

## JOB SHEET 6

### SINTAKS PERULANGAN 1

#### 1. Kompetensi

Mahasiswa memahami serta mampu membuat program dalam bahasa Java menggunakan instruksi perulangan.

#### 2. Alat Dan Bahan:

1. PC/Laptop
2. JDK
3. Text editor (Sublime)

#### 3. Ulasan Teori:

*Loop* adalah suatu blok atau kelompok instruksi yang dilaksanakan secara berulang-ulang. Perulangan yang disebut juga *repetition* akan membuat efisiensi proses dibandingkan jika dioperasikan secara manual.

Perulangan yang dijelaskan pada *jobsheet* ini adalah :

- Perulangan dengan **for**
- Perulangan dengan **while**
- Perulangan dengan **do-while**

##### 3.1 For

For adalah kode yang digunakan untuk menjalankan serangkaian kode secara berulang-ulang. Pada kode for ini terdapat beberapa komponen yang dicantumkan, antara lain: (1) inisialisasi, (2) kondisi, (3) perubahan nilai, (4) statement yang diulang. Berikut ini format sintaks untuk kode for.

```
for (inisialisasi; kondisi; perubahan_nilai) {  
    statement;  
    ...  
}
```

Berikut ini adalah contoh skrip untuk mencetak tulisan “Hello dasar pemrograman” sebanyak 10 kali.

```
for (int a = 0; a < 10; a++) {  
    System.out.println(“Hello dasar pemrograman”);  
}
```

##### 3.2 While

Kode while merupakan kode alternatif untuk melakukan perulangan selain for. Cara kerjanya sama, namun sintaks (aturan penulisan) yang berbeda. Berikut sintaks while

```
while(kondisi) {  
    statement;  
    perubahan nilai;  
}
```

Berikut ini adalah contoh skrip untuk mencetak tulisan “Hello dasar pemrograman” sebanyak 10 kali.

```
int a = 0;
while (a < 10) {
    System.out.println("Hello dasar pemrograman");
}
```

### 3.2 Do - While

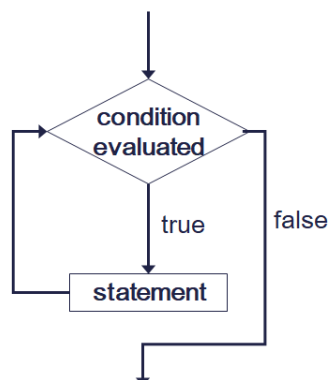
Kode do-while merupakan kode while-do dengan sintaks yang berbeda. Cara kerja dowhile relatif sama dengan while. Berikut sintaks untuk do-while.

```
do {
    statement;
    perubahan_nilai;
} while (kondisi);
```

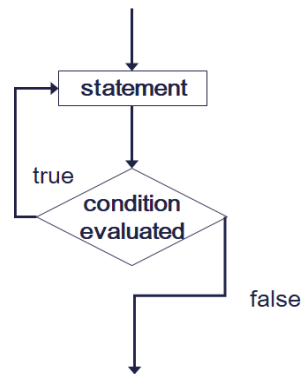
Berikut ini adalah contoh skrip untuk mencetak tulisan “Hello dasar pemrograman” sebanyak 10 kali.

```
int a = 0;
do {
    System.out.println("Hello");
    a++;
} while (a < 10);
```

Ketiga jenis *loop* tersebut sama-sama memiliki kondisi yang merupakan batasan suatu perulangan dilakukan. Cara kerja *loop* menggunakan **for** dan **while** dijelaskan pada Gambar 1. Batasan yang menjadi kondisi suatu perulangan didefinisikan dulu di awal, kemudian dilanjutkan dengan *statement* yang harus di-*looping*. Berbeda dengan **for** dan **while**, Gambar 2 menjelaskan tentang alur perulangan menggunakan **do-while**. Setelah inisialisasi dilakukan, akan di proses dulu *statement* yang harus dijalankan, baru kemudian dilakukan pembatasan *looping* dalam penulisan kondisi.



Gambar 1 Flowchart perulangan *for* dan *while*



Gambar 2 Flowchart perulangan do-while

### 3.3 Break dan Continue

Break dan continue tergolong ke dalam keyword di bahasa pemrograman java, yang keduanya digunakan pada suatu kondisi tertentu, pada perulangan seperti while, do while dan for. Jika fungsi break digunakan untuk menghentikan suatu pernyataan (statement), dan jika fungsi continue digunakan untuk mengabaikan, lalu melanjutkan suatu pernyataan pada perulangan. Keyword break dan continue juga biasa digunakan, bersamaan dengan Control Flow seperti if else, dan switch case di dalam program java

## 4. Langkah Praktikum:

1. Tulis ulang program untuk melakukan perulangan sebagai berikut :

### a) Perulangan dengan for

```
public static void main(String[] args) {  
    int angka, fac, i;  
    System.out.println ("====PROGRAM MENGHITUNG NILAI FAKTORIAL DENGAN FOR====");  
    System.out.print("Masukkan bilangan : ");  
  
    Scanner input = new Scanner (System.in);  
    angka = input.nextInt();  
    fac =1;  
    for (i=1; i<=angka;i++)  
    {  
        fac = fac*i;  
    }  
    System.out.printf("Nilai faktorial bilangan tersebut adalah : %d \n", fac);  
}
```

b) Perulangan dengan **while**

```
public static void main(String[] args) {
    Scanner input = new Scanner (System.in);
    int angka, fac, i;
    System.out.println ("====PROGRAM MENGHITUNG NILAI FAKTORIAL DENGAN WHILE====");
    System.out.print("Masukkan bilangan : ");

    angka = input.nextInt();
    fac =1;
    i = 1;
    while (i<=angka)
    {
        fac = fac*i;
        i++;
    }
    System.out.printf("Nilai faktorial bilangan tersebut adalah : %d \n", fac);
}
```

c) Perulangan dengan **do-while**

```
public static void main(String[] args) {
    Scanner input = new Scanner (System.in);
    int angka, fac, i;
    System.out.println ("====PROGRAM MENGHITUNG NILAI FAKTORIAL DENGAN DO-WHILE====");
    System.out.print("Masukkan bilangan : ");

    angka = input.nextInt();
    fac =1;
    i = 1;
    do
    {
        fac = fac*i;
        i++;
    }
    while (i<=angka);
    System.out.printf("Nilai faktorial bilangan tersebut adalah : %d \n", fac);
}
```

2. Cocokkan hasil *running* program yang sudah Anda buat apakah sudah sesuai dengan tampilan berikut ini?

```
====PROGRAM MENGHITUNG NILAI FAKTORIAL====
Masukkan bilangan : 5
Nilai faktorial bilangan tersebut adalah : 120
```

3. Salinlah program perulangan dengan menggunakan *break* berikut :

a) Perulangan dengan **for**

```
public static void main(String[] args) {
    Scanner input = new Scanner (System.in);
    int angka, b;
    System.out.println ("=====PROGRAM LOOP DENGAN BREAK=====");
    for (b=0; true;){
        System.out.print("Masukkan bilangan : ");
        angka = input.nextInt();
        b += angka;

        if (b>50) break;
    }
    System.out.printf("Angka berhenti pada jumlah angka : %d \n", b);
}
```

b) Perulangan dengan **while**

```
public static void main(String[] args) {
    Scanner input = new Scanner (System.in);
    int angka, b;
    System.out.println ("=====PROGRAM LOOP DENGAN BREAK=====");
    b=0;
    while (true){
        System.out.print("Masukkan bilangan : ");
        angka = input.nextInt();
        b += angka;
        if (b>50) break;
    }
    System.out.printf("Angka berhenti pada angka : %d \n", b);
}
```

c) Perulangan dengan **do-while**

```
public static void main(String[] args) {
    Scanner input = new Scanner (System.in);
    int angka, b;
    System.out.println ("=====PROGRAM LOOP DENGAN BREAK=====");
    b=0;
    do
    {
        System.out.print("Masukkan bilangan : ");
        angka = input.nextInt();
        b += angka;
        if (b>50) break;
    }
    while (true);
    System.out.printf("Angka berhenti pada angka : %d \n", b);
}
```

4. Cocokkan hasil *running* program *looping* menggunakan *break* yang sudah Anda buat apakah sudah sesuai dengan tampilan berikut ini?

=====PROGRAM LOOP DENGAN BREAK=====

Masukkan bilangan : 2  
 Masukkan bilangan : 12  
 Masukkan bilangan : 22  
 Masukkan bilangan : 32  
 Angka berhenti pada angka : 68

5. Salinlah program perulangan dengan menggunakan *continue* berikut :

```
public static void main(String[] args) {
    Scanner input = new Scanner (System.in);
    int angka, b, i, count;
    double avg, total;
    System.out.println ("=====PROGRAM LOOP DENGAN CONTINUE=====");
    b=0;
    count=0;

    for (i=0;i<5;i++){
        System.out.print("Masukkan bilangan : ");
        angka = input.nextInt();
        if (angka>=50) continue;
        b += angka;
        count ++;
    }
    total =(double)b;
    System.out.printf("Jumlah angka kurang dari 50: %.2f \n", total);
    avg = (double)b/count;
    System.out.printf("Rata-rata angka kurang dari 50: %.2f \n", avg);
}
```

6. Cocokkan hasil *running* program *looping* menggunakan *continue* yang sudah Anda buat apakah sudah sesuai dengan tampilan berikut ini?

=====PROGRAM LOOP DENGAN CONTINUE=====

Masukkan bilangan : 30  
 Masukkan bilangan : 40  
 Masukkan bilangan : 50  
 Masukkan bilangan : 60  
 Masukkan bilangan : 20  
 Jumlah angka kurang dari 50: 90.00  
 Rata-rata angka kurang dari 50: 30.00

## 5. Pertanyaan

- Misalkan, Anda diminta membuat sebuah program Java yang meminta masukan sebuah bilangan bulat **n**. Kemudian, program menampilkan karakter **\*** di layar sebanyak **n kali**. Manakah di antara kedua potongan program di bawah ini yang lebih baik dan aman ? mengapa ?

|   |   |
|---|---|
| <pre>/* misal: masukan user n sudah ditampung di variabel integer n */ int i = 0;  while (i &lt; n) {     System.out.print("*");     i++; }</pre> | <pre>/* misal: masukan user n sudah ditampung di variabel integer n */ int i = 0;  while (i != n) {     System.out.print("*");     i++; }</pre> |
|---|---|

- Apakah *output* dari ketiga potongan program di bawah ini:

|  |  |   |
|--|--|---|
| <pre>int r = 1; int i = 1; int a = 2; int n = 4;  while (i &lt;= n) {     r = r * a;     i++; } System.out.print(r);</pre> | <pre>int n = 5; boolean stop = false;  int i = 1; while (!stop) {     if (i &gt;= n) {         stop = true;     } else {         if (i % 2 == 1) {             System.out.print("#");         } else {             System.out.print("*");         }         i++;     } }</pre> | <pre>int n = 5; long hasil = 1; for (int i = 1; i &lt;= n; i++){     hasil = hasil * i; }  System.out.println(n+"!= "+hasil);</pre> |
|--|--|---|

- Salinlah program perulangan berikut, dan cocokkan hasil running program apakah sesuai dengan hasil di bawah ini

|  |   |
|--|---|
| <pre>public static void main(String[] args) {     int a=5;     for (int b=1; b&lt;=a; b++){         for(int c=1; c&lt;=b; c++){             System.out.print("*");         }         System.out.println();     } }</pre> | <pre>★ ★★ ★★★ ★★★★ ★★★★★ BUILD SUCCESSFUL</pre> |
|--|---|

- a. Kembangkan program tersebut sehingga mampu untuk menampilkan output sebagai berikut

```

      *
     **
    ***
   ****
  *****
BUILD SUCCESSFUL
    
```

## 6. Tugas

1. **(SumAvgGenap)** Buatlah program dengan menggunakan bahasa Java yang meminta masukan *user* sebuah bilangan bulat **N** ( $N > 0$ ). Program kemudian menampilkan penjumlahan N bilangan genap positif pertama (**bilangan genap  $\geq 0$** ).

Contoh:

- Jika *user* memasukkan  $N = 10$ , program akan menghitung banyaknya jumlah bilangan positive di dalam range bilangan 1-10 kemudian menampilkan penjumlahan bilangan positive bilangan bilangan diantara 1-10 yaitu :  
 $0 + 2 + 4 + 6 + 10 = 30$ .  
 Setelah itu program akan menampilkan rata-rata dari bilangan positive yang telah dijumlahkan tadi.
- Contoh output program

Menghitung Jumlah Bilangan Genap dari N Bilangan

```

Masukan range bilangan = 10
Banyaknya Bilangan genap dari 1 sampai 10 adalah 5
Bilangan genap Ke-1 adalah 2
Bilangan genap Ke-2 adalah 4
Bilangan genap Ke-3 adalah 6
Bilangan genap Ke-4 adalah 8
Bilangan genap Ke-5 adalah 10
Jumlah bilangan genap dari 1 sampai 10 = 30
Rata-rata bilangan genap dari 1 sampai 10 = 6.0
BUILD SUCCESSFUL (total time: 2 seconds)
    
```

Silakan Anda rancang sendiri untuk tampilan programnya

2. **(SumKGanjil)** Buatlah program dengan menggunakan bahasa Java yang meminta masukan *user* sebuah bilangan bulat **N** ( $N > 0$ ). Program kemudian menampilkan penjumlahan N bilangan ganjil positif pertama (**bilangan genap  $\geq 0$** ).

Contoh:

- Jika *user* memasukkan  $N = 5$ , program akan menghitung banyaknya jumlah bilangan positive di dalam range bilangan 1-5 kemudian menampilkan penjumlahan kuadrat bilangan ganjil diantara 1-5 yaitu :  
 $1^2 + 3^2 + 5^2 = 35$ .  
 Setelah itu program akan menampilkan rata-rata dari jumlah kuadrat bilangan negatif tersebut.
- Contoh output program

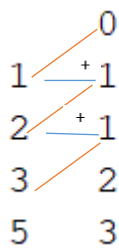


Menghitung Jumlah Kuadrat Bilangan Ganjil dari N Bilangan

---

```
Masukan range bilangan = 5
Banyaknya Bilangan Ganjil dari 1 sampai 5 adalah 3
Bilangan Ganjil Ke-1 adalah 1
Bilangan Ganjil Ke-2 adalah 3
Bilangan Ganjil Ke-3 adalah 5
Jumlah bilangan Ganjil dari 1 sampai 5 = 35.0
Rata-rata bilangan Ganjil dari 1 sampai 5 = 11.67
BUILD SUCCESSFUL (total time: 4 seconds)
```

3. **(Fibonacci)** Buatlah sebuah program yang menampilkan deret bilangan sebagai berikut



0  
1 + 1  
2 + 1  
3 + 2  
5 + 3

Dimana bilangan yang terletak di sebelah kiri adalah hasil penambahan dari bilangan di atasnya sebagai contoh

0+1 = 1  
1+1 = 2  
2+1 = 3