



POLITEKNIK MASAMY INTERNASIONAL

SK Menristekdikti RI Nomor: 731/KPT/I/2018

Jalan Ikan Paus No.10-15 Kertosari Banyuwangi - 68411
Telp (0333) 3384593 – <http://polmain.info>

PROGRAM STUDI D3 TEKNIK KOMPUTER

Form:
B.Ak/eva/04/20

Nama Dosen : *Arif Fahmi, S.T.,M.T.*

Mata Kuliah : Pemrograman Berorientasi Object

Semester : 4 (Empat)

Kode Mata Kuliah : TKV4044

Th. Akdm : 2019/2020.

BAB

STRUKTUR KONTROL PEMROGRAMAN JAVA

SUB BAB :

1. Struktur pemilihan (if, else-if, dan switch-case)
2. Struktur pengulangan (while, do-while, for)
3. Struktur percabangan (break, continue, return)

TUJUAN MATERI

Pada bagian ini, kita akan mempelajari tentang struktur kontrol dimana kita dapat mengubah cara eksekusi pada pernyataan yang dibuat di program kita.

REFERENSI

1. J.Eck, D. (2006). *Introduction to Programming Using Java*. Geneva.
2. Ady Wicaksono, Dasar – Dasar Pemrograman Java 2, Penerbit PT Elex Media Komputindo, Jakarta, 2002

STRUKTUR KONTROL PEMROGRAMAN JAVA

Pada dasarnya struktur kontrol digunakan untuk mengatur jalannya alur program sesuai dengan yang kita inginkan. Struktur-struktur ini dikategorikan kedalam tiga jenis, yaitu: pemilihan, pengulangan, dan statemen percabangan (jum statement). Pemilihan digunakan untuk menentukan pernyataan mana yang akan dieksekusi tergantung dari ekspresi atau kondisi yang didefinisikan. Pengulangan digunakan untuk melakukan eksekusi terhadap pernyataan secara berulang sesuai dengan kondisi yang ditentukan. Adapun struktur percabangan digunakan untuk memindahkan proses eksekusi ke bagian kode program yang kita inginkan.

1. Struktur Pemilihan

Pemrograman Java menyediakan duabuaah statemen untuk proses pemilihan, yaitu “ if ” dan “ switch ”. kedua buah statemen tersebut digunakan untuk mengntrol eksekusistatemen tergantung pada kondisi yang ditentukan sebelumnya.

Kalau kita perhatikan, alur pengekseskusion sebuah kode program dikerjakan satu per satu dari atas sampai ke bawah. Baris demi baris dibaca, kemudian komputer mengerjakan apa yang diperintahkan. Misalnya seperti ini:



Pada gambar diatas alur programnya satu, tidak ada belokan atau percabangan. Percabangan hanyalah sebuah istilah yang digunakan untuk menyebut alur program yang bercabang. Percabangan juga dikenal dengan “Control Flow”, “Struktur Kondisi”, “Struktur IF”, “Decision”, dsb. Semuanya itu sama. Pada diagram alur (*Flow Chart*) seperti di atas, alurnya memang satu. Tapi setelah kita menggunakan percabangan, alurnya akan bertambah menjadi seperti gambar berikut ini.



Lalu bagaimana cara menulis kode percabangan dalam Java?

Caranya: menggunakan kata kunci if, else, switch, dan case, dan operator ternary.

1.1 Percabangan “ IF “

Pernyataan *if* akan menentukan sebuah pernyataan (atau blok kode) yang akan eksekusi jika dan hanya jika persyaratan bernilai benar(*true*).

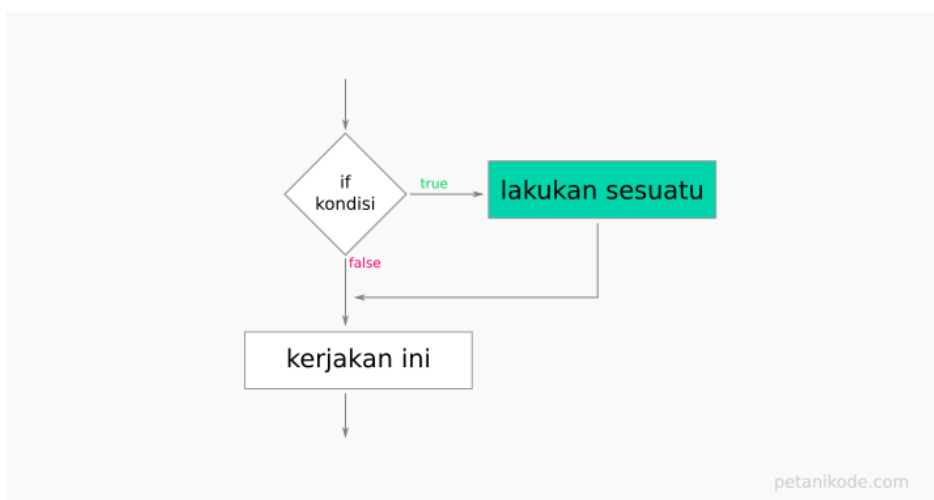
Bentuk dari pernyataan *if*,

```
If (boolean_expression )
    statement;

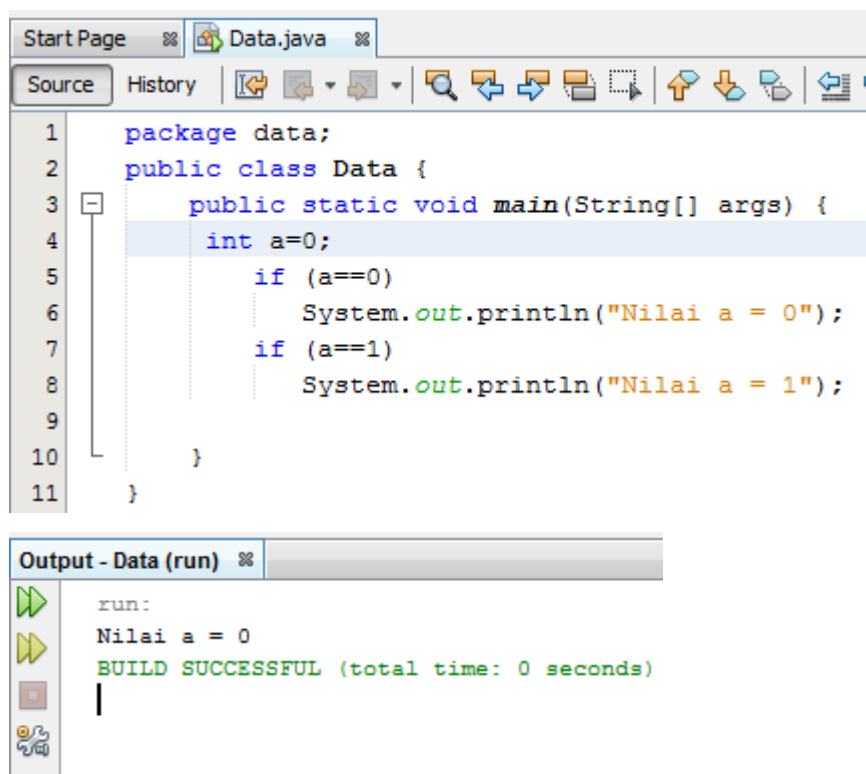
atau

if( boolean_expression ){
    statement1;
    statement2;
    . . .
}
```

dimana, *boolean_expression* adalah sebuah pernyataan logika (*true/false*) atau variabel bertipe *boolean*.



Contoh listing program “If” menggunakan software Netbeans



```
1 package data;
2 public class Data {
3     public static void main(String[] args) {
4         int a=0;
5         if (a==0)
6             System.out.println("Nilai a = 0");
7         if (a==1)
8             System.out.println("Nilai a = 1");
9     }
10 }
11 }
```

Output - Data (run)

```
run:
Nilai a = 0
BUILD SUCCESSFUL (total time: 0 seconds)
```



Java Programming

```
package data;

public class Data {

    public static void main(String[] args) {

        int a=0;

        if (a==0)

            System.out.println("Nilai a = 0");

        if (a==1)

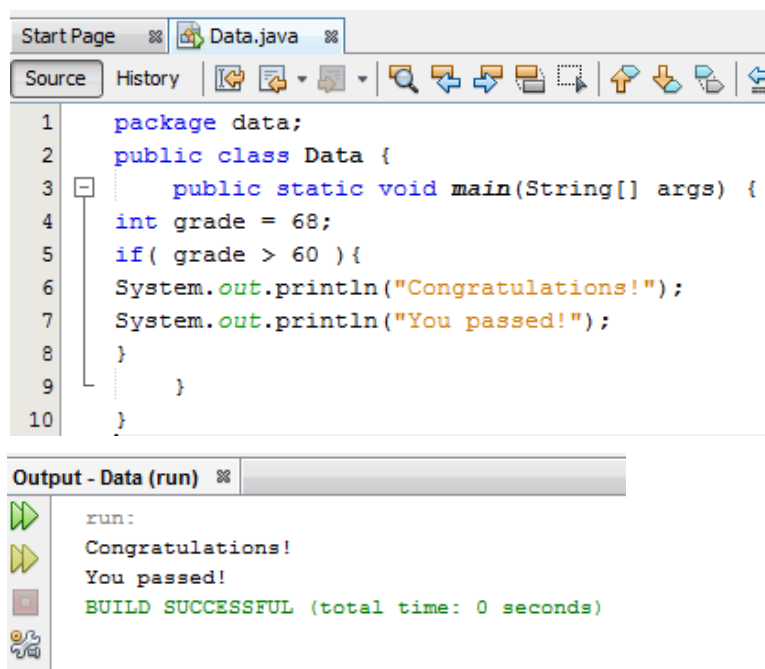
            System.out.println("Nilai a = 1");

    }

}
```

Output : Program akan menampilkan nilai a = 0 saja, karena pada if yang kedua, kondisi tidak memenuhi atau salah.

Contoh lain;



```
package data;

public class Data {

    public static void main(String[] args) {

        int grade = 68;

        if( grade > 60 ){

            System.out.println("Congratulations!");

            System.out.println("You passed!");

        }

    }

}
```

Catatan ;

Boolean_expression pada pernyataan if harus merupakan nilai boolean).Hal ini berarti persyaratan harus bernilai **true** atau **false**.

Masukkan statement di dalam blok if. Contohnya,

```
if( boolean_expression ){
    //statement1;
    //statement2;
}
```

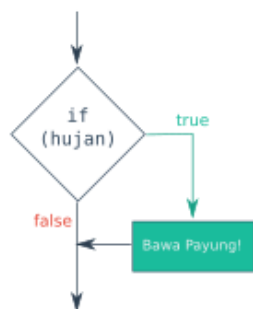
1.2 Percabangan “ IF/ELSE”

Percabangan if else digunakan saat kita memiliki **dua pernyataan** dengan syarat tertentu.

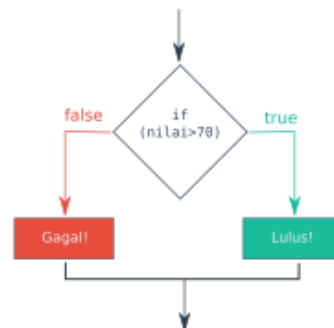
Sintaks if-else seperti berikut :

```
if(kondisi) {  
    pernyataan1  
}else{  
    pernyataan2  
}
```

Jika hasil dari if benar, maka pernyataan1 yang dijalankan, sedangkan jika salah, pernyataan dua yang akan dijalankan.

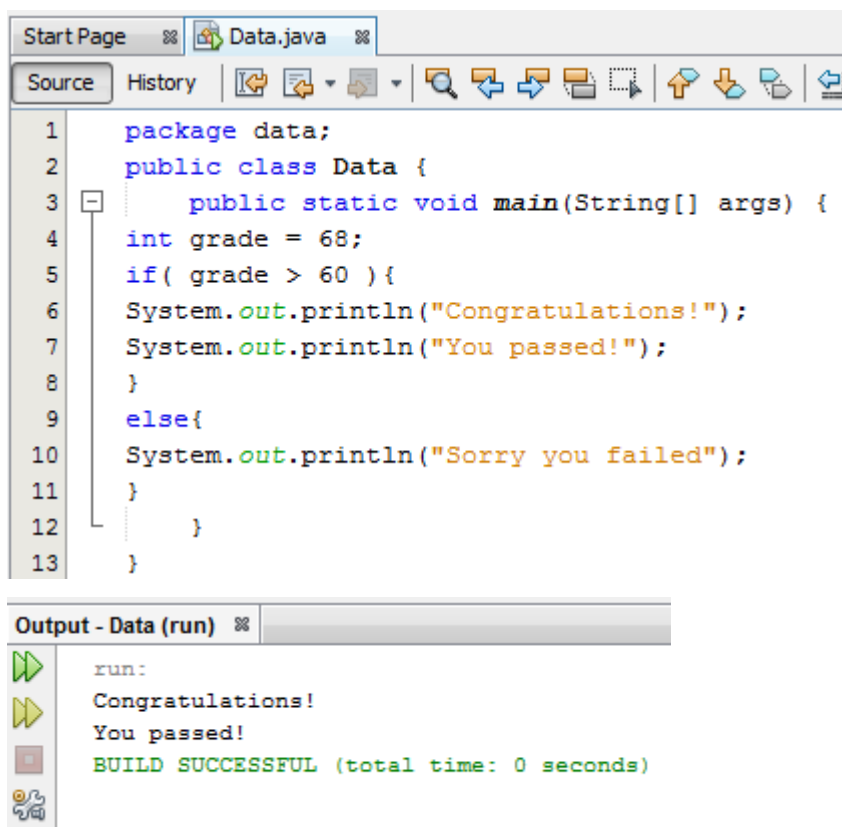


```
if (hujan) {  
    System.out.println("Bawa Payung!");  
}
```



```
if (nilai > 70) {  
    System.out.println("Lulus!");  
} else {  
    System.out.println("Gagal!");  
}
```

Contoh listing program “ If / else ” menggunakan software Netbeans



```
1 package data;  
2 public class Data {  
3     public static void main(String[] args) {  
4         int grade = 68;  
5         if( grade > 60 ){  
6             System.out.println("Congratulations!");  
7             System.out.println("You passed!");  
8         }  
9         else{  
10            System.out.println("Sorry you failed");  
11        }  
12    }  
13 }
```

Output - Data (run) ✖

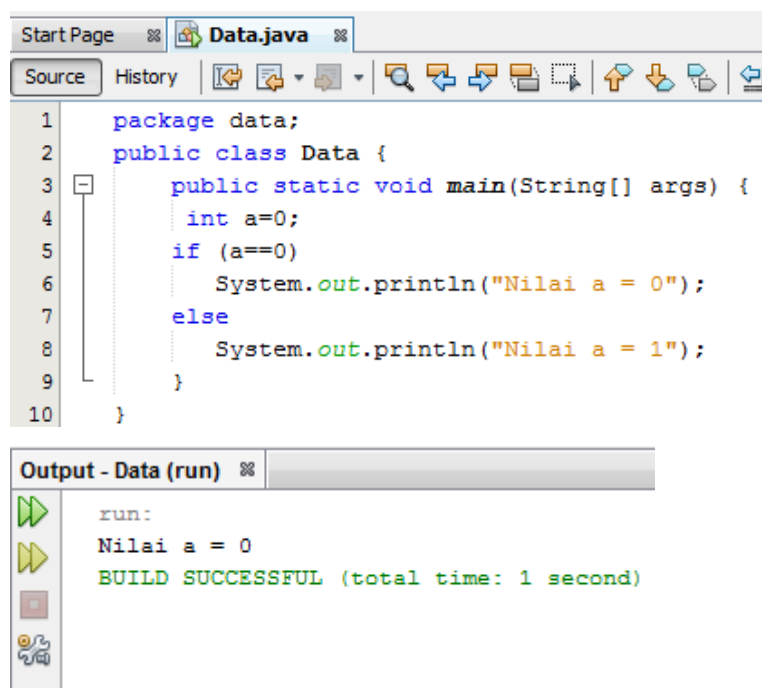
```
run:  
Congratulations!  
You passed!  
BUILD SUCCESSFUL (total time: 0 seconds)
```

```
package data;

public class Data {

    public static void main(String[] args) {
        int grade = 68;
        if( grade > 60 ){
            System.out.println("Congratulations!");
            System.out.println("You passed!");
        }
        else{
            System.out.println("Sorry you failed");
        }
    }
}
```

Contoh lain ;



The screenshot shows an IDE window titled 'Data.java' with the following code:

```
1 package data;
2 public class Data {
3     public static void main(String[] args) {
4         int a=0;
5         if (a==0)
6             System.out.println("Nilai a = 0");
7         else
8             System.out.println("Nilai a = 1");
9     }
10 }
```

Below the code editor is the 'Output - Data (run)' window, which displays the following output:

```
run:
Nilai a = 0
BUILD SUCCESSFUL (total time: 1 second)
```

Output : program akan menampilkan *nilai a = 0*, karena kondisi if bernilai benar, jika pada inisialisasi nilai a tidak bernilai 0, maka program akan menampilkan *nilai a = 1*.

Catatan ;

1. Untuk menghindari kebingungan, selalu letakkan sebuah pernyataan atau beberapa pernyataan di dalam blok if-else didalam tanda kurawal {},
2. Anda dapat memiliki blok if-else yang bersarang. Ini berarti anda dapat memiliki blok if-else yang lain di dalam blok if-else. Contohnya,

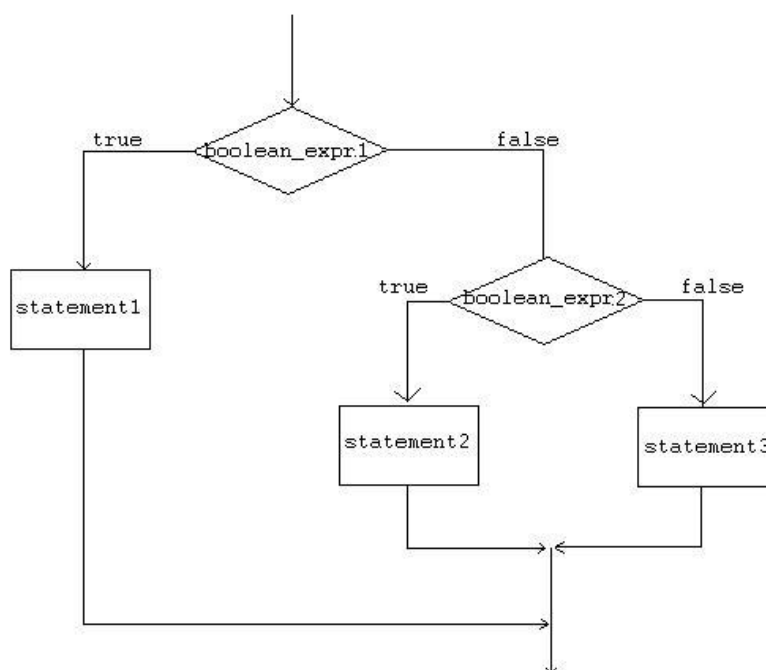
```
if( boolean_expression ){
    if( boolean_expression ){
        ...
    }
}
else{
    ...
}
```

1.3 Percabangan “ Else - IF ”

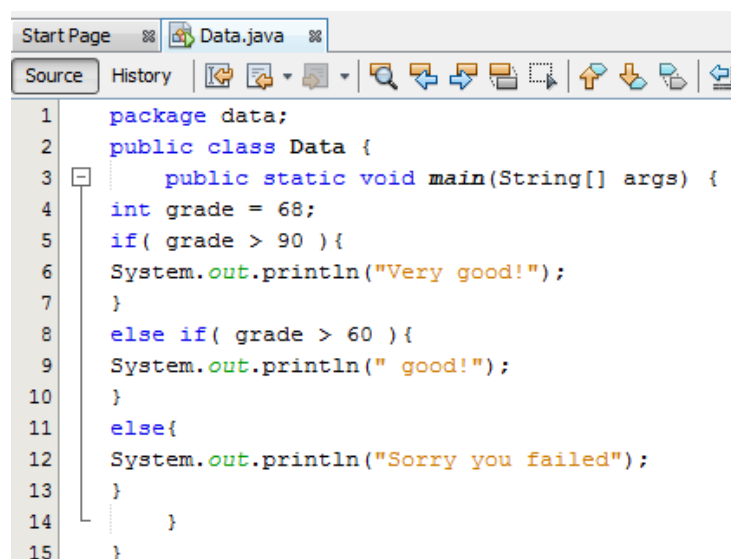
Percabangan yang digunakan saat kita memiliki banyak kondisi (lebih dari 2) dan banyak pernyataan (lebih dari 2). Sintaks dari else-if seperti berikut :

```
if(boolean_expression1 )
    statement1;
else if( boolean_expression2 )
    statement2;
else
    statement3;
```

Sebagai catatan : anda dapat memiliki banyak blok else-if sesudah pernyataan *if*. Blok *else* bersifat opsional dan dapat dihilangkan. Pada contoh yang ditampilkan di atas, jika *boolean_expression1* bernilai *true*, maka program akan mengeksekusi *statement1* dan melewati pernyataan yang lain. Jika *boolean_expression2* bernilai *true*, maka program akan mengeksekusi *statement2* dan melewati *statement3*.



Contoh listing program “ else -if ” menggunakan software Netbeans



```
1 package data;
2 public class Data {
3     public static void main(String[] args) {
4         int grade = 68;
5         if( grade > 90 ){
6             System.out.println("Very good!");
7         }
8         else if( grade > 60 ){
9             System.out.println(" good!");
10        }
11        else{
12            System.out.println("Sorry you failed");
13        }
14    }
15 }
```



Java Programming

```
Output - Data (run) %
run:
good!
BUILD SUCCESSFUL (total time: 0 seconds)
```

```
package data;
public class Data {
    public static void main(String[] args) {
        int grade = 68;
        if( grade > 90 ){
            System.out.println("Very good!");
        }
        else if( grade > 60 ){
            System.out.println(" good!");
        }
        else{
            System.out.println("Sorry you failed");
        }
    }
}
```

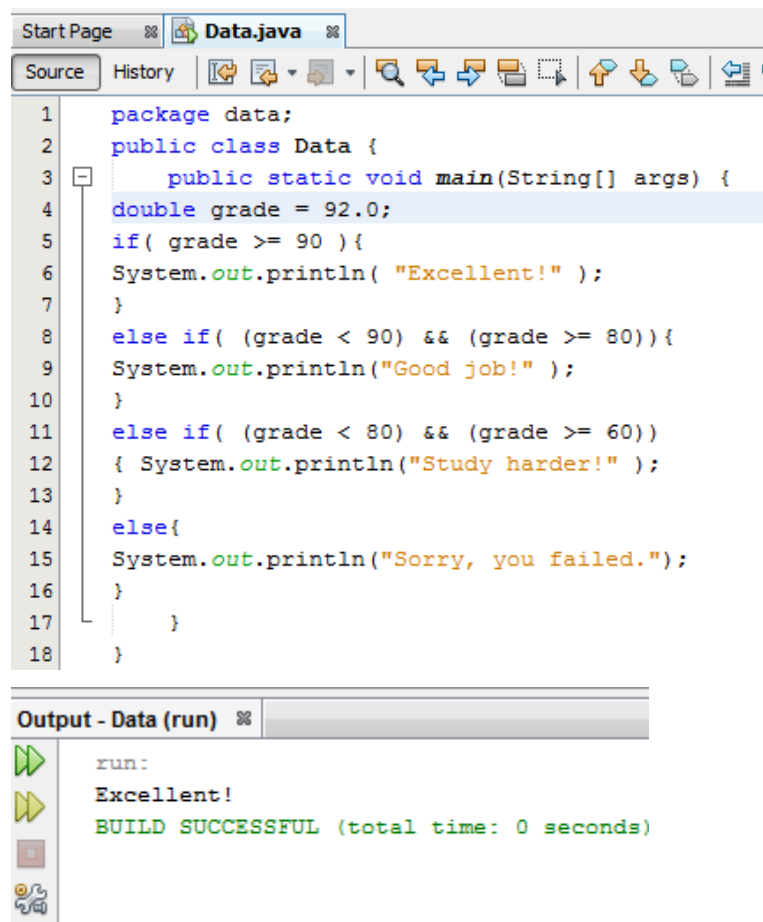
Contoh lain ;

```
Start Page % Data.java %
Source History
1 package data;
2 public class Data {
3     public static void main(String[] args) {
4         int a=2;
5         if (a==0)
6             System.out.println("Nilai a = 0");
7         else if(a==1)
8             System.out.println("Nilai a = 1");
9         else if(a==2)
10            System.out.println("Nilai a = 2");
11     }
12 }
```

```
Output - Data (run) %
run:
Nilai a = 2
BUILD SUCCESSFUL (total time: 0 seconds)
```

```
package data;
public class Data {
    public static void main(String[] args) {
        int a=2;
        if (a==0)
            System.out.println("Nilai a = 0");
        else if(a==1)
            System.out.println("Nilai a = 1");
        else if(a==2)
            System.out.println("Nilai a = 2");
    }
}
```


Contoh lain;



```
1 package data;
2 public class Data {
3     public static void main(String[] args) {
4         double grade = 92.0;
5         if( grade >= 90 ){
6             System.out.println( "Excellent!" );
7         }
8         else if( (grade < 90) && (grade >= 80)){
9             System.out.println("Good job!" );
10        }
11        else if( (grade < 80) && (grade >= 60))
12        { System.out.println("Study harder!" );
13        }
14        else{
15            System.out.println("Sorry, you failed.");
16        }
17    }
18 }
```

Output - Data (run)

```
run:
Excellent!
BUILD SUCCESSFUL (total time: 0 seconds)
```

Catatan ;

1. Kondisi pada statement *if* tidak mengevaluasi nilai logika *boolean*. Contohnya :

```
//SALAH
int number = 0;
if( number ){
    //some statements here
}
```

Variabel *number* tidak memiliki nilai Boolean.

2. Menggunakan operator `=` sebagai operator perbandingan yang seharusnya adalah operator `==` .
Contohnya,

```
//SALAH
int number = 0;
if( number = 0 ){
    //Beberapa pernyataan
}
```

Seharusnya kode tersebut ditulis,

```
//BENAR
int number = 0;
if( number == 0 ){
    //beberapa pernyataan
}
```

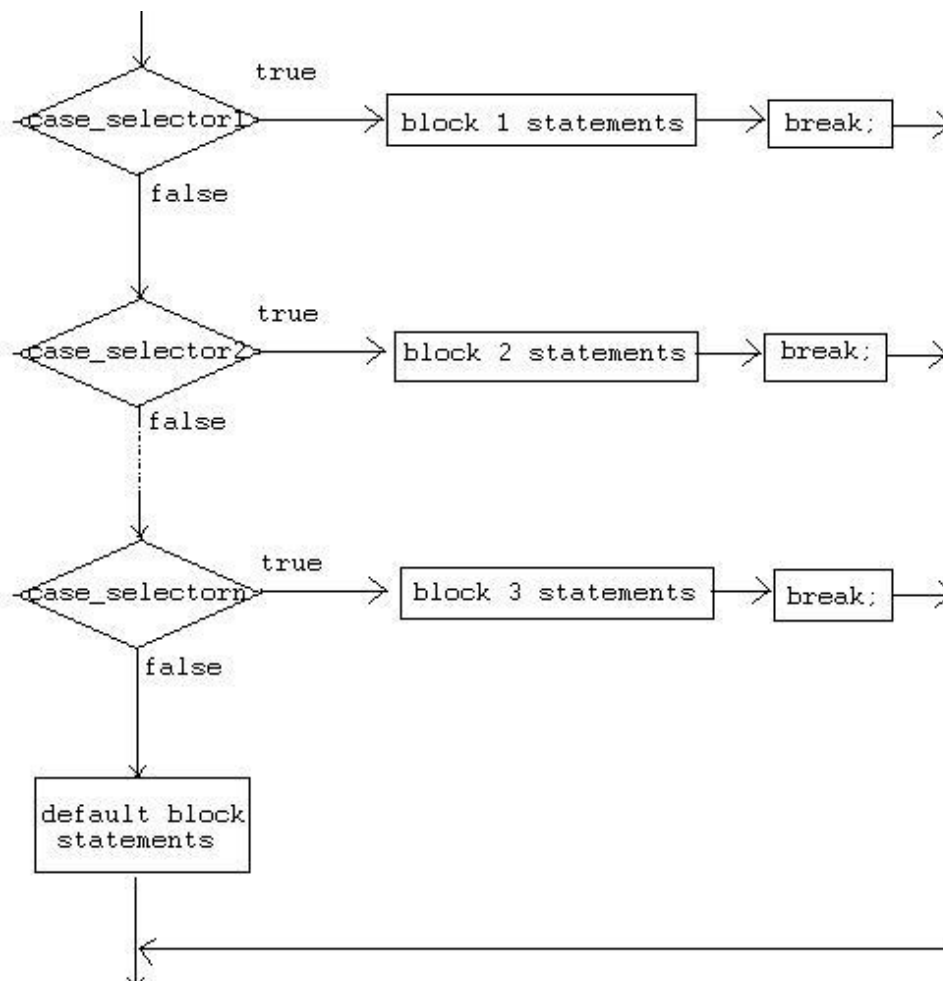
3. Penulisan **elseif** yang seharusnya ditulis sebagai **else if**.

1.4 Switch-case

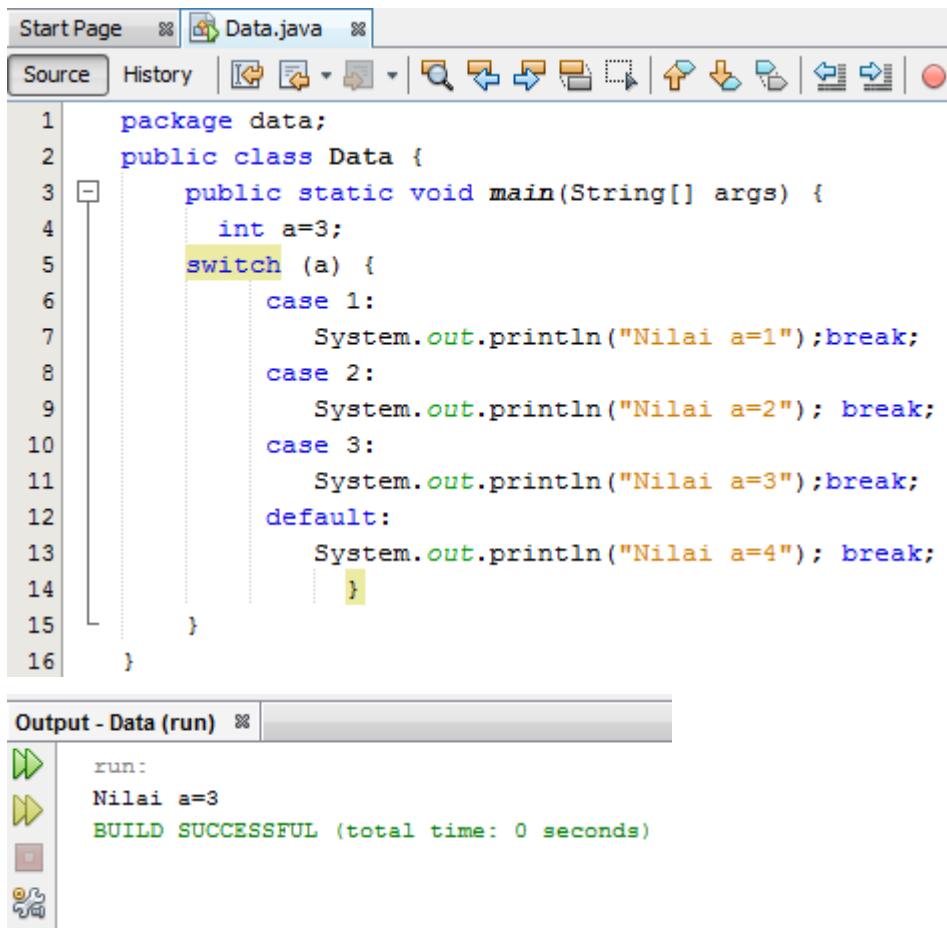
Cara lain untuk membuat cabang adalah dengan menggunakan kata kunci **switch**. Switch mengkonstruksikan cabang untuk beberapa kondisi dari nilai.

Bentuk statement switch,

```
switch( switch_expression ){  
    case case_selector1:  
        statement1; //  
        statement2; //block 1  
        ... //  
        break;  
  
    case case_selector2:  
        statement1; //  
        statement2; //block 2  
        ... //  
        break;  
    ...  
    default:  
        statement1; //  
        statement2; //block n  
        ... //  
        break;  
}
```



Contoh listing program “ Switch-case ” menggunakan software Netbeans



The screenshot shows the NetBeans IDE with a file named 'Data.java'. The code is as follows:

```
1 package data;
2 public class Data {
3     public static void main(String[] args) {
4         int a=3;
5         switch (a) {
6             case 1:
7                 System.out.println("Nilai a=1");break;
8             case 2:
9                 System.out.println("Nilai a=2"); break;
10            case 3:
11                System.out.println("Nilai a=3");break;
12            default:
13                System.out.println("Nilai a=4"); break;
14        }
15    }
16 }
```

The output window shows the following text:

```
run:
Nilai a=3
BUILD SUCCESSFUL (total time: 0 seconds)
```

```
package data;
public class Data {
    public static void main(String[] args) {
        int a=3;
        switch (a) {
            case 1:
                System.out.println("Nilai a=1");break;
            case 2:
                System.out.println("Nilai a=2"); break;
            case 3:
                System.out.println("Nilai a=3");break;
            default:
                System.out.println("Nilai a=4"); break;
        }
    }
}
```

Output : Program akan menampilkan *nilai a = 2* saja, karena kondisi bernilai benar, perhatikan break dibelakang pernyataan, jika break ini dihapus, maka semua pernyataan akan dijalankan. Break digunakan untuk keluar dari switch-case saat 1 pernyataan sudah dijalankan.



Contoh lain;

The screenshot shows an IDE window with a tab for 'Data.java'. The code is as follows:

```
1 package data;
2 public class Data {
3     public static void main(String[] args) {
4         int grade = 95;
5         switch(grade){
6             case 100:
7                 System.out.println( "Excellent!" );
8                 break;
9             case 90:
10                System.out.println("Good job!" );
11                break;
12            case 80:
13                System.out.println("Study harder!" );
14                break;
15            default:
16                System.out.println("Sorry, you failed.");
17            }
18        }
19    }
```

Below the code editor is the 'Output - Data (run)' window, which displays the following output:

```
run:
Sorry, you failed.
BUILD SUCCESSFUL (total time: 0 seconds)
```

```
package data;

public class Data {

    public static void main(String[] args) {

        int grade = 95;
        switch(grade){
            case 100:
                System.out.println( "Excellent!" );
                break;
            case 90:
                System.out.println("Good job!" );
                break;
            case 80:
                System.out.println("Study harder!" );
                break;
            default:
                System.out.println("Sorry, you failed.");
            }

        }

    }
```

2. Struktur Pengulangan

Struktur kontrol pengulangan adalah berupa pernyataan dari Java yang memungkinkan kita untuk mengeksekusi blok code berulang-ulang sesuai dengan jumlah tertentu yang diinginkan. Ada tiga macam jenis dari struktur kontrol pengulangan yaitu *while*, *do-while*, dan *for-loops*.

2.1 while loop

Pernyataan *while loop* adalah pernyataan atau blok pernyataan yang diulang-ulang sampai mencapai kondisi yang cocok.

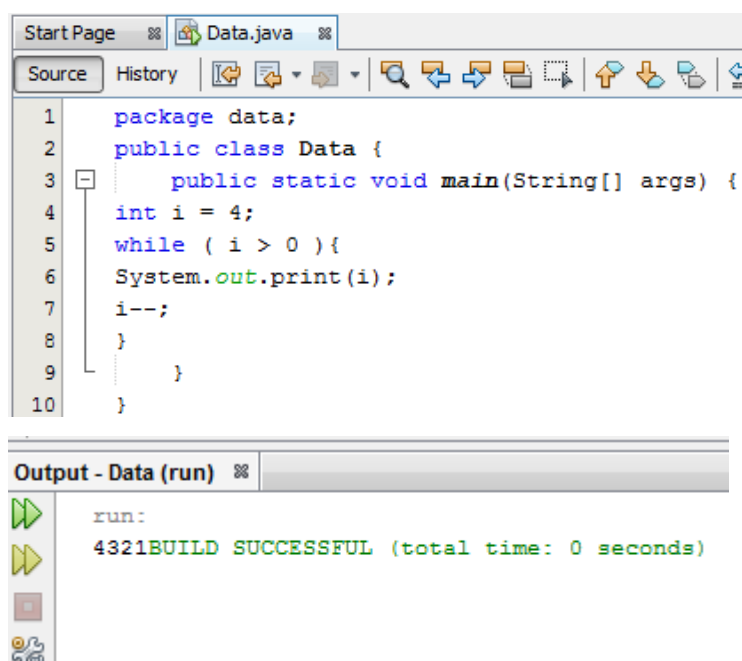
Bentuk pernyataan while,

```
while( boolean_expression ){  
    statement1;  
    statement2;  
    . . .  
}
```

Pernyataan di dalam *while loop* akan dieksekusi berulang-ulang selama kondisi *boolean_expression* bernilai benar (*true*).

Contoh, pada kode dibawah ini,

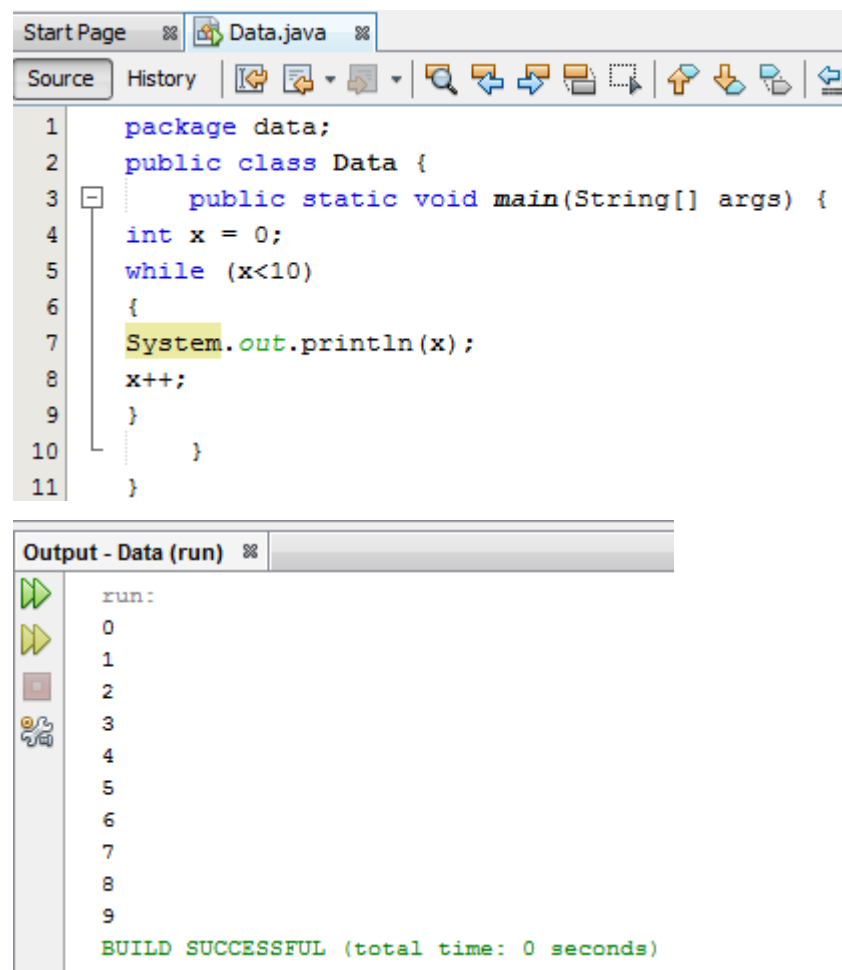
```
int i = 4;  
while ( i > 0 ){  
    System.out.print(i);  
    i--;  
}
```



Contoh diatas akan mencetak angka 4321 pada layar. Perlu dicatat jika bagian *i--*; dihilangkan, akan menghasilkan pengulangan yang terus menerus (**infinite loop**). Sehingga, ketika menggunakan *while loop* atau bentuk pengulangan yang lain, pastikan Anda memberikan pernyataan yang membuat pengulangan berhenti pada suatu kondisi.

Contoh program while loop menggunakan software netbeans

Contoh 1



```
1 package data;
2 public class Data {
3     public static void main(String[] args) {
4         int x = 0;
5         while (x<10)
6         {
7             System.out.println(x);
8             x++;
9         }
10    }
11 }
```

Output - Data (run)

```
run:
0
1
2
3
4
5
6
7
8
9
BUILD SUCCESSFUL (total time: 0 seconds)
```

2.2 do- while loop

Do-while loop mirip dengan *while-loop*. Pernyataan di dalam *do-while loop* akan dieksekusi beberapa kali selama kondisi bernilai benar(*true*).

Perbedaan antara *while* dan *do-while loop* adalah dimana pernyataan di dalam *do-while loop* akan dieksekusi sedikitnya **satu kali**.

Bentuk pernyataan do-while,

```
do{
    statement1;
    statement2;
    . . .
}
while( boolean_expression );
```

Pernyataan di dalam *do-while loop* akan dieksekusi pertama kali, dan akan dievaluasi kondisi dari *boolean_expression*. Jika nilai pada *boolean_expression* tersebut bernilai *true*, pernyataan di dalam *do-while loop* akan dieksekusi lagi. Berikut ini beberapa contoh do-while loop:



Start Page Data.java

Source History

1 package data;

2 public class Data {

3 public static void main(String[] args) {

4 int x = 0;

5 do

6 {

7 System.out.println(x);

8 x++;

9 }while (x<10);

10 }

11 }

Output - Data (run)

run:

0

1

2

3

4

5

6

7

8

9

BUILD SUCCESSFUL (total time: 0 seconds)

```
package data;

public class Data {

    public static void main(String[] args) {

        int x = 0;

        do

        {

            System.out.println(x);

            x++;

        }while (x<10);

    }

}
```

Contoh lain

Start Page Data.java

Source History

1 package data;

2 public class Data {

3 public static void main(String[] args) {

4 //one loop

5 // statement is executed once

6 do System.out.println("hello");

7 while (false);

8 }

9 }

10 }

Output - Data (run)

run:

hello

BUILD SUCCESSFUL (total time: 0 seconds)

2.3 for loop

Pernyataan *for loop* memiliki kondisi hampir mirip seperti struktur pengulangan sebelumnya yaitu melakukan pengulangan untuk mengeksekusi kode yang sama sebanyak jumlah yang telah ditentukan.

Bentuk dari *for loop*,

```
for (InitializationExpression; LoopCondition; StepExpression){  
    statement1;  
    statement2;  
    ...  
}
```

dimana,

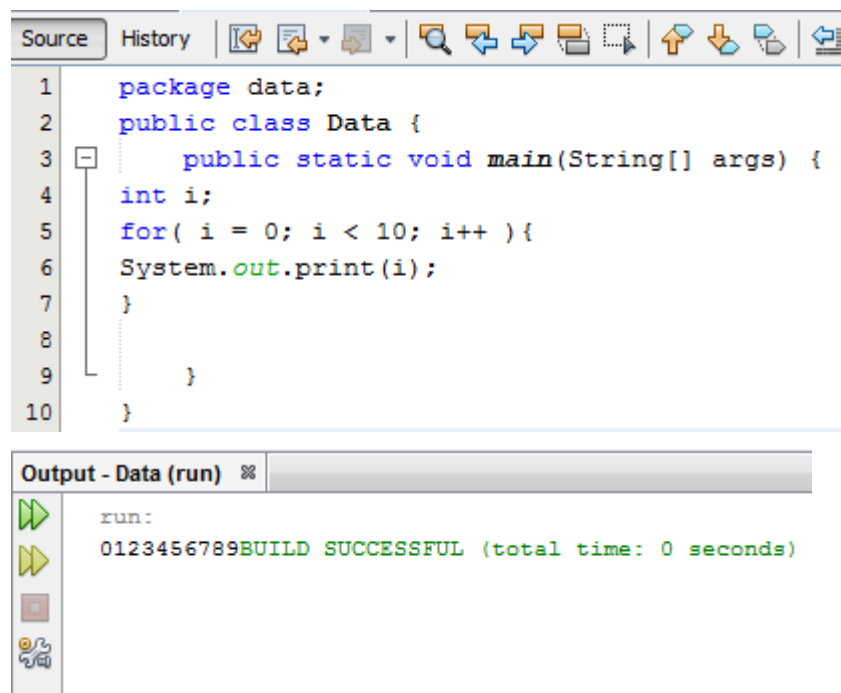
InitializationExpression - inisialisasi dari variabel loop.

LoopCondition - membandingkan variabel loop pada nilai batas tertentu

StepExpression - melakukan update pada variabel loop.

Berikut ini adalah contoh dari *for loop*,

```
int i;  
  
for( i = 0; i < 10; i++ ){  
    System.out.print(i);  
}
```



The screenshot shows an IDE window with a source code editor and an output console. The source code is as follows:

```
1 package data;  
2 public class Data {  
3     public static void main(String[] args) {  
4         int i;  
5         for( i = 0; i < 10; i++ ){  
6             System.out.print(i);  
7         }  
8     }  
9 }  
10 }
```

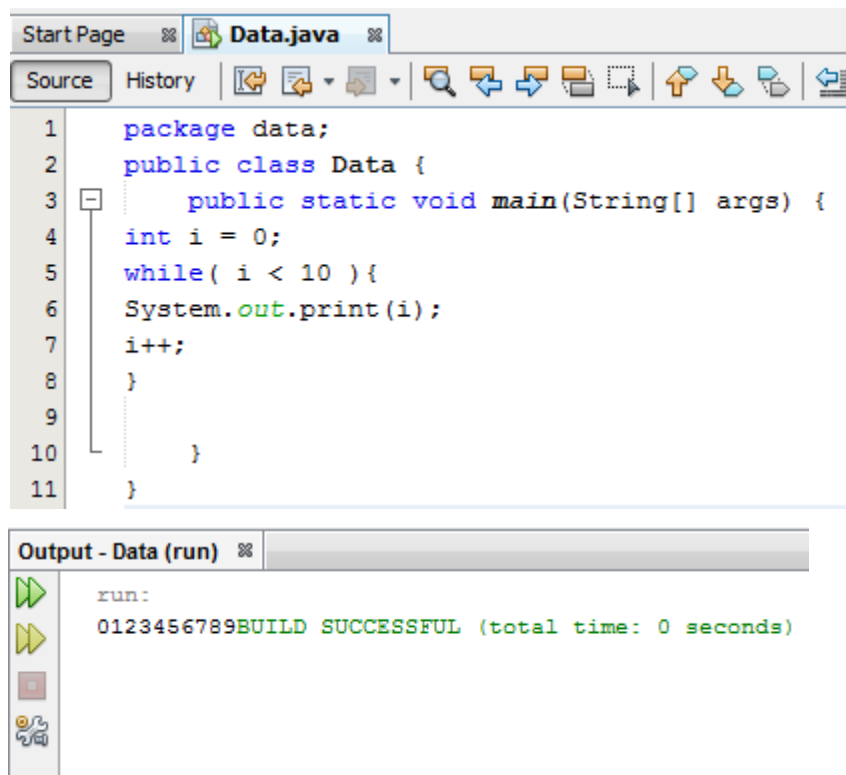
The output console shows the result of running the program:

```
run:  
0123456789BUILD SUCCESSFUL (total time: 0 seconds)
```

Pada contoh ini, pernyataan *i=0* merupakan inisialisasi dari variabel. Selanjutnya, kondisi *i<10* diperiksa. Jika kondisi bernilai *true*, pernyataan di dalam *for loop* dieksekusi. Kemudian, ekspresi *i++* dieksekusi, lalu akan kembali pada bagian pemeriksaan terhadap kondisi *i<10* lagi. Kondisi ini akan dilakukan berulang-ulang sampai mencapai nilai yang salah (*false*).

Contoh tadi, adalah contoh yang sama dari while loop,

```
int i = 0;
while( i < 10 ){
    System.out.print(i);
    i++;
}
```



The screenshot shows an IDE window titled 'Data.java' with the following code:

```
1 package data;
2 public class Data {
3     public static void main(String[] args) {
4         int i = 0;
5         while( i < 10 ){
6             System.out.print(i);
7             i++;
8         }
9     }
10 }
11 }
```

Below the code editor is the 'Output - Data (run)' window, which displays the following output:

```
run:
0123456789BUILD SUCCESSFUL (total time: 0 seconds)
```

3. Struktur Percabangan

Pernyataan percabangan memungkinkan kita untuk mengatur aliran eksekusi program. Java memberikan tiga bentuk pernyataan percabangan: break, continue dan return.

3.1 Statemen break

Pernyataan break memiliki dua bentuk: tidak berlabel (*unlabeled*) dan berlabel (*labeled*).

A. Statemen break tidak berlabel (unlabeled)

Pernyataan *break* tidak berlabel (*unlabeled*) digunakan untuk menghentikan jalannya pernyataan *switch*. Selain itu pernyataan *break unlabeled* juga bisa digunakan untuk menghentikan pernyataan-pernyataan *for*, *while* atau *do-while loop*.

Contoh program,

```
package data;
public class Data {
    public static void main(String[] args) {
        String names[] = {"Beah", "Bianca", "Lance", "Belle", "Nico", "Yza", "Gem",
            "Ethan"};
        StringsearchName = "Yza";
        boolean foundName = false;
        for(int i=0; i< names.length; i++){
            if( names[i].equals( searchName )){
                foundName = true;
            }
        }
    }
}
```



Java Programming

```
break;
}
}
if( foundName ){
    System.out.println( searchName + " found!" );
}
else{
    System.out.println( searchName + " not found." );
}
}
}
```

The screenshot shows an IDE window titled 'Data.java'. The code is as follows:

```
1 package data;
2 public class Data {
3     public static void main(String[] args) {
4         String names[] = {"Beah", "Bianca", "Lance", "Belle", "Nico", "Yza", "Gem", "Ethan"};
5         String searchName = "Yza";
6         boolean foundName = false;
7         for(int i=0; i< names.length; i++){
8             if( names[i].equals( searchName )){
9                 foundName = true;
10                break;
11            }
12        }
13        if( foundName ){
14            System.out.println( searchName + " found!" );
15        }
16        else{
17            System.out.println( searchName + " not found." );
18        }
19    }
20 }
```

Below the code editor is the 'Output - Data (run)' window, which displays the following text:

```
run:
Yza found!
BUILD SUCCESSFUL (total time: 0 seconds)
```

Pada contoh diatas, jika string “Yza” ditemukan, pengulangan pada *for loop* akan dihentikan dan akan dilanjutkan ke pernyataan berikutnya yang terletak setelah pernyataan *for*.

B. Statemen break berlabel

Bentuk label dari pernyataan *break* akan menghentikan pernyataan di luarnya, dimana sebelumnya harus diberikan label yang sudah di spesifikasikan pada program pada pernyataan *break*. Program berikut ini akan mencari nilai dalam array dua dimensi. Terdapat dua pengulangan bersarang (*nested loop*). Ketika sebuah nilai ditemukan, *break* akan menghentikan pernyataan yang diberi label *searchLabel* yang terletak di luar pernyataan *for loop*.

Contoh program;

```
package data;

public class Data {

    public static void main(String[] args) {

        int[][] numbers = {{1, 2, 3},
        {4, 5, 6},
        {7, 8, 9}};

        int searchNum = 5;
```



Java Programming

```
boolean foundNum = false;

searchLabel:
for( int i=0; i<numbers.length; i++ ){
    for( int j=0; j<numbers[i].length; j++ ){ if( searchNum == numbers[i][j] ){
        foundNum = true;
        break searchLabel;
    }
}
}

if( foundNum ){
    System.out.println( searchNum + " found!" );
}

else{
    System.out.println( searchNum + " not found!" );
}

}
```

```
1 package data;
2 public class Data {
3     public static void main(String[] args) {
4         int[][] numbers = {{1, 2, 3},
5                             {4, 5, 6},
6                             {7, 8, 9}};
7         int searchNum = 5;
8         boolean foundNum = false;
9         searchLabel:
10        for( int i=0; i<numbers.length; i++ ){
11            for( int j=0; j<numbers[i].length; j++ ){ if( searchNum == numbers[i][j] ){
12                foundNum = true;
13                break searchLabel;
14            }
15        }
16    }
17    if( foundNum ){
18        System.out.println( searchNum + " found!" );
19    }
20    else{
21        System.out.println( searchNum + " not found!" );
22    }
23 }
24 }
```

Output - Data (run)

```
run:
5 found!
BUILD SUCCESSFUL (total time: 0 seconds)
```

Pernyataan break menghentikan pernyataan yang diberi label; dan tidak menjalankan aliran kontrol apapun pada label. Aliran kontrol pada label akan diberikan secara otomatis pada pernyataan yang terletak dibawah label

3.2 Statemen Continue

Pernyataan *continue* memiliki dua bentuk: berlabel dan tidak berlabel. Anda dapat menggunakan pernyataan *continue* untuk melanjutkan pengulangan yang sedang dijalankan oleh pernyataan *for*, *while*, atau *do-while* loop.

A. Statemen continue tidak berlabel (*unlabeled*)

Bentuk pernyataan *continue* tidak berlabel (*unlabeled*) akan melewati bagian pernyataan setelah pernyataan ini dituliskan dan memeriksa ekspresi logika (*boolean*) yang mengontrol pengulangan. Jika ekspresi logika (*boolean*) masih bernilai *true*, maka pengulangan tetap dilanjutkan. Pada dasarnya pernyataan ini akan melanjutkan bagian pengulangan pada pernyataan loop.

Berikut ini adalah contoh dari penghitungan angka dari “Beah” dalam suatu array.

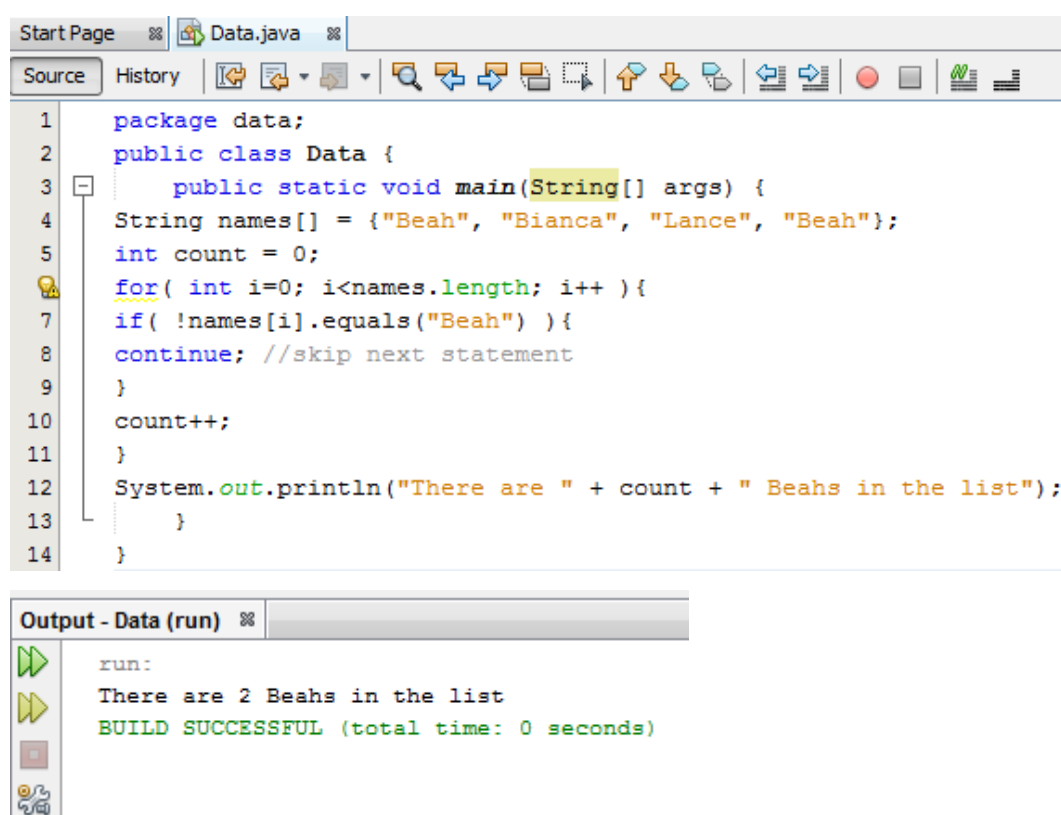
```
package data;

public class Data {

    public static void main(String[] args) {

        String names[] = {"Beah", "Bianca", "Lance", "Beah"};
        int count = 0;
        for( int i=0; i<names.length; i++ ){
            if( !names[i].equals("Beah") ){
                continue; //skip next statement
            }
            count++;
        }
        System.out.println("There are " + count + " Beahs in the list");
    }

}
```



The screenshot shows an IDE window with a tab for 'Data.java'. The code editor displays the same Java code as shown in the previous block. Below the code editor, the 'Output - Data (run)' window is visible, showing the execution results. The output indicates that the program ran successfully and printed 'There are 2 Beahs in the list'.

```
run:
There are 2 Beahs in the list
BUILD SUCCESSFUL (total time: 0 seconds)
```

B. Statemen continue berlabel (*labeled*)

Bentuk pernyataan continue berlabel (*labeled*) akan melanjutkan pengulangan yang sedang terjadi dan dilanjutkan ke pengulangan berikutnya dari pernyataan pengulangan yang diberi label (tanda).

Contoh program;

```
package data;

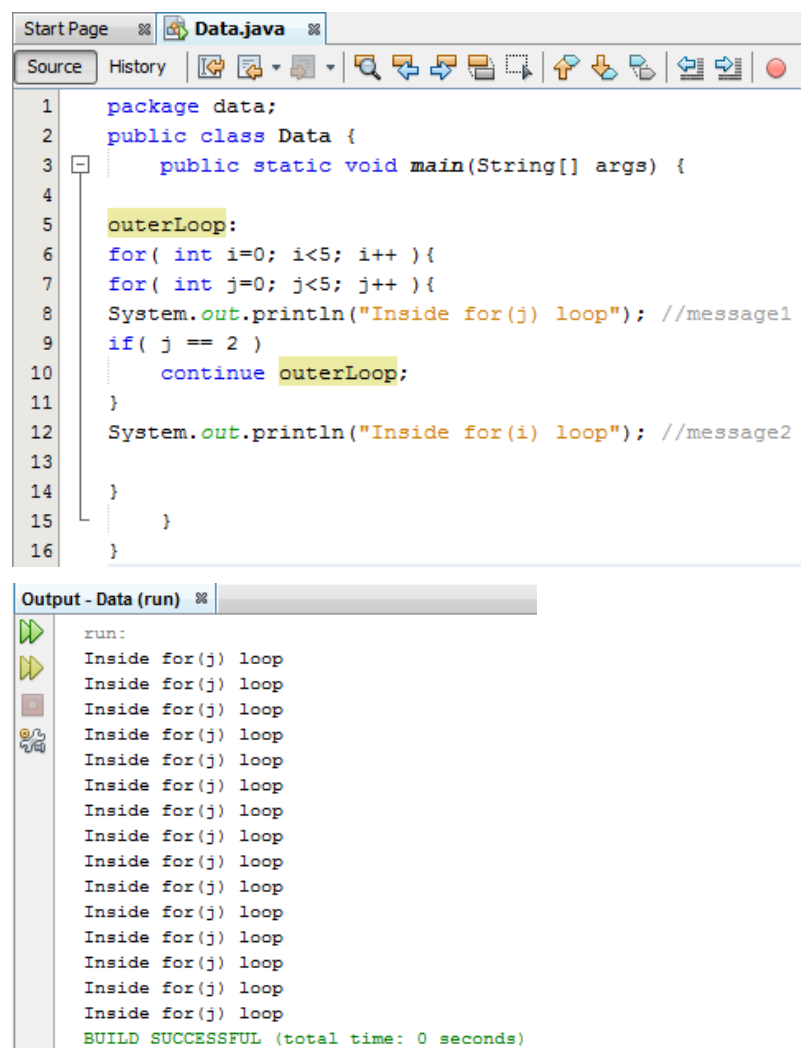
public class Data {

    public static void main(String[] args) {

        outerLoop:
        for( int i=0; i<5; i++ ){
            for( int j=0; j<5; j++ ){
                System.out.println("Inside for(j) loop"); //message1
                if( j == 2 )
                    continue outerLoop;
            }
            System.out.println("Inside for(i) loop"); //message2
        }

    }

}
```



```
Start Page  Data.java
Source History
1 package data;
2 public class Data {
3     public static void main(String[] args) {
4
5         outerLoop:
6         for( int i=0; i<5; i++ ){
7             for( int j=0; j<5; j++ ){
8                 System.out.println("Inside for(j) loop"); //message1
9                 if( j == 2 )
10                    continue outerLoop;
11             }
12             System.out.println("Inside for(i) loop"); //message2
13         }
14     }
15 }
16

Output - Data (run)
run:
Inside for(j) loop
Inside for(j) loop
Inside for(j) loop
Inside for(j) loop
Inside for(j) loop
Inside for(j) loop
Inside for(j) loop
Inside for(j) loop
Inside for(j) loop
Inside for(j) loop
Inside for(j) loop
Inside for(j) loop
Inside for(j) loop
Inside for(j) loop
Inside for(j) loop
BUILD SUCCESSFUL (total time: 0 seconds)
```

Pada contoh ini, bagian message2 tidak pernah akan dicetak, karena pernyataan continue akan melewati pengulangan.

3.3 Statemen return

Pernyataan *return* digunakan untuk keluar dari sebuah method. Pernyataan *return* memiliki dua bentuk: memberikan sebuah nilai, dan tidak memberikan nilai. Untuk memberikan sebuah nilai, cukup berikan nilai (atau ekspresi yang menghasilkan sebuah nilai) sesudah kata *return*. Contohnya

```
return ++count;
```

atau

```
return "Hello";
```

Tipe data dari nilai yang diberikan harus sama dengan tipe dari method yang dibuat. Ketika sebuah method *void* dideklasikan, gunakan bentuk *return* yang tidak memberikan nilai. Contohnya,

```
return;
```

Kita akan membahas lebih lanjut tentang pernyataan *return* ketika mempelajari tentang method.



Tugas

Catatan.

- a. Tugas dikumpulkan maks pada hari Jum'at tgl 11-04-2020 pukul 16.00
- b. Tugas boleh dikumpulkan dalam bentuk (PDF,PPT,WORD)
- c. File dikirim ke email : fahmi03031995@gmail.com
- d. Tugas program dapat dikerjakan di aplikasi (Dcoder, java ide for android, maupun software Netbeans,atau Notepad)
- e. Format file dengan tata cara: NamaMhs_NIM_PBO

Soal

1. Tulislah sebuah program untuk menerapkan statemen **if-else**.
2. Tulislah sebuah program yang menggunakan **if-else-if** bertangga atau **if-else** bersarang untuk menentukan musim dalam setahun.
(catatan :
musim dingin = desember, januari, februari;
musim semi = maret, april, mei;
musim panas = juni, juli agustus;
musim gugur = september, oktober, november)
3. Buat sebuah program yang mencetak nama Anda selama seratus kali. Buat tiga versi program ini menggunakan while loop, do while dan for-loop.