# JOBSHEET 6

# SINTAKS PERULANGAN 1

## Kompetensi

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

## Alat Dan Bahan:

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

## 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 berulangulang. 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;

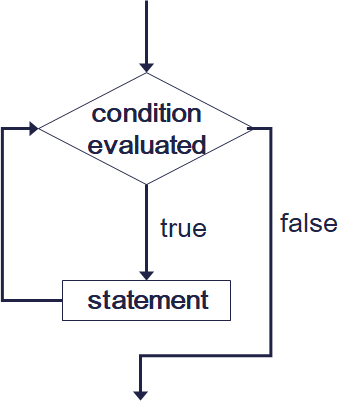
do {

System.out.println(“Hello”);

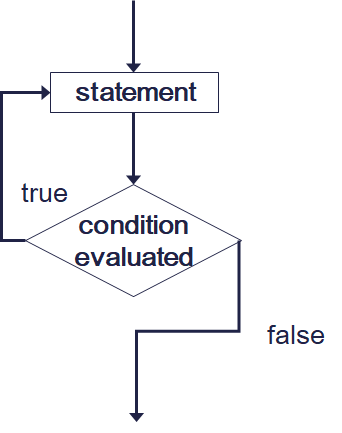
a++;

} 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

## Langkah Praktikum:

1. Tulis ulang program untuk melakukan perulangan sebagai berikut :
2. Perulangan dengan **for**

|  |
| --- |
|  |

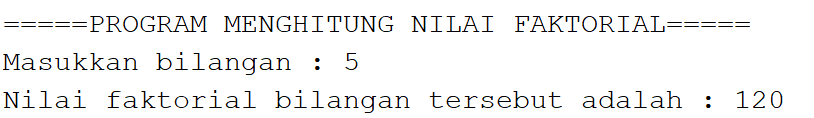
1. Perulangan dengan **while**

|  |
| --- |
|  |

1. Perulangan dengan **do-while**

|  |
| --- |
|  |

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



1. Salinlah program perulangan dengan menggunakan *break* berikut :
2. Perulangan dengan **for**

|  |
| --- |
|  |

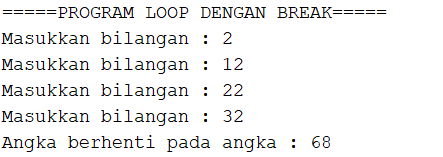
1. Perulangan dengan **while**

|  |
| --- |
|  |

1. Perulangan dengan **do-while**

|  |
| --- |
|  |

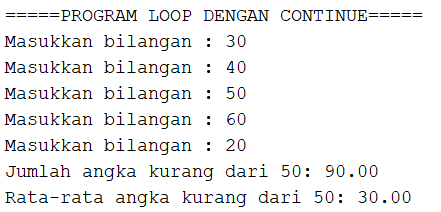
1. Cocokkan hasil *running* program *looping* menggunakan *break* yang sudah Anda buat apakah sudah sesuai dengan tampilan berikut ini?



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

|  |
| --- |
|  |

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



## Pertanyaan

1. 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 ?

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

1. Apakah *output* dari ketiga potongan program di bawah ini:

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

## 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 ≥ 0)**.

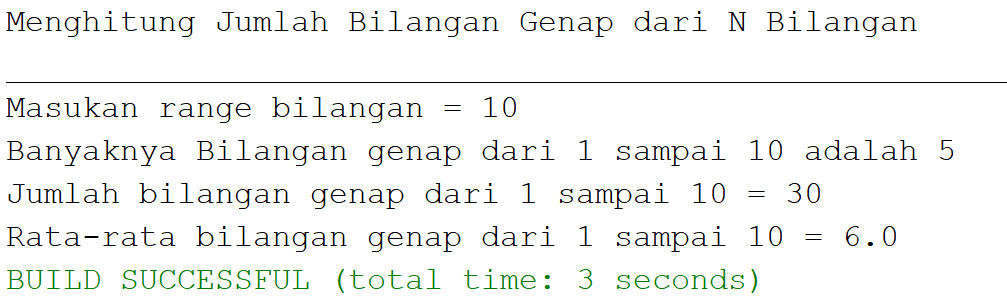
Contoh:

* 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



Silakan Anda rancang sendiri untuk tampilan programnya

1. **(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 ≥ 0)**.

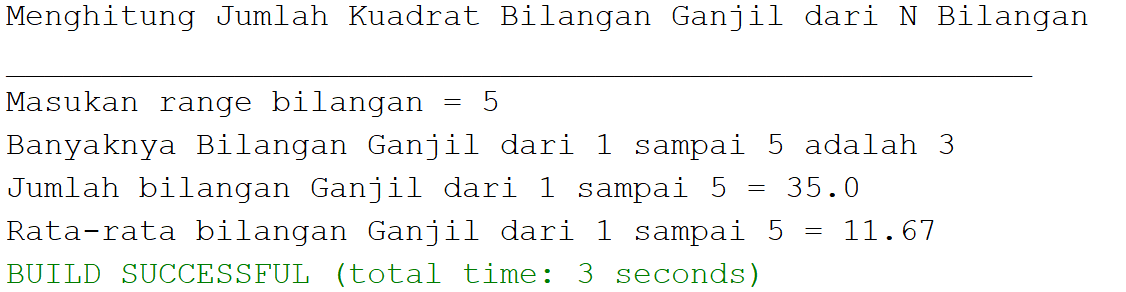
Contoh:

* 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 :

12 + 32 + 52 = 35.

Setelah itu program akan menampilkan rata-rata dari jumlah kuadrat bilangan negatif tersebut.

* Contoh output program



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

0 1 1 2 3 5 8

Dimana rangkaian tersebut diperoleh dengan dimulai dari menjumlahkan angka 0+1=1, 1+1=2, 1+2=3 kemudian menjumlah 2+3=5, 5+8=13, dan seterusnya

