N = sc.nextInt();
for(int i=1; i<=N; i++){
        System.out.print("*");
}
```

7. Compile dan jalankan program!Dan pastikan program jalan seperti saat percobaan 1.
8. Perhatikan sintaks perulangan yang digunakan untuk mencetak * sebanyak N kali ke arah samping. Di step-6 di atas kode **for** (kota merah) kita jadikan sebagai *inner loop*.
9. Kita looping lagi inner loop sebanyak N kali untuk menghasilkan output seperti tahap 1. Maka perlu ditambahkan perulangan luar (*outer loop*).

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

10. Simpan perubahan, compile dan jalankan program!
11. Amati hasilnya, maka hasilnya harusnya akan serupa dengan tampilan di bawah ini.

```
Masukkan Nilai N = 5
*****
*****
*****
*****
*****
```

Pertanyaan

1. Perhatikan perulangan luar. Jika pada sintaks **`for`**, inisialisasi `iOuter=1` diubah menjadi `iOuter=0`, apa yang akibatnya? Mengapa bisa demikian?
2. Kembalikan program semula dimana inisialisasi **`iOuter=1`**. Kemudian perhatikan perulangan dalam, Jika pada sintaks **`for`**, inisialisasi `i=1` diubah menjadi `i=0`, apa yang akibatnya? Mengapa bisa demikian?
3. Jadi, apakah perbedaan kegunaan antara perulangan luar dengan perulangan yang berada didalamnya?
4. Mengapa perlu ditambahkan sintaks `System.out.println();` di bawah perulangan dalam? Apa akibatnya jika sintaks tersebut dihilangkan?
5. Silakan commit dan push ke repository Anda.

4.3 Percobaan 3 : Bintang Segitiga

1. Pada percobaan ke-3 akan dilakukan percobaan segitiga *, dengan sama siku dengan tinggi sebesar N. Misalkan N dimasukan 5, 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. Buat method **`main()`**, dan isikan kode program berikut kedalam method **`main()`**.

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

5. Compile dan jalankan program! Amati apa yang terjadi.

Pertanyaan

1. Perhatikan, apakah output yang dihasilkan dengan nilai N = 5 sesuai dengan tampilan berikut?

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

2. Jika tidak sesuai, bagian mana saja yang harus diperbaiki/ditambahkan? Jelaskan setiap bagian yang perlu diperbaiki/ditambahkan.

4.4 Percobaan 4 : Kuis Tebak Angka

1. Buat class baru dengan nama Triangle dan simpan dengan nama file **`Quiz.java`**. **`Import class Random`** dan **`class Scanner`**, di baris awal program.

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

2. Buat fungsi **`main()`**
3. Di dalam fungsi **`main()`** deklarasikan objek dari **`class Random`** dan **`class Scanner`**. **`Class Random`**, pada kasus ini digunakan untuk mengaccak angka.

```
Random random = new Random();
Scanner input = new Scanner(System.in);
```

4. Kemudian pada baris selanjutnya, tambahkan sintaks seperti di bawah ini.

```
char menu='y';
do{
        int number = random.nextInt(10) + 1;
        boolean success = false;
        do {
                System.out.print("Tebak angka (1-10): ");
                int answer = input.nextInt();
                input.nextLine();
                success = (answer == number);
        } while(!success);
        System.out.print("Apakah Anda ingin mengulang permainan (Y/y)?");
        menu = input.nextLine().charAt(0);
} while(menu=='y' || menu=='Y');
```

Catatan: Statement input.nextLine() pada potongan kode di atas, digunakan untuk mengabaikan karakter new line

5. Compile dan jalankan program.
6. Amati jalannya alur program tersebut.

Pertanyaan

1. Jelaskan alur program di atas!
2. Apa yang harus dilakukan untuk tidak melanjutkan (tidak mengulangi) permainan tersebut?
3. Modifikasi program di atas, sehingga bisa menampilkan informasi mengenai : input nilai tebakan yang dimasukan oleh user apakah lebih kecil atau lebih besar dari jawaban/*number* yang di random!
4. Silakan commit dan push ke repository Anda.

4.5 Percobaan 5: Mengisi dan menampilkan array

1. Buatlah kelas baru dengan nama NestedLoop_NIM
2. Buatlah fungsi main()
3. Dalam fungsi main() tambahkan deklarasi untuk Scanner dan deklarasi array 2 dimenasi dengan jumlah baris 5 dan kolom 7.
4. Tambahkan kode baris seperti berikut

```
for (int i = 0; i < temps.length; i++) {
    System.out.println("Kota ke-" + i);
    for (int j = 0; j < temps[0].length; j++) {
        System.out.print("Hari ke-" + (j + 1) + ": ");
        temps[i][j] = scanner.nextDouble();
    }
    System.out.println();
}
```

5. Selanjutnya, tambahkan juga beberapa baris kode berikut

```
for (int i = 0; i < temps.length; i++) {
    System.out.print("Kota ke-" + (i + 1) + ": ");
    for (int j = 0; j < temps[0].length; j++) {
        System.out.print(temps[i][j] + " ");
    }
    System.out.println();
}
```

6. Compile dan jalankan program.
7. Amati jalannya alur program tersebut.
   Catatan: Contoh program di atas adalah contoh kasus untuk menampung data suhu pada sebuah kota, cek kembali slide pada minggu lalu.

Pertanyaan

1. Jelaskan alur program di atas!
2. Silakan modifikasi program di atas pada bagian untuk menampilkan array menggunakan foreach!
3. Modifikasi program di atas sehingga bisa menampilkan nilai rata-rata masing-masing kota!
4. Silakan commit dan push ke repository Anda.

5. Tugas individu dan kelompok

1. Buatlah program untuk mencetak tampilan segitiga angka seperti di bawah ini berdasarkan input N (nilai N minimal 3). Contoh N = 5

```
    1
   12
  123
 1234
12345
```

2. Buatlah program untuk mencetak tampilan segitiga bintang seperti di bawah ini berdasarkan input N (nilai N minimal 5). Contoh N = 7

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

3. Buatlah program untuk mencetak tampilan persegi angka seperti di bawah ini berdasarkan input N (nilai N minimal 3). Contoh N = 3, dan N = 5

```
            5 5 5 5 5
            5       5
3 3 3       5       5
3   3       5       5
3 3 3       5 5 5 5 5
```

4. Implementasikan flowchart dari fitur-fitur yang telah Anda buat pada tugas teori sebelumnya tentang nested loop!
5. Jangan lupa, semoga kode program harus di-push ke repository Anda.