# 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

# 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

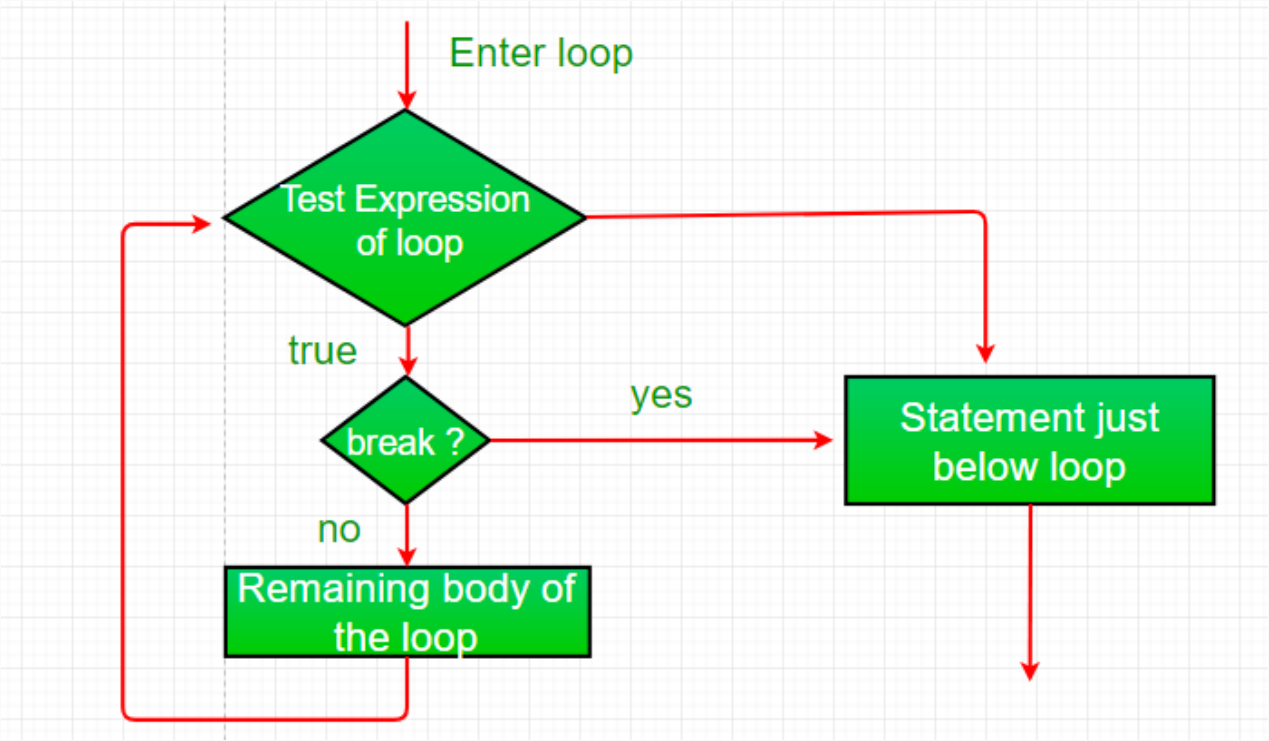
# Sintaks dan Konsep BREAK

**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

# Flowchart BREAK




**Gambar 1:** Flowchart Break

**Sumber:** GeeksforGeeks - Java 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

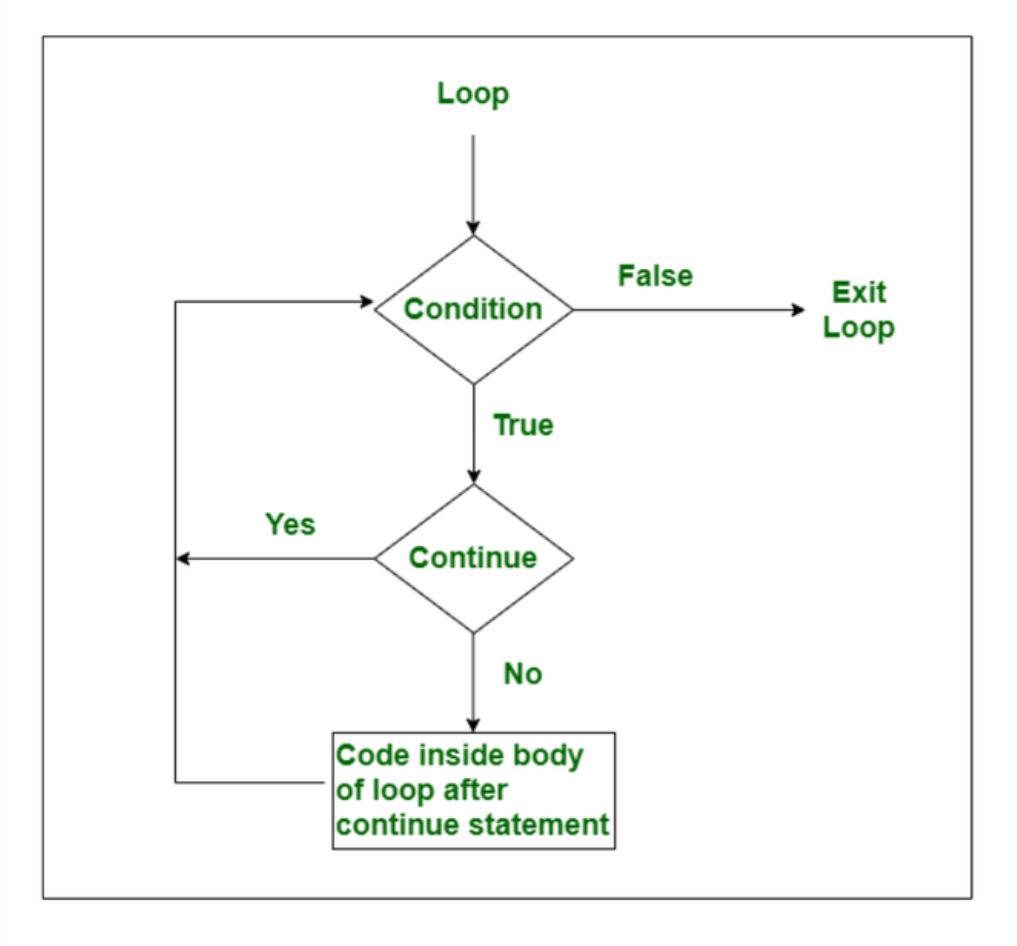
# Sintaks dan Konsep CONTINUE

**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

# Flowchart Continue




**Gambar 2:** Flowchart Continue

**Sumber:** GeeksforGeeks - Java 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

## 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 }
15
```

## 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 realistis");
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
       di index: " + i);
9                  break;  // Berhenti setelah ketemu pertama
10             }
11         }
12     }
13 }
14
```

# Latihan

# Latihan 1: Bilangan Prima dengan BREAK

**Soal 1: Optimasi Cek Bilangan Prima**
Buat program yang meminta input sebuah bilangan bulat positif `n`, kemudian menentukan apakah `n` merupakan bilangan prima atau bukan.

**Gunakan BREAK** untuk mengoptimasi proses - berhenti pengecekan begitu ditemukan pembagi selain 1 dan n itu sendiri.

**Contoh:**

- Input: 7 $\rightarrow$ Output: "7 adalah bilangan prima"
- Input: 9 $\rightarrow$ Output: "9 bukan bilangan prima"

# Latihan 2: Skip Bilangan Genap

**Soal 2: Cetak Bilangan Ganjil dengan CONTINUE**
Buat program yang meminta input batas atas `n`, kemudian mencetak semua bilangan ganjil dari 1 sampai `n`.

**Gunakan CONTINUE** untuk melewatkan bilangan genap dalam loop.

**Contoh (n = 10):**

- Output: 1 3 5 7 9

# Latihan 3: Validasi Input dengan RETURN

**Soal 3: Early Exit dengan RETURN**
Buat program yang meminta input usia pengguna. Gunakan **RETURN** untuk early exit ketika:

- Input usia negatif $\rightarrow$ tampilkan "Usia tidak valid" dan hentikan program
- Input usia $> 150 \rightarrow$ tampilkan "Usia tidak realistis" dan hentikan program
- Selain itu $\rightarrow$ tampilkan "Usia valid: [usia] tahun"

**Contoh:**

- Input: -5 $\rightarrow$ Output: "Usia tidak valid"
- Input: 25 $\rightarrow$ Output: "Usia valid: 25 tahun"

# Kesimpulan

# 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 method untuk method main. 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/`

# Referensi

**Referensi:**


- **GeeksforGeeks - Java Break Statement**
  `https://www.geeksforgeeks.org/java/break-statement-in-java/`
- **GeeksforGeeks - Java Continue Statement**
  `https://www.geeksforgeeks.org/java/continue-statement-in-java/`

# Terima Kasih