PERNYATAAN JUMP

OBJEKTIF :

1. Mahasiswa Mampu Memahami Pernyataan *Jump* pada Java.
2. Mahasiswa Mampu Menggunakan *Software* IntelliJ IDEA dalam Pembuatan Program dengan Pernyataan *Jump*.

PENDAHULUAN

Pernyataan *jump*(lompatan) merupakan kata kunci yang bertindak sebagai pengendali untuk merubah arah kendali program. Pada Java, terdapat beberapa pernyataan *jump* sebagai berikut:

1. `Break`
2. `Continue`
3. `Return`

Pernyataan-pernyataan ini mentransfer kendali ke bagian lain pada program, melakukan lompatan dari suatu baris pada program ke baris yang lain. Dengan adanya *keyword* tersebut, maka baris yang terdapat setelah *keyword* akan diabaikan. *Keyword-keyword* ini, umumnya terdapat pada sebuah perulangan untuk menghentikan dan melanjutkan pernyataan di dalam *statement* pada kondisi tertentu.

5.1 BREAK

Perintah `break` pada Java, umumnya digunakan untuk mengakhiri sebuah eksekusi dalam *statement*. Perintah `break` digunakan untuk keluar dari kendali suatu perulangan dan juga `switch`. Perintah `break` akan memberhentikan pernyataan di dalam *statement* dan tidak akan mengeksekusi pernyataan berikutnya yang masih tersisa di dalam perulangan.

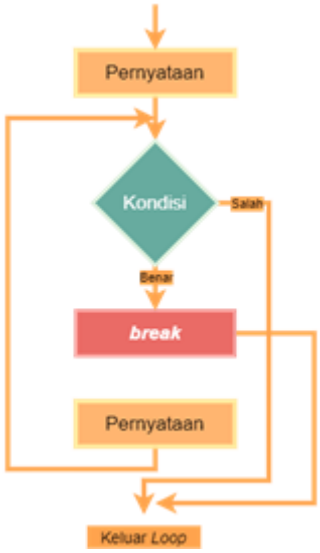
Terdapat dua penggunaan pernyataan `break`, yaitu:

1. Untuk mengakhiri pernyataan pada `switch`

```
Switch {
case: statement
break;
}
```

2. Untuk keluar dari *loop*

```
for(ekspresi) {
   if(kondisi) {
    break;
 }
}
```



Pada ilustrasi di atas, pernyataan `break` digunakan untuk keluar dari sebuah *loop*.

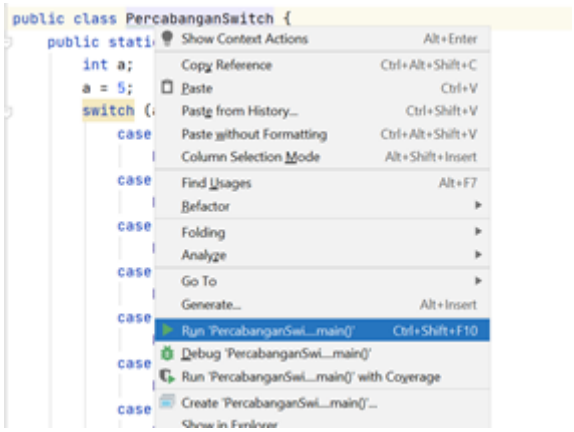
Untuk Mengakhiri Pernyataan pada `Switch`

Pernyataan `break` pada percabangan `switch` adalah opsional. Jika kita meniadakan `break`, maka eksekusi akan terus dilakukan ke pernyataan `case` berikutnya. Pernyataan `break` diadakan sebagai batas akhir dari *statement*. Untuk lebih mudah dipahami, berikut ini merupakan contoh program yang tidak menggunakan pernyataan `break`:

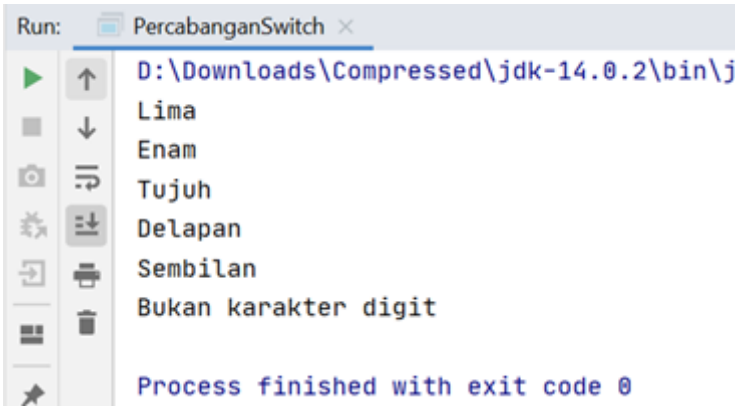
```
package com.integratedlaboratory.program;

public class PercabanganSwitch {
    public static void main (String args[]) {
        int a;
        a = 5;
        switch (a) {
            case 0: System.out.println("Nol");
            case 1: System.out.println("Satu");
            case 2: System.out.println("Dua");
            case 3: System.out.println("Tiga");
            case 4: System.out.println("Empat");
            case 5: System.out.println("Lima");
            case 6: System.out.println("Enam");
            case 7: System.out.println("Tujuh");
            case 8: System.out.println("Delapan");
            case 9: System.out.println("Sembilan");
            default: System.out.println("Bukan karakter digit");
        }
    }
}
```

Perintah:

Tekan tombol Ctrl+Shift+F10 untuk melakukan Run pada IntelliJ IDEA atau dengan melakukan klik kanan pada file java seperti berikut:



Hasil program:



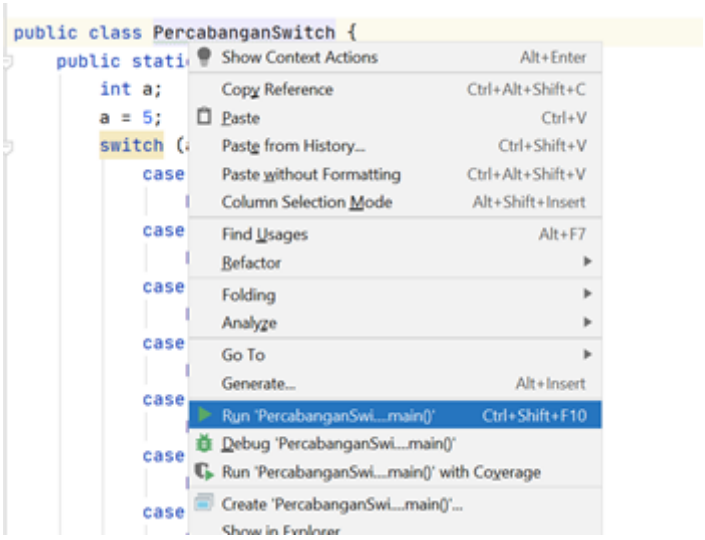
Berikut ini merupakan contoh program percabangan `switch` dengan menggunakan pernyataan `break`:

```
package com.integratedlaboratory.program;

public class PercabanganSwitch {
    public static void main (String args[]) {
        int a;
        a = 5;
        switch (a) {
            case 0: System.out.println("Nol");
                break;
            case 1: System.out.println("Satu");
                break;
            case 2: System.out.println("Dua");
                break;
            case 3: System.out.println("Tiga");
                break;
            case 4: System.out.println("Empat");
                break;
            case 5: System.out.println("Lima");
                break;
            case 6: System.out.println("Enam");
                break;
            case 7: System.out.println("Tujuh");
                break;
            case 8: System.out.println("Delapan");
                break;
            case 9: System.out.println("Sembilan");
                break;
            default: System.out.println("Bukan karakter digit");
```

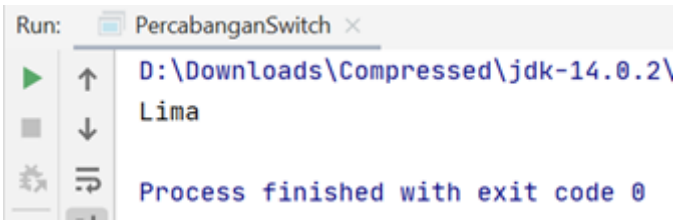
```
        }
    }
}
```

Perintah:

Tekan tombol Ctrl+Shift+F10 untuk melakukan Run pada IntelliJ IDEA atau dengan melakukan klik kanan pada file java seperti berikut:



Hasil program:



Terlihat bahwa dalam program pertama yang tidak menggunakan pernyataan `break`, program terus dilanjutkan ke *statement* selanjutnya setelah menemukan kondisi yang diberikan. Sedangkan pada program kedua, dengan menggunakan pernyataan `break`, *statement* langsung diberhentikan.

Untuk Keluar dari *Loop*

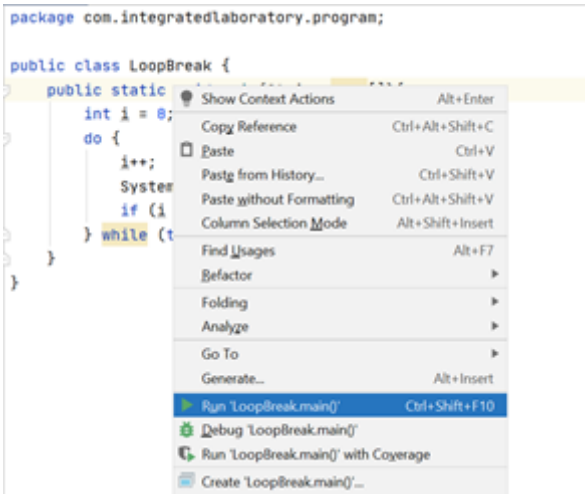
Pernyataan `break` keluar dari *loop* dan memotong kondisi *loop*. Dengan pernyataan `break`, kita akan memaksakan pengakhiran *loop*, memotong ekspresi kondisi, dan sisa kondisi setelah `break`. Ketika break berada di dalam *loop*, maka *loop* akan diakhiri dan kendali program akan diberikan ke pernyataan setelah *loop*.

Berikut ini adalah contoh program *loop* dengan menggunakan pernyataan `break`:

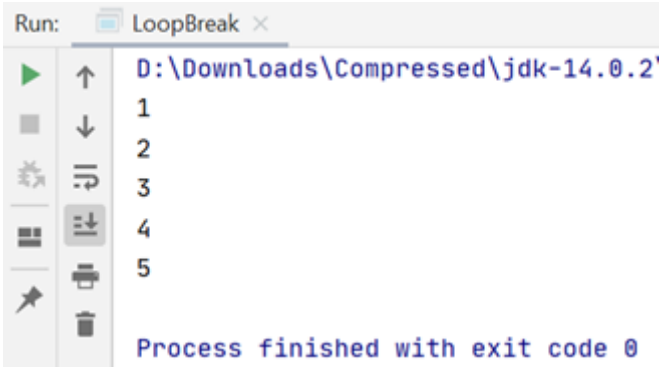
```java
package com.integratedlaboratory.program;

public class LoopBreak {
    public static void main(String args[]){
        int i = 0;
        do {
            i++;
            System.out.println(i);
            if (i == 5) break;
        } while (true);
    }
}
```

Perintah:

Tekan tombol Ctrl+Shift+F10 untuk melakukan Run pada IntelliJ IDEA atau dengan melakukan klik kanan pada file java seperti berikut:



Hasil program:



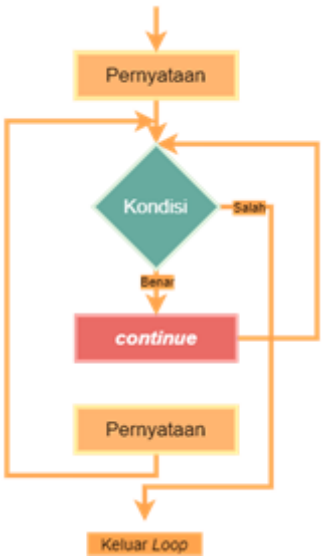
Dengan ada nya pernyataan `break`, saat a bernilai 5, eksekusi program akan menghentikan pengulangan `do-while`. Jika `break` dihilangkan, maka program akan terus melakukan *looping*.

5.2 CONTINUE

Pernyataan `continue` dapat digunakan untuk mengakhiri proses iterasi yang sedang berlangsung, melewati dan tidak mengsekusi sisa pernyataan yang masih ada di dalam pernyataan perulangan untuk kemudian melanjutkan proses ke iterasi berikutnya. Pernyataan `continue` tidak menghentikan program, melainkan hanya melewati perulangan saja. Baris-baris program setelah `continue` dalam perulangan tersebut akan diabaikan. `Continue` digunakan

untuk segera berlanjut ke perulangan berikutnya. Berikut ini merupakan penggunaan umum `continue` pada perulangan:

```
for(ekspresi) {
         if(kondisi) {
            continue;
         }
}
```

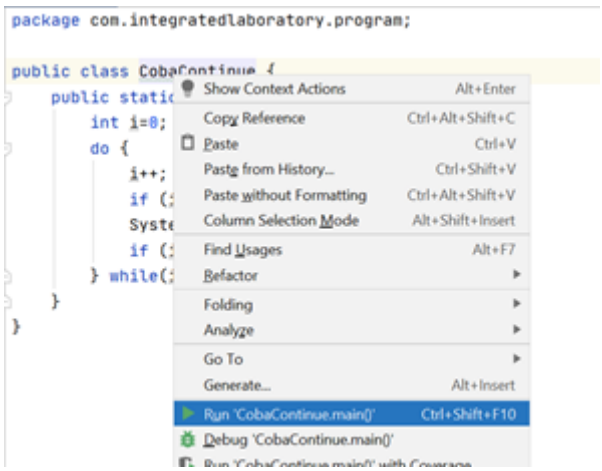


Berikut ini merupakan contoh penggunaan `continue` pada perulangan:

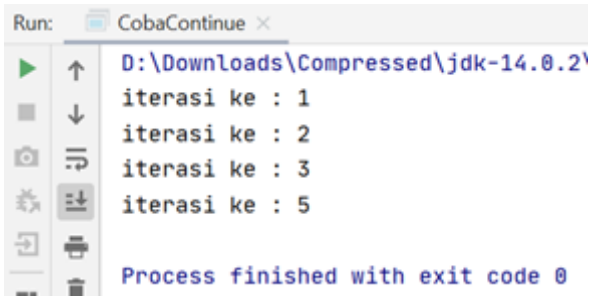
```
package com.integratedlaboratory.program;

public class CobaContinue {
    public static void main(String args[]) {
        int i=0;
        do {
            i++;
            if (i==4) continue;
            System.out.println("iterasi ke : "+i);
            if (i==5) break;
        } while(i <= 7);
    }
}
```

Perintah:

Tekan tombol Ctrl+Shift+F10 untuk melakukan Run pada IntelliJ IDEA atau dengan melakukan klik kanan pada file java seperti berikut:



Hasil program:



Hasil nya, tidak terdapat iterasi ke 4, dikarenakan saat i bernilai 4 maka kendali program akan melewati iterasi ke 4 dan melanjutkan ke iterasi berikutnya.

5.3 RETURN

Pernyataan `return` digunakan untuk keluar dari suatu *method*. Baris-baris program yang ada pada *method* yang sama setelah `return`, maka blok *method* tersebut akan diabaikan. Kemudian akan dilanjutkan eksekusi setelah blok *method* tersebut. Perintah `return` digunakan untuk mengembalikan sebuah nilai. Berikut ini merupakan sintaks umum penggunaan `return` pada program:

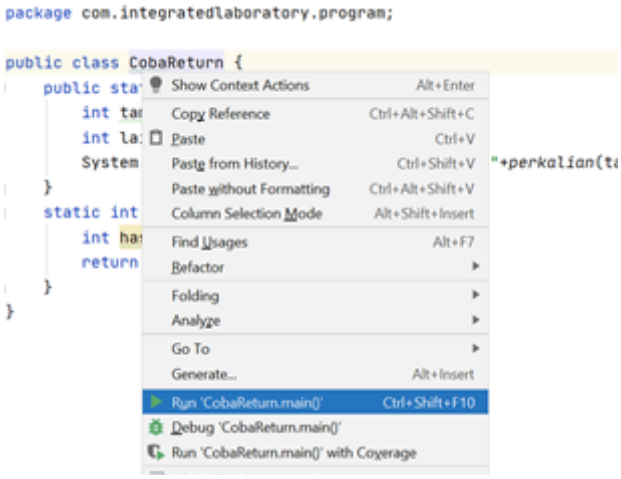
```
static tipe_data nama_method(ekspresi) {
     statement;
     return true;
}
```

Berikut ini merupakan contoh program dengan pernyataan `return`:

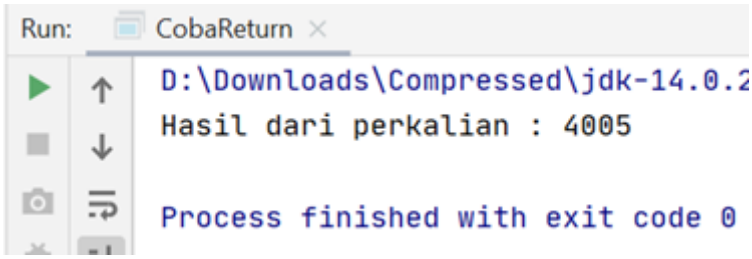
```java
package com.integratedlaboratory.program;

public class CobaReturn {
    public static void main(String[]args){
        int tambah = 20;
        int lain = 5;
        System.out.println("Hasil dari perkalian : "+perkalian(tambah) + lain);
    }
    static int perkalian(int a){
        int hasil = a * a;
        return hasil;
    }
}
```

Perintah:

Tekan tombol Ctrl+Shift+F10 untuk melakukan Run pada IntelliJ IDEA atau dengan melakukan klik kanan pada file java seperti berikut:



Hasil program:



Program di atas melakukan 20 dikali 20 dengan hasil yang seharusnya adalah 400. Lalu 400 tersebut dijumlahkan dengan 5, sehingga hasil seharusnya adalah 405. Namun pada *output* hasilnya adalah 4005 dikarenakan variabel 'lain' hanya menambah nilai nya saja, bukan menjumlahkan.

REFERENSI:

[1] Hariyanto, Bambang. 2017. *Esensi-Esensi Bahasa Pemrograman Java Revisi Kelima*. Bandung: Informatika.

[2] URL: https://docplayer.info/47530648-Looping-break-continue-nested-loop.html. Diakses pada 30 Juli 2020.

[3] Thakhur, Nishtha. 2018. "How to use break and continue statements in Java?", https://www.tutorialspoint.com/How-to-use-break-and-continue-statements-in-Java. Diakses pada 30 Juli 2020.