

Algoritma dan Pemrograman Komputer 1

Bab 8: Pernyataan Percabangan

Aslam Pandu Tasminto – 5002241025

M. Ma'ruf Qomaruddin Kafi – 5002241095

October 28, 2025

Departemen Matematika

Fakultas Sains dan Analitika Data

Institut Teknologi Sepuluh Nopember

Daftar Isi

Pengenalan Pernyataan Percabangan

BREAK Statement

CONTINUE Statement

RETURN Statement

Perbandingan BREAK vs CONTINUE vs RETURN

Best Practices

Contoh Kasus

Latihan

Kesimpulan

Referensi

Pengenalan Pernyataan Percabangan

Konsep Pernyataan Percabangan dalam Java

Apa itu Percabangan?

Pernyataan percabangan adalah pernyataan yang digunakan untuk **mengatur alur eksekusi** program dengan cara mengubah urutan normal eksekusi kode. Seperti **tombol kontrol** dalam program untuk:

- **Berhenti** di tengah proses (break)
- **Lompati** langkah tertentu (continue)
- **Keluar** dari fungsi (return)

Analoginya:

Bayangkan sedang membaca buku:

- break = berhenti baca bab ini
- continue = lewati halaman ini
- return = tutup buku, selesai membaca

BREAK Statement

Sintaks dan Konsep BREAK

Fungsi BREAK Statement

Menghentikan eksekusi loop (perulangan) atau percabangan (if else dan switch case) secara paksa dan keluar dari blok kode tersebut.

Jenis BREAK	Deskripsi
break (unlabeled)	Menghentikan loop/switch terdekat
break label (labeled)	Menghentikan loop dengan label tertentu

Tabel 1: Jenis-jenis BREAK Statement

BREAK Tidak Berlabel (Unlabeled)

Contoh BREAK dalam FOR Loop

```
1 public class BreakUnlabeled {  
2     public static void main(String[] args) {  
3         for (int i = 0; i < 5; i++) {  
4             System.out.println("i = " + i);  
5             if (i == 2) {  
6                 break; // Keluar loop saat i = 2  
7             }  
8         }  
9         System.out.println("Loop selesai");  
10    }  
11}  
12}
```

Output Program

```
i = 0  
i = 1  
i = 2  
  
Loop selesai
```

BREAK Berlabel (Labeled)

Contoh BREAK dengan Label

```
1 public class BreakLabeled {  
2     public static void main(String[] args) {  
3         int x = 35;  
4         int y = 4;  
5         int dummy = y;  
6  
7         mulai:  
8         while(true) {  
9             if(y > x) {  
10                 System.out.println("Nilai kelipatan y yang mendekati x adalah " + y);  
11                 break mulai; // Keluar dari blok 'mulai'  
12             }  
13             y = y + dummy;  
14         }  
15     }  
16 }  
17 }
```

Output Program

Nilai kelipatan y yang mendekati x adalah 36

CONTINUE Statement

Sintaks dan Konsep CONTINUE

Fungsi CONTINUE Statement

Melompati sisa iterasi saat ini dalam loop dan langsung melanjutkan ke iterasi berikutnya.

Jenis CONTINUE	Deskripsi
continue (unlabeled)	Melompati iterasi dalam loop terdekat
continue label (labeled)	Melompati iterasi dalam loop berlabel

Tabel 2: Jenis-jenis CONTINUE Statement

CONTINUE Tidak Berlabel (Unlabeled)

Contoh CONTINUE dalam FOR Loop

```
1 public class ContinueUnlabeled {  
2     public static void main(String[] args) {  
3         for(int i = 0; i < 5; i++) {  
4             if(i == 2) {  
5                 continue; // Lewati saat i = 2  
6             }  
7             System.out.println("i = " + i);  
8         }  
9     }  
10 }  
11 }
```

Output Program

```
i = 0  
i = 1  
i = 3  
i = 4
```

RETURN Statement

Sintaks dan Konsep RETURN

Fungsi RETURN Statement

Menghentikan eksekusi program atau method.

Dalam **main method** return digunakan untuk menghentikan program secara paksa.

Note: Return untuk method akan dipelajari di Modul 9 (Fungsi/Method)

Jenis RETURN	Deskripsi
return (di main)	Menghentikan program secara paksa
return value (di method)	Mengembalikan nilai (Modul 9)

Tabel 3: Jenis-jenis RETURN Statement

RETURN untuk Menghentikan Program

Contoh RETURN di Main Method

```
1 class kembali {
2     public static void main (String args[]) {
3         System.out.println("Masukan nilai suatu bilangan");
4         Scanner baca = new Scanner(System.in);
5         int k = baca.nextInt();
6
7         if(k % 2 == 0) {
8             System.out.println(k + " merupakan bilangan genap");
9             return; // BERHENTI di sini ketika k genap
10        }
11
12        System.out.println(k + " merupakan bilangan ganjil");
13    }
14 }
```

Output Program (input = 4)

```
Masukan nilai suatu bilangan
```

```
4
```

```
4 merupakan bilangan genap
```

RETURN untuk Menghentikan Program (Part 2)

Method check() - Akan dipelajari di Modul 9

```
1 // Method ini akan dipelajari di Modul 9
2 public static String check(int k) {
3     if(k % 2 == 0) {
4         return "genap";
5     } else {
6         return "ganjil";
7     }
8 }
9 }
10 }
```

Output Program (jika input = 7)

Masukan nilai suatu bilangan

7

7 merupakan bilangan ganjil

Perbandingan BREAK vs CONTINUE vs RETURN

Perbandingan BREAK vs CONTINUE vs RETURN

Kriteria	BREAK	CONTINUE	RETURN
Efek	Menghentikan loop	Skip iterasi	Keluar method
Scope	Loop/switch	Loop saja	Method saja
Eksekusi selanjutnya	Keluar loop	Iterasi berikutnya	Pemanggil method
Penggunaan	Pencarian, exit condition	Filter data	Early exit, validasi
Dapat berlabel	Ya	Ya	Tidak
Dipakai di switch	Ya	Tidak	Ya

Tabel 4: Perbandingan BREAK vs CONTINUE vs RETURN Statement

Best Practices

Best Practices Penggunaan Percabangan

Tips dan Rekomendasi

- **Gunakan BREAK dengan hati-hati** - Dapat membuat alur program sulit dilacak
- **Hindari BREAK/CONTINUE berlebihan** - Dapat mengurangi readability
- **Pertimbangkan alternatif** - Terkadang kondisi loop yang baik lebih baik daripada BREAK
- **Gunakan CONTINUE untuk skip kondisi** - Lebih jelas daripada nested if
- **Gunakan RETURN untuk early exit** - Efisien untuk validasi input di awal method
- **Hindari multiple return points** - Kecuali untuk early validation yang jelas

Contoh Kasus

Studi Kasus 1: Validasi Input

Validasi Input dengan RETURN Early

```
1 public class InputValidator {  
2     public static void main(String[] args) {  
3         System.out.println("Masukkan usia:");  
4         Scanner input = new Scanner(System.in);  
5         int age = input.nextInt();  
6  
7         if(age < 0) {  
8             System.out.println("Usia tidak valid");  
9             return; // Hentikan program  
10        }  
11        if(age > 150) {  
12            System.out.println("Usia tidak realistik");  
13            return; // Hentikan program  
14        }  
15  
16        System.out.println("Usia valid: " + age);  
17    }  
18}  
19
```

Studi Kasus 2: Pencarian Karakter

Pencarian dengan BREAK

```
1 public class CariKarakter {
2     public static void main(String[] args) {
3         String teks = "programming";
4         char target = 'g';
5
6         for(int i = 0; i < teks.length(); i++) {
7             if(teks.charAt(i) == target) {
8                 System.out.println("Karakter '" + target + "' ditemukan
9                 di index: " + i);
10                break; // Berhenti setelah ketemu pertama
11            }
12        }
13    }
14 }
```

Latihan

Latihan 1: Bilangan Prima dengan BREAK

Soal 1: Cek Bilangan Prima

Buat program yang mengecek apakah suatu bilangan prima.

Gunakan **BREAK** untuk mengoptimasi proses pengecekan.

Latihan 2: Skip Bilangan Genap

Soal 2: Cetak Bilangan Ganjil dengan CONTINUE

Buat program yang mencetak bilangan ganjil dari 1 sampai 20.

Gunakan **CONTINUE** untuk melewaskan bilangan genap.

Kesimpulan

Inti Bab 8: Pernyataan Percabangan

- **BREAK**: Menghentikan perulangan/struktur kontrol secara paksa
- **CONTINUE**: Melompati iterasi saat ini dan lanjut ke iterasi berikutnya
- **RETURN**: Keluar dari . Untuk method non main bisa dengan nilai kembalian

Referensi

Referensi

Referensi:

- **Modul Praktikum Algoritma dan Pemrograman - Modul 8**
Departemen Matematika FSAD ITS
- **Oracle Java Tutorials - Branching Statements**
<https://docs.oracle.com/javase/tutorial/java/nutsandbolts/branch.html>
- **GeeksforGeeks - Java Break and Continue**
<https://www.geeksforgeeks.org/break-and-continue-statement-in-java/>
- **Tutorialspoint - Java Methods**
https://www.tutorialspoint.com/java/java_methods.htm

Terima Kasih