

---

# 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

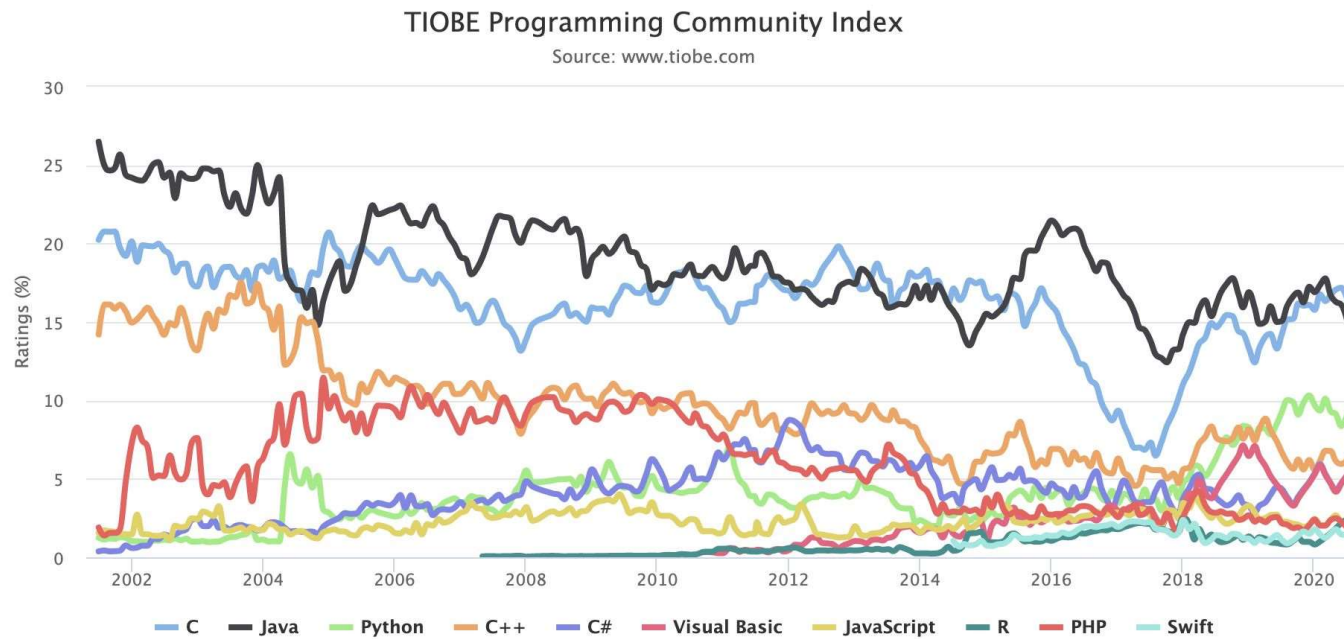


## Versi Java

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

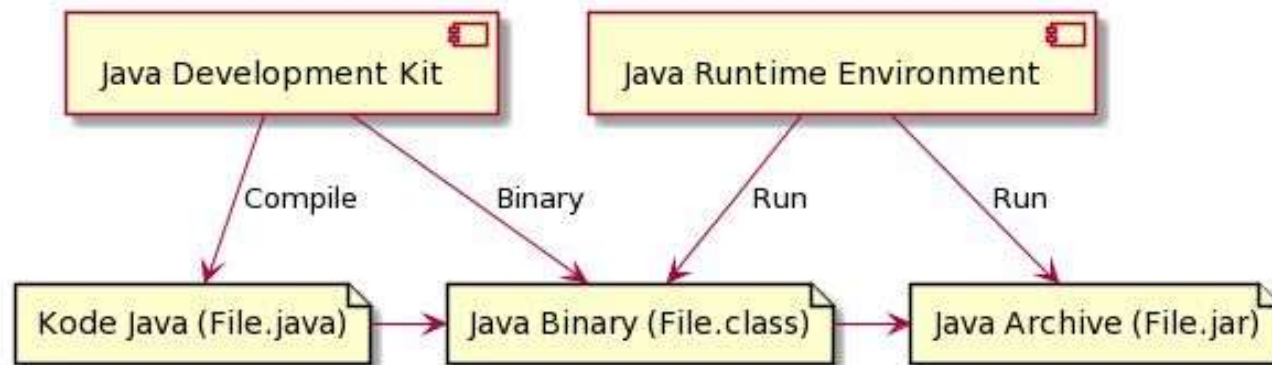


# Java Virtual Machine

- 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>

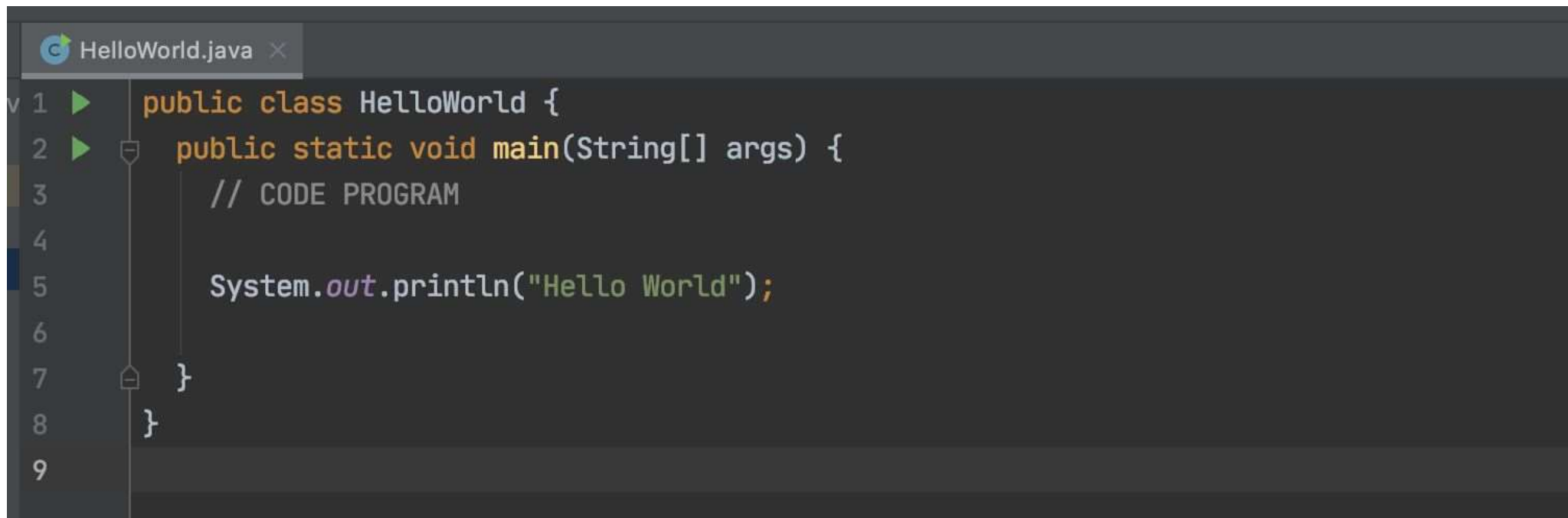
---

# Program Hello World





# Program Hello World



```
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
```



# 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



# Type 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	-9,223,372,036,854,775,808	9,223,372,036,854,775,807	8 bytes	0



## Kode : Integer Number

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

```
}  
}
```



# Floating Point Number

Type Data	Min	Max	Size	Default
float	$3.4\text{e-}038$	$3.4\text{e+}038$	4 bytes	0.0
double	$1.7\text{e-}308$	$1.7\text{e+}308$	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 teksnya
- 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();
```

```
}
```

---

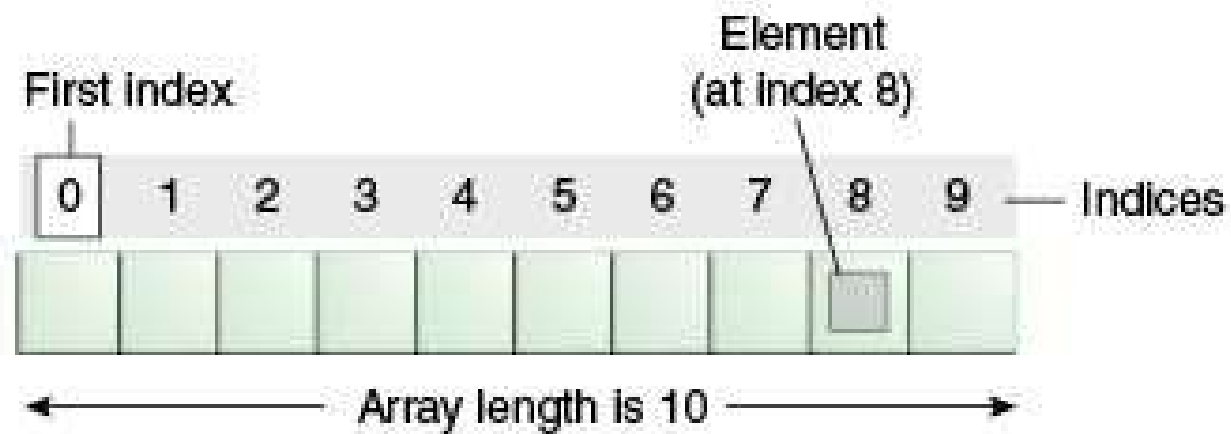
# Tipe Data Array



# 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
<code>array[index]</code>	Mengambil data di array
<code>array[index] = value</code>	Mengubah data di array
<code>array.length</code>	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
++	$a = a + 1$
--	$a = a - 1$
-	Negative
+	Positive
!	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
>=	Lebih Dari Sama Dengan
<=	Kurang Dari Sama Dengan
==	Sama Dengan
!=	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 komplet, 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 diakhir 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);  
}  
}
```