

---

# Pengenalan Java

---

# Sejarah Java

- Java adalah bahasa pemrograman yang dibuat oleh James Gosling saat bekerja di Sun Microsystem
- Java dirilis ke public tahun 1995
- Java adalah bahasa pemrograman berorientasi objek dan mendukung pengelolaan memori secara otomatis
- Saat ini perusahaan Sun Microsystem telah dibeli oleh Oracle
- Java terkenal dengan write once, run anywhere, karena binary program Java di-generate secara independen dan bisa dijalankan di Java Virtual Machine yang terinstall di berbagai sistem operasi

---

# Teknologi Java

- Java Standard Edition
- Java Enterprise Edition
- Java Micro Edition



# Versi Java

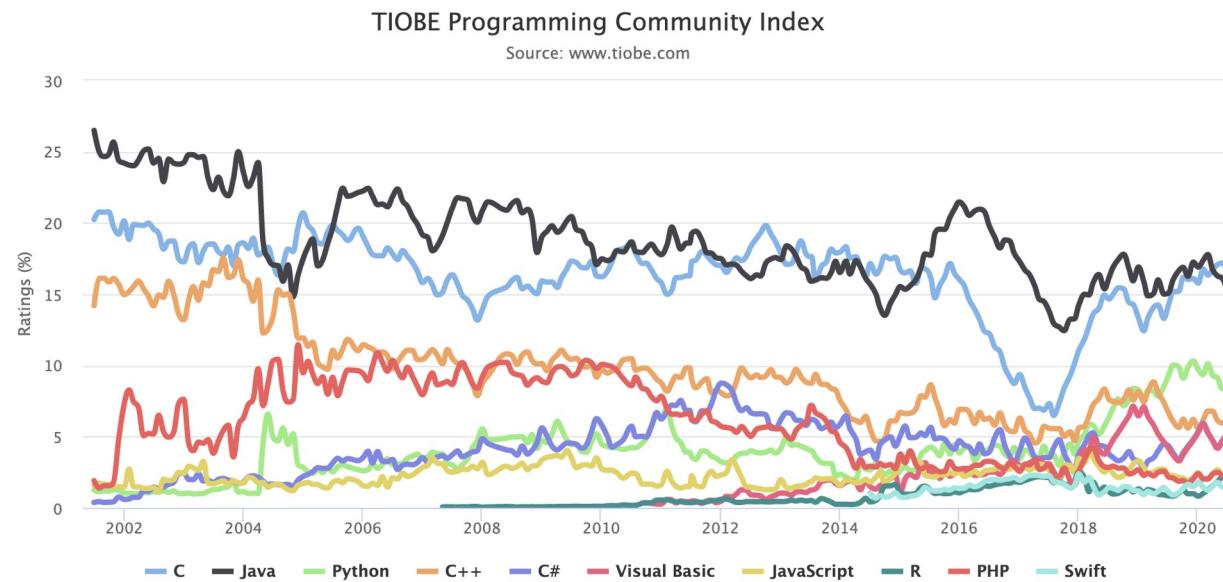
| Version  | Release Date |
|----------|--------------|
| Java 1.0 | 1996/01/23   |
| Java 1.1 | 1997/02/19   |
| Java 1.2 | 1998/12/08   |
| Java 1.3 | 2000/05/08   |
| Java 1.4 | 2002/02/06   |
| Java 5   | 2004/09/30   |
| Java 6   | 2006/12/11   |
| Java 7   | 2011/07/07   |
| Java 8   | 2014/03/18   |

|         |            |
|---------|------------|
| Java 9  | 2017/09/21 |
| Java 10 | 2018/03/20 |
| Java 11 | 2018/09/25 |
| Java 12 | 2019/03/19 |
| Java 13 | 2019/09/17 |
| Java 14 | 2020/03/17 |
| Java 15 | 2020/09/15 |
| Java 16 | March 2021 |

Data Source

---

# Kenapa Belajar Java



---

# Dimana Java Banyak Digunakan?

- Backend, banyak perusahaan besar saat ini menggunakan Java sebagai aplikasi backend nya seperti Twitter, Netflix, Spotify, Amazon, Alibaba, Blibli, dan lain-lain
- Big Data, teknologi-teknologi big data yang saat ini populer, kebanyakan adalah teknologi Java, seperti Apache Hadoop, Elasticsearch, Apache Cassandra, Apache Spark, Apache Kafka, dan lain-lain
- Android, di Android kita bisa menggunakan Java dan Kotlin untuk membuat aplikasi nya

---

## JRE vs JDK

- JRE singkatan dari Java Runtime Environment
- JDK singkatan dari Java Development Kit

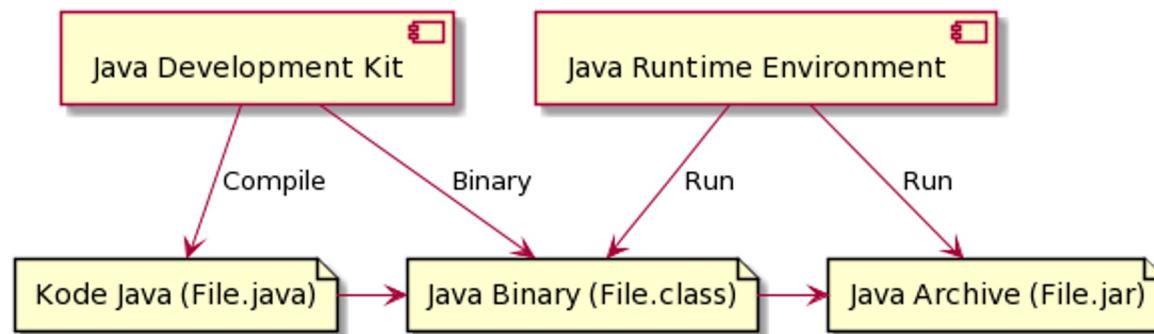
---

# Java Virtual Machine

- Java sendiri hanyalah bahasa pemrograman, otak dibalik teknologi Java sebenarnya sebuah teknologi yang disebut Java Virtual Machine
- Java Virtual Machine merupakan program yang digunakan untuk mengeksekusi binary file Java
- Karena JVM hanya mengerti binary file, sehingga akhirnya banyak bahasa pemrograman yang mengadopsi teknologi JVM, seperti Kotlin, Scala, Groovy dan lain-lain
- Dengan begitu, banyak bahasa pemrograman yang lebih canggih dari Java, namun mereka tetap jalan di JVM yang sudah terbukti stabil dan bagus

---

# Proses Development Program Java



---

# Menginstall Java



# OpenJDK

- OpenJDK adalah salah satu implementasi Java Development Kit yang opensource dan gratis
- <https://openjdk.java.net/>

---

# OpenJDK vs yang lain

- Oracle JDK : <https://www.oracle.com/java/technologies/javase-downloads.html>
- Amazon Corretto : <https://aws.amazon.com/id/corretto/>
- Zulu : <https://www.azul.com/downloads/zulu-community/>

---

# Download OpenJDK

- <https://jdk.java.net/>



# Setting PATH

- Windows : <https://medium.com/programmer-zaman-now/setting-java-path-di-windows-4da2c65d8298>
- Linux atau Mac

```
# Add to .bashrc or .profile or .zshrc

export JAVA_HOME="/Library/Java/JavaVirtualMachines/jdk1.8.0_241.jdk/Contents/Home"
export PATH="$JAVA_HOME/bin:$PATH"
```



# Integrated Development Environment

- IDE adalah smart editor yang digunakan untuk mengedit kode program
- IDE juga digunakan untuk melakukan otomatisasi proses kompilasi kode program dan otomatisasi proses menjalankan program

IDE untuk Java

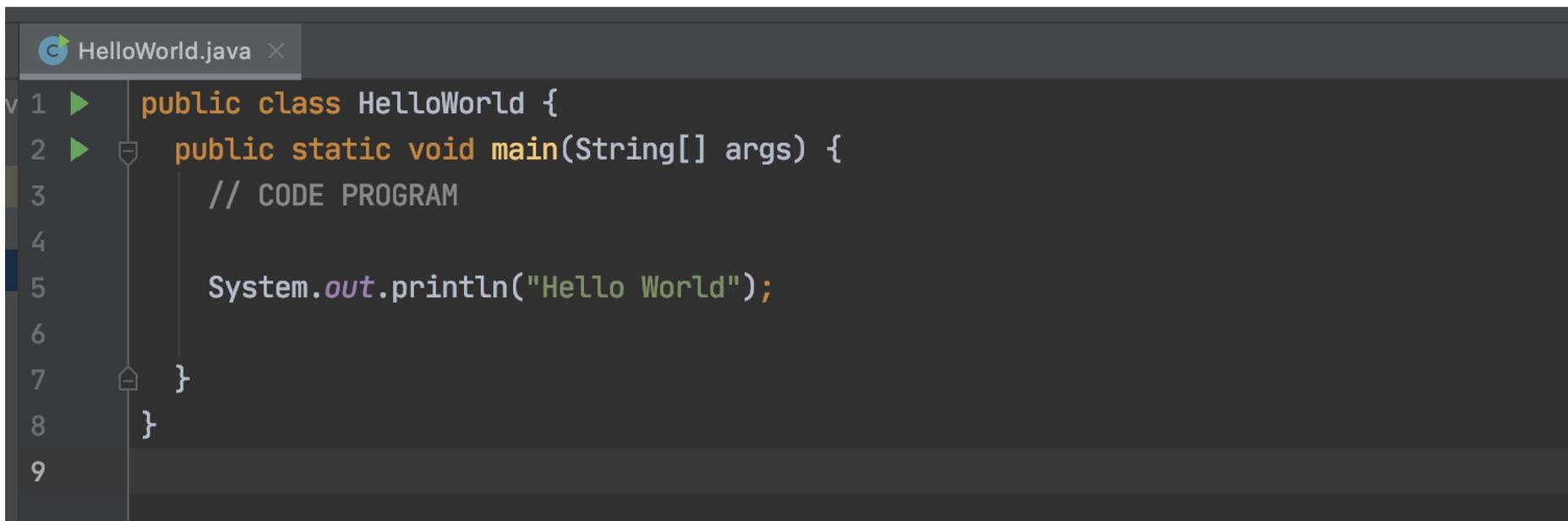
- IntelliJ IDEA Ultimate / Community : <https://www.jetbrains.com/idea/>
- Eclipse : <https://www.eclipse.org/downloads/packages/>
- NetBeans : <https://netbeans.apache.org/>
- JDeveloper : <https://www.oracle.com/application-development/technologies/jdeveloper.html>

---

# Program Hello World



# Program Hello World



The screenshot shows a code editor window with the following details:

- Title Bar:** Shows the file name "HelloWorld.java" with a close button.
- Code Area:** Displays the Java code for a "Hello World" program.

```
v 1 ► public class HelloWorld {  
2 ►   ►   public static void main(String[] args) {  
3 ►     // CODE PROGRAM  
4  
5       System.out.println("Hello World");  
6  
7     }  
8   }  
9 }
```

The code uses color-coded syntax highlighting: blue for keywords like `public`, `class`, `void`, and `main`, orange for the class name `HelloWorld` and variable `args`, purple for the output stream `System.out`, and green for the string literal `Hello World`.
- Status Bar:** A small portion of the status bar is visible at the bottom, showing some icons.



# Kompilasi Kode Java

```
→ src javac HelloWorld.java
→ src ls -l
total 16
-rw-r--r-- 1 khannedy  staff  425 Jul  6 12:17 HelloWorld.class
-rw-r--r-- 1 khannedy  staff  136 Jul  6 12:16 HelloWorld.java
→ src java HelloWorld
Hello World
→ src █
```

---

# Tipe Data Number

---

# Tipe Data Number

- Integer Number
- Floating Point Number



# Integer Number

| Tipe Data | Min                            | Max                           | Size    | Default |
|-----------|--------------------------------|-------------------------------|---------|---------|
| byte      | -128                           | 127                           | 1 byte  | 0       |
| short     | -32,768                        | 32,767                        | 2 bytes | 0       |
| int       | -2,147,483,648                 | 2,147,483,647                 | 4 bytes | 0       |
| long      | -9,223,372,036,<br>854,775,808 | 9,223,372,036,<br>854,775,807 | 8 bytes | 0       |

---

## Kode : Integer Number

```
byte iniByte = 100;
short iniShort = 1000;
int iniInt = 1000000;
long iniLong = 10000000;
long iniLong2 = 10000000L;
```

```
}
```



# Floating Point Number

| Tipe Data | Min      | Max      | Size    | Default |
|-----------|----------|----------|---------|---------|
| float     | 3.4e-038 | 3.4e+038 | 4 bytes | 0.0     |
| double    | 1.7e-308 | 1.7e+308 | 8 bytes | 0.0     |



## Kode : Floating Point Number

```
float iniFloat = 10.12F;  
double iniDouble = 12.2424;
```

```
}
```



## Kode : Literals

```
int decimalInt = 25;  
int hexInt = 0xA132B;  
int binInt = 0b01010101;
```

```
}
```



# Kode : Underscore

```
long balance = 1_000_000_000_000L;
int sum = 60_000_000;
```

```
}
```

---

# Konversi Tipe Data Number

- Widening Casting (Otomatis) : byte -> short -> int -> long -> float -> double
- Narrowing Casting (Manual) : double -> float -> long -> int -> char -> short -> byte

---

# Kode : Konversi Tipe Data Number

```
byte iniByte = 10;
short iniShort = iniByte;
int iniInt = iniShort;
long iniLong = iniInt;
float iniFloat = iniLong;
double iniDouble = iniFloat;

float iniFloat2 = (float) iniDouble;
long iniLong2 = (long) iniFloat2;
int iniInt2 = (int) iniLong2;
short iniShort2 = (short) iniInt2;
```

---

# Tipe Data Character

---

# Tipe Data Character

- Data Character (huruf) di Java direpresentasikan oleh tipe char.
- Untuk membuat data char di Java, kita bisa menggunakan tanda ' (petik satu) di awal dan di akhir karakter

---

## Kode : Character

```
char e = 'E';
char k = 'K';
char o = 'O';

System.out.print(e);
System.out.print(k);
System.out.print(o);
```



---

# Tipe Data Boolean

---

# Tipe Data Boolean

- Tipe data boolean adalah tipe data yang memiliki 2 nilai, yaitu benar dan salah
- Tipe data boolean di Java direpresentasikan dengan kata kunci boolean
- Nilai benar direpresentasikan dengan kata kunci true
- Nilai salah direpresentasikan dengan kata kunci false
- Default value untuk boolean adalah false

---

## Kode : Boolean

```
boolean benar = true;  
boolean salah = false;  
  
System.out.println(benar);  
System.out.println(salah);
```



---

# Tipe Data String

---

# Tipe Data String

- Tipe data String adalah tipe data yang berisikan data kumpulan karakter atau sederhananya adalah teks
- Di Java, tipe data String direpresentasikan dengan kata kunci String
- Untuk membuat String di Java, kita menggunakan karakter “ (petik dua) sebelum dan setelah teks nya
- Default value untuk String adalah null

---

# Kode : String

```
String firstName = "Eko Kurniawan";
String lastName = "Khannedy";

System.out.println(firstName);
System.out.println(lastName);

}
}
```

---

# Kode : Menggabungkan String

```
String firstName = "Eko Kurniawan";
String lastName = "Khannedy";
String fullName = firstName + " " + lastName;

System.out.println(firstName);
System.out.println(lastName);
System.out.println(fullName);

}
```

---

# Variable

---

# Variable

- Variable adalah tempat untuk menyimpan data
- Java adalah bahasa static type, sehingga sebuah variable hanya bisa digunakan untuk menyimpan tipe data yang sama, tidak bisa berubah-ubah tipe data seperti di bahasa pemrograman PHP atau JavaScript
- Untuk membuat variable di Java kita bisa menggunakan nama tipe data lalu diikuti dengan nama variable nya
- Nama variable tidak boleh mengandung whitespace (spasi, enter, tab), dan tidak boleh seluruhnya number



## Kode : Variable

```
String name;  
name = "Eko Kurniawan Khannedy";  
  
int age = 30;  
String address = "Indonesia";  
  
System.out.println(name);  
System.out.println(age);  
System.out.println(address);
```

---

## Kata Kunci var

- Sejak versi Java 10, Java mendukung pembuatan variabel dengan kata kunci var, sehingga kita tidak perlu menyebutkan tipe datanya
- Namun perlu diingat, saat kita menggunakan kata kunci var untuk membuat variable, kita harus menginisiasi value / nilai dari variable tersebut secara langsung

---

## Kode : Kata Kunci var

```
var name; // error
name = "Eko Kurniawan Khannedy";

var age = 30;
var address = "Indonesia";

System.out.println(name);
System.out.println(age);
System.out.println(address);
```

---

# Kata Kunci final

- Secara default, variable di Java bisa diubah-ubah nilainya
- Jika kita ingin membuat sebuah variable yang datanya tidak boleh diubah setelah pertama kali dibuat, kita bisa menggunakan kata kunci final
- Istilah variabel seperti ini, banyak juga yang menyebutnya konstan



## Kode : Kata Kunci final

```
final String name = "Eko Kurniawan Khannedy";
var age = 30;
var address = "Indonesia";

name = "Nama Diubah"; // error

System.out.println(name);
System.out.println(age);
System.out.println(address);
```

---

# Tipe Data Bukan Primitif

---

# Tipe Data Bukan Primitif

- Tipe data primitif adalah tipe bawaan di dalam bahasa pemrograman. Tipe data primitif tidak bisa diubah lagi
- Tipe data number, char, boolean adalah tipe data primitif. Tipe data primitif selalu memiliki default value
- Tipe data String bukan tipe data primitif, tipe data bukan primitif tidak memiliki default value, dan bisa bernilai null
- Tipe data bukan primitif bisa memiliki method/function (yang akan dibahas nanti)
- Di Java, semua tipe data primitif memiliki representasi tipe data bukan primitif nya

---

# Representasi Tipe Data Primitif (1)

| Tipe Data Primitif | Tipe Data Bukan Primitif |
|--------------------|--------------------------|
| byte               | Byte                     |
| short              | Short                    |
| int                | Integer                  |
| long               | Long                     |
| float              | Float                    |
| double             | Double                   |

---

## Representasi Tipe Data Primitif (2)

| Tipe Data Primitif | Tipe Data Bukan Primitif |
|--------------------|--------------------------|
| char               | Character                |
| boolean            | Boolean                  |

---

## Kode : Tipe Data Bukan Primitif

```
Integer iniInteger = 10;
Long iniLong = 10L;
Boolean iniBoolean = true;

Short iniShort; // null
iniShort = 100;

}
```

---

# Kode : Konversi Dari Tipe Primitif

```
int age = 30;  
  
Integer ageObject = age;  
  
int ageAgain = ageObject;  
  
short shortAge = ageObject.shortValue();  
byte byteAge = ageObject.byteValue();  
  
}
```

---

# Tipe Data Array

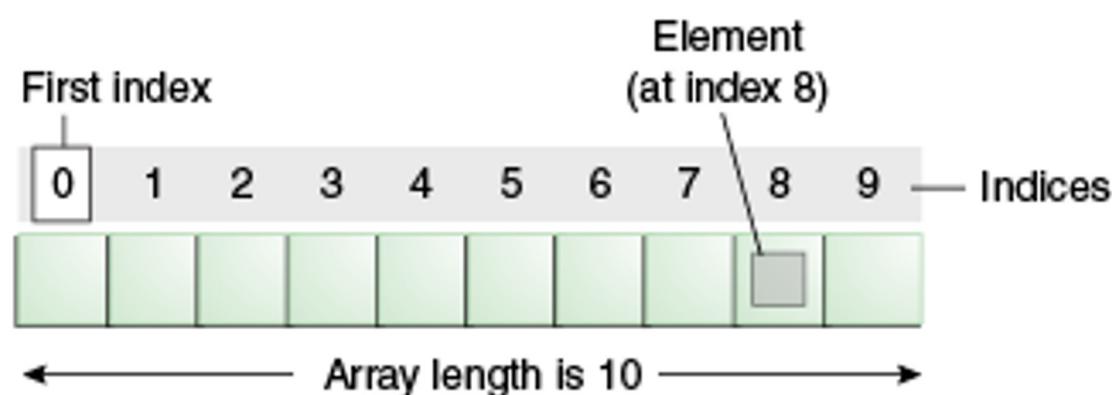
---

# Tipe Data Array

- Array adalah tipe data yang berisikan kumpulan data dengan tipe yang sama
- Jumlah data di Array tidak bisa berubah setelah pertama kali dibuat

---

## Cara Kerja Array



---

## Kode : Membuat Array

```
String[] arrayString;  
arrayString = new String[3];  
arrayString[0] = "Eko";  
arrayString[1] = "Kurniawan";  
arrayString[2] = "Khannedy";
```

```
}
```

```
}
```

---

# Kode : Array Initializer

```
int[] arrayInt = new int[]{  
    10, 90, 80, 67, 29  
};  
  
long[] arrayLong = {  
    10, 90, 80, 67, 29  
};  
}  
}
```

---

# Operasi di Array

| Operasi Array        | Keterangan              |
|----------------------|-------------------------|
| array[index]         | Mengambil data di array |
| array[index] = value | Mengubah data di array  |
| array.length         | Mengambil panjang array |



# Kode : Operasi di Array

```
long[] arrayLong = {  
    10, 90, 80, 67, 29  
};  
  
arrayLong[0] = 100;  
  
System.out.println(arrayLong[0]);  
System.out.println(arrayLong[1]);  
System.out.println(arrayLong.length);
```

---

# Kode : Array di dalam Array

```
String[][] members = {
    {"Eko", "Kurniawan", "Khannedy"},
    {"Budi", "Nugraha"},
    {"Joko", "Morro"},
};

String[] member1 = members[0];
System.out.println(member1[0]);

System.out.println(members[1][0]);
System.out.println(members[2][0]);
```

---

# Operasi Matematika



# Operasi Matematika

| Operator | Keterangan     |
|----------|----------------|
| +        | Penjumlahan    |
| -        | Pengurangan    |
| *        | Perkalian      |
| /        | Pembagian      |
| %        | Sisa Pembagian |

---

# Kode : Operasi Matematika

```
int a = 100;  
int b = 10;  
  
System.out.println(a + b);  
System.out.println(a - b);  
System.out.println(a * b);  
System.out.println(a / b);  
System.out.println(a % b);  
  
}
```

---

# Augmented Assignments

| Operasi Matematika | Augmented Assignments |
|--------------------|-----------------------|
| $a = a + 10$       | $a += 10$             |
| $a = a - 10$       | $a -= 10$             |
| $a = a * 10$       | $a *= 10$             |
| $a = a / 10$       | $a /= 10$             |
| $a = a \% 10$      | $a \%= 10$            |



# Kode : Augmented Assignments

```
int c = 100;  
  
c += 10;  
System.out.println(c);  
  
c -= 10;  
System.out.println(c);  
  
c *= 10;  
System.out.println(c);
```



# Unary Operator

| Operator        | Keterangan        |
|-----------------|-------------------|
| <code>++</code> | $a = a + 1$       |
| <code>--</code> | $a = a - 1$       |
| <code>-</code>  | Negative          |
| <code>+</code>  | Positive          |
| <code>!</code>  | Boolean kebalikan |

---

# Kode : Unary Operator

```
int d = +100;
int e = -10;

d++;
System.out.println(d);

d--;
System.out.println(d);

System.out.println(!true);
}
```

---

# Operasi Perbandingan

---

# Operasi Perbandingan

- Operasi perbandingan adalah operasi untuk membandingkan dua buah data
- Operasi perbandingan adalah operasi yang menghasilkan nilai boolean (benar atau salah)
- Jika hasil operasinya adalah benar, maka nilainya adalah true
- Jika hasil operasinya adalah salah, maka nilainya adalah false

---

# Operator Perbandingan

| Operator | Keterangan              |
|----------|-------------------------|
| >        | Lebih Dari              |
| <        | Kurang Dari             |
| $\geq$   | Lebih Dari Sama Dengan  |
| $\leq$   | Kurang Dari Sama Dengan |
| $=$      | Sama Dengan             |
| $\neq$   | Tidak Sama Dengan       |

---

# Kode : Operasi Perbandingan

```
int value1 = 100;  
int value2 = 100;  
  
System.out.println(value1 > value2);  
System.out.println(value1 < value2);  
System.out.println(value1 >= value2);  
System.out.println(value1 <= value2);  
System.out.println(value1 == value2);  
System.out.println(value1 != value2);
```



---

# Operasi Boolean

---

# Operasi Boolean

| Operator | Keterangan |
|----------|------------|
| &&       | Dan        |
|          | Atau       |
| !        | Kebalikan  |

---

# Operasi &&

| Nilai 1 | Operator | Nilai 2 | Hasil |
|---------|----------|---------|-------|
| true    | &&       | true    | true  |
| true    | &&       | false   | false |
| false   | &&       | true    | false |
| false   | &&       | false   | false |

---

# Operasi ||

| Nilai 1 | Operator | Nilai 2 | Hasil |
|---------|----------|---------|-------|
| true    |          | true    | true  |
| true    |          | false   | true  |
| false   |          | true    | true  |
| false   |          | false   | false |

---

# Operasi !

| Operator | Nilai 2 | Hasil |
|----------|---------|-------|
| !        | true    | false |
| !        | false   | true  |

---

# Kode : Operasi Boolean

```
var absen = 70;
var nilaiAkhir = 80;

var lulusAbsen = absen >= 75;
var lulusNilaiAkhir = nilaiAkhir >= 75;

var lulus = lulusAbsen && lulusNilaiAkhir;
System.out.println(lulus);

}
```

---

# **Expression, Statement & Block**

---

# Expression

- Expression adalah konstruksi dari variabel, operator dan pemanggilan method yang mengevaluasi menjadi sebuah single value
- Expression adalah core component dari statement



# Kode : Expression

```
int value;  
value = 10;  
  
System.out.println(value = 100);  
}  
}
```

---

# Statement

- Statement bisa dibilang adalah kalimat lengkap dalam bahasa.
- Sebuah statement berisikan execution komplit, biasanya diakhiri dengan titik koma
- Ada beberapa jenis statement :
  - Assignment expression
  - Penggunaan ++ dan --
  - Method invocation
  - Object creation expression



# Kode : Statement

```
// assignment statement
double aValue = 8933.234;
// increment statement
aValue++;
// method invocation statement
System.out.println("Hello World!");
// object creation statement
Date date = new Date();
```

---

# Block

- Block adalah kumpulan statement yang terdiri dari nol atau lebih statement.
- Block diawali dan diakhiri dengan kurung kurawal { }



# Kode : Block

```
▶ public class ExpressionApp {  
▶   public static void main(String[] args) {  
▶     double aValue = 8933.234;  
▶     aValue++;  
▶     System.out.println("Hello World!");  
▶     Date date = new Date();  
▶   }  
▶ }
```

---

# If Statement

---

# If Statement

- Dalam Java, if adalah salah satu kata kunci yang digunakan untuk percabangan
- Percabangan artinya kita bisa mengeksekusi kode program tertentu ketika suatu kondisi terpenuhi
- Hampir di semua bahasa pemrograman mendukung if expression

---

## Kode : If Statement

```
var nilai = 70;
var absen = 90;

if(nilai >= 75 && absen >= 75){
    System.out.println("Anda Lulus");
}

}
```

---

# Else Statement

- Blok if akan dieksekusi ketika kondisi if bernilai true
- Kadang kita ingin melakukan eksekusi program tertentu jika kondisi if bernilai false
- Hal ini bisa dilakukan menggunakan else expression

## Kode : Else Statement

```
var nilai = 70;
var absen = 90;

if (nilai >= 75 && absen >= 75) {
    System.out.println("Anda Lulus");
} else {
    System.out.println("Anda Tidak Lulus");
}
```

---

# Else If Statement

- Kada dalam If, kita butuh membuat beberapa kondisi
- Kasus seperti ini, di Java kita bisa menggunakan Else If expression
- Else if di Java bisa lebih dari satu

---

## Kode : Else If Statement

```
if (nilai >= 80 && absen >= 80) {  
    System.out.println("Nilai Anda A");  
} else if (nilai >= 70 && absen >= 70) {  
    System.out.println("Nilai Anda B");  
} else if (nilai >= 60 && absen >= 60) {  
    System.out.println("Nilai Anda C");  
} else if (nilai >= 50 && absen >= 50) {  
    System.out.println("Nilai Anda D");  
} else {  
    System.out.println("Nilai Anda E");  
}
```

---

# Switch Statement

---

# Switch Statement

- Kadang kita hanya butuh menggunakan kondisi sederhana di if statement, seperti hanya menggunakan perbandingan ==
- Switch adalah statement percabangan yang sama dengan if, namun lebih sederhana cara pembuatannya
- Kondisi di switch statement hanya untuk perbandingan ==

# Kode : Switch Statement

```
switch (nilai) {  
    case "A":  
        System.out.println("Wow Anda Lulus Dengan Baik");  
        break;  
    case "B":  
    case "C":  
        System.out.println("Anda Lulus");  
        break;  
    case "D":  
        System.out.println("Anda Tidak Lulus");  
        break;  
    default:  
        System.out.println("Mungkin Anda Salah Jurusan");  
}
```

---

# Switch Lambda

- Di Java versi 14, diperkenalkan switch expression dengan lambda
- Ini lebih mempermudah saat penggunaan switch expression karena kita tidak perlu lagi menggunakan kata kunci break

---

## Kode : Switch Lambda

```
var nilai = "A";

switch (nilai) {
    case "A" -> System.out.println("Wow Anda Lulus Dengan Baik");
    case "B", "C" -> System.out.println("Anda Lulus");
    case "D" -> System.out.println("Anda Tidak Lulus");
    default -> {
        System.out.println("Mungkin Anda Salah Jurusan");
    }
}
```

---

# Kata Kunci yield

- Di Java 14, ada kata kunci baru yaitu yield, dimana kita menggunakan kata kunci yield untuk mengembalikan nilai pada switch statement
- Ini sangat mempermudah kita ketika butuh membuat data berdasarkan kondisi switch statement

---

## Kode : Switch Tanpa yield

```
var nilai = "A";
String ucapan;

switch (nilai) {
    case "A" -> ucapan = "Wow Anda Lulus Dengan Baik";
    case "B", "C" -> ucapan = "Anda Lulus";
    case "D" -> ucapan = "Anda Tidak Lulus";
    default -> ucapan = "Mungkin Anda Salah Jurusan";
}

System.out.println(ucapan);
```

---

# Kode : Switch Dengan yield

```
var nilai = "A";
String ucapan = switch (nilai) {
    case "A":
        yield "Wow Anda Lulus Dengan Baik";
    case "B", "C":
        yield "Anda Lulus";
    case "D":
        yield "Anda Tidak Lulus";
    default:
        yield "Mungkin Anda Salah Jurusan";
};
System.out.println(ucapan);
```

---

# Ternary Operator

---

# Ternary Operator

- Ternary operator adalah operator sederhana dari if statement
- Ternary operator terdiri dari kondisi yang dievaluasi, jika menghasilkan true maka nilai pertama diambil, jika false, maka nilai kedua diambil

---

## Kode : Tanpa Ternary Operator

```
var nilai = 75;
String ucapan;

if (nilai >= 75) {
    ucapan = "Selamat Anda Lulus";
} else {
    ucapan = "Silahkan Coba Lagi";
}

}
```

---

# Kode : Dengan Ternary Operator

```
var nilai = 75;  
String ucapan = nilai >= 75 ? "Selamat Anda Lulus" : "Silahkan Coba Lagi";
```

```
System.out.println(ucapan);
```

```
}
```

---

# For Loop

---

# For Loop

- For adalah salah satu kata kunci yang bisa digunakan untuk melakukan perulangan
- Blok kode yang terdapat di dalam for akan selalu diulangi selama kondisi for terpenuhi

---

# Sintak Perulangan For

```
for(init statement; kondisi; post statement){  
    // block perulangan  
}
```

- Init statement akan dieksekusi hanya sekali di awal sebelum perulangan
- Kondisi akan dilakukan pengecekan dalam setiap perulangan, jika true perulangan akan dilakukan, jika false perulangan akan berhenti
- Post statement akan dieksekusi setiap kali diakhiri perulangan
- Init statement, Kondisi dan Post Statement tidak wajib diisi, jika Kondisi tidak diisi, berarti kondisi selalu bernilai true

---

# Kode : Perulangan Tanpa Henti

```
for(;;){  
    System.out.println("Perulangan Tanpa Henti");  
}  
}
```

---

# Kode : Perulangan Dengan Kondisi

```
var counter = 1;

for (; counter <= 10; ) {
    System.out.println("Perulangan Ke-" + counter);
    counter++;
}

}
```

---

# Kode : Perulangan Dengan Init Statement

```
for (var counter = 1; counter <= 10; ) {  
    System.out.println("Perulangan Ke-" + counter);  
    counter++;  
}  
}  
}
```



# Kode : Perulangan Dengan Post Statement

```
for (var counter = 1; counter <= 10; counter++) {  
    System.out.println("Perulangan Ke-" + counter);  
}  
}  
}
```

---

# While Loop

---

# While Loop

- While loop adalah versi perulangan yang lebih sederhana dibanding for loop
- Di while loop, hanya terdapat kondisi perulangan, tanpa ada init statement dan post statement



# Kode : While Loop

```
var counter = 1;

while (counter <= 10) {
    System.out.println("Perulangan Ke-" + counter);
    counter++;
}

}
```

---

# Do While Loop

---

## Do While Loop

- Do While loop adalah perulangan yang mirip dengan while
- Perbedaannya hanya pada pengecekan kondisi
- Pengecekan kondisi di while loop dilakukan di awal sebelum perulangan dilakukan, sedangkan di do while loop dilakukan setelah perulangan dilakukan
- Oleh karena itu dalam do while loop, minimal pasti sekali perulangan dilakukan walaupun kondisi tidak bernilai true

---

# Kode : Do While Loop

```
var counter = 100;

do {
    System.out.println("Perulangan Ke-" + counter);
    counter++;
} while (counter <= 10);

}
```

---

# Break & Continue

---

# Break & Continue

- Pada switch statement, kita sudah mengenal kata kunci break, yaitu untuk menghentikan case dalam switch
- Sama dengan pada perulangan, break juga digunakan untuk menghentikan seluruh perulangan.
- Namun berbeda dengan continue, continue digunakan untuk menghentikan perulangan saat ini, lalu melanjutkan ke perulangan selanjutnya



# Kode : Break

```
var counter = 1;
while (true) {
    System.out.println("Perulangan ke-" + counter);
    counter++;

    if (counter > 10) {
        break;
    }
}
```

---

## Kode : Continue

```
for (int counter = 1; counter <= 100; counter++) {  
    if (counter % 2 == 0) {  
        continue;  
    }  
  
    System.out.println("Perulangan Ganjil-" + counter);  
}  
}
```

---

# For Each

---

## For Each

- Kadang kita biasa mengakses data array menggunakan perulangan
- Mengakses data array menggunakan perulangan sangat bertele-tele, kita harus membuat counter, lalu mengakses array menggunakan counter yang kita buat
- Namun untungnya, di Java terdapat perulangan for each, yang bisa digunakan untuk mengakses seluruh data di Array secara otomatis

---

# Kode : Tanpa For Each

```
String[] array = {
    "Eko", "Kurniawan", "Khannedy",
    "Programmer", "Zaman", "Now"
};

for (int i = 0; i < array.length; i++) {
    System.out.println(array[i]);
}

}
```

---

## Kode : For Each

```
String[] array = {
    "Eko", "Kurniawan", "Khannedy",
    "Programmer", "Zaman", "Now"
};

for (var value : array) {
    System.out.println(value);
}

}
```

---

# Method

---

# Method

- Method adalah block kode program yang akan berjalan saat kita panggil
- Sebelumnya kita sudah menggunakan method `println()` untuk menampilkan tulisan di console
- Untuk membuat method di Java, kita bisa menggunakan kata kunci `void`, lalu diikuti dengan nama method, kurung () dan diakhiri dengan block
- Kita bisa memanggil method dengan menggunakan nama method lalu diikuti dengan kurung ()
- Di bahasa pemrograman lain, Method juga disebut dengan Function

---

# Kode : Method

```
public static void main(String[] args) {
    sayHelloWorld();
}

static void sayHelloWorld(){
    System.out.println("Hello World");
}

}
```

---

# Method Parameter

---

# Method Parameter

- Kita bisa mengirim informasi ke method yang ingin kita panggil
- Untuk melakukan hal tersebut, kita perlu menambahkan parameter atau argument di method yang sudah kita buat
- Cara membuat parameter sama seperti cara membuat variabel
- Parameter ditempatkan di dalam kurung () di deklarasi method
- Parameter bisa lebih dari satu, jika lebih dari satu, harus dipisah menggunakan tanda koma



# Kode : Method Parameter

```
public static void main(String[] args) {
    sayHello("Eko", "Khannedy");
}

static void sayHello(String firstName, String lastName) {
    System.out.println("Hello " + firstName + " " + lastName);
}
```

---

# Method Return Value

---

# Method Return Value

- Secara default, method itu tidak menghasilkan value apapun, namun jika kita ingin, kita bisa membuat sebuah method mengembalikan nilai
- Agar method bisa menghasilkan value, kita harus mengubah kata kunci void dengan tipe data yang dihasilkan
- Dan di dalam block method, untuk menghasilkan nilai tersebut, kita harus menggunakan kata kunci return, lalu diikuti dengan data yang sesuai dengan tipe data yang sudah kita deklarasikan di method
- Di Java, kita hanya bisa menghasilkan 1 data di sebuah method, tidak bisa lebih dari satu

---

# Kode : Method Return Value

```
public static void main(String[] args) {
    var a = 100;
    var b = 200;
    var c = sum(a, b);

    System.out.println(c);
}

static int sum(int value1, int value2) {
    var total = value1 + value2;
    return total;
}
```

---

# Method Variable Argument

---

# Method Variable Argument

- Kadang kita butuh mengirim data ke method sejumlah data yang tidak pasti
- Biasanya, agar bisa seperti ini, kita akan menggunakan Array sebagai parameter di method tersebut
- Namun di Java, kita bisa menggunakan variable argument, untuk mengirim data yang berisi jumlah tak tentu, bisa nol atau lebih
- Parameter dengan tipe variable argument, hanya bisa ditempatkan di posisi akhir parameter

---

# Kode : Tanpa Variable Argument

```
static void sayCongrats(String name, int[] values) {  
    int total = 0;  
    for (var value : values) {  
        total += value;  
    }  
    int finalValue = total / values.length;  
  
    if (finalValue >= 75) {  
        System.out.println("Selamat " + name + ", Anda Lulus");  
    } else {  
        System.out.println("Maaf " + name + ", Anda Lulus");  
    }  
}
```

# Kode : Dengan Variable Argument

```
public static void main(String[] args) {
    sayCongrats( name: "Eko", ...values: 80, 90, 79, 48);
}

@ static void sayCongrats(String name, int... values) {
    int total = 0;
    for (var value : values) {
        total += value;
    }
    int finalValue = total / values.length;
    if (finalValue >= 75) {
        System.out.println("Selamat " + name + ", Anda Lulus");
    } else {
```

---

# Method Overloading

---

# Method Overloading

- Method overloading adalah kemampuan membuat method dengan nama yang sama lebih dari sekali.
- Namun ada ketentuannya, yaitu data parameter di method tersebut harus berbeda-beda, enta jumlah atau tipe data parameternya
- Jika ada yang sama, maka program Java kita akan error



# Kode : Method Overloading

```
static void sayHello() {  
    System.out.println("Hello");  
}  
  
static void sayHello(String firstName) {  
    System.out.println("Hello " + firstName);  
}  
  
static void sayHello(String firstName, String lastName) {  
    System.out.println("Hello " + firstName + " " + lastName);  
}
```

---

# Recursive Method

---

# Recursive Method

- Recursive method adalah kemampuan method memanggil method dirinya sendiri
- Kadang memang ada banyak problem, yang lebih mudah diselesaikan menggunakan recursive method, seperti contohnya kasus factorial



# Kode : Factorial Loop

```
f
  static int factorial(int value) {
    var result = 1;
    for (int i = 1; i <= value; i++) {
      result *= i;
    }
    return result;
  }
```

---

# Kode : Factorial Recursive

```
static int factorialRecursive(int value) {  
    if (value == 1) {  
        return 1;  
    } else {  
        return value * factorialRecursive( value: value - 1);  
    }  
}
```

---

# Problem Dengan Recursive

- Walaupun recursive method itu sangat menarik, namun kita perlu berhati-hati
- Jika recursive terlalu dalam, maka akan ada kemungkinan terjadi error StackOverflow, yaitu error dimana stack method terlalu banyak di Java
- Kenapa problem ini bisa terjadi? Karena ketika kita memanggil method, Java akan menyimpannya dalam stack, jika method tersebut memanggil method lain, maka stack akan menumpuk terus, dan jika terlalu dalam, maka stack akan terlalu besar, dan bisa menyebabkan error StackOverflow

---

# Kode : Error StackOverflow

```
static void loop(int value) {
    if (value == 0) {
        System.out.println("Selesai");
    } else {
        System.out.println("Loop-" + value);
        loop( value: value - 1);
    }
}
```

---

# Scope

---

# Scope

- Di Java, variable hanya bisa diakses di dalam area dimana mereka dibuat.
- Hal ini disebut scope
- Contoh, jika sebuah variable dibuat di method, maka hanya bisa diakses di method tersebut, atau jika dibuat didalam block, maka hanya bisa diakses didalam block tersebut



# Kode : Scope

```
@ [ ] static void sayHello(String name) {  
    String hello = "Hello " + name;  
    if (!name.isBlank()) {  
        String hi = "Hi " + name;  
        System.out.println(hi);  
    }  
  
    System.out.println(hello);  
    System.out.println(hi); // error  
}  
}
```

---

# Komentar

---

# Komentar

- Kadang dalam membuat program, kita sering menempatkan komentar di kode program tersebut
- Komentar adalah kode program yang akan dihiraukan saat proses kompilasi, sehingga di binary code Java, tidak akan ada kode komentar tersebut
- Biasanya komentar digunakan untuk dokumentasi

# Kode : Komentar

```
/*
 * Menghitung jumlah a + b
 *
 * @param a nilai a
 * @param b nilai b
 * @return a + b
 */
static int sum(int a, int b) {
    // jumlahkan a + b
    return a + b;
}
```