# Hello World

Hai Coders, selamat datang di interactive learning Java! Disini Kamu akan belajar pemrograman Java mulai dari yang paling dasar.

Di dalam pemrograman Java Kamu dapat melihat bahwa struktur kode Java selalu dibungkus dengan class. Selain itu bagian inti program akan dieksekusi di dalam method main().

Nama class boleh apa saja, hanya saja nama class harus sama dengan nama file. Misalkan nama class Main maka nama filenya harus Main.java.

File kode java berekstensi .java dan bila sudah di-*compile* akan menjadi .class, misalnya main.java dicompile menjadi main.class. Untuk mengkompilasi kode Java gunakan perintah javac namafile.java pada terminal. Untuk menjalankan bytecode Java gunakan perintah java NamaClass di terminal.

Pada tutorial pertama ini kita akan mencoba menampilkan output teks "Hello World".

## Praktek

* Silahkan eksekusi kode di editor sebelah kanan dengan mengklik tombol "Jalankan" dan lihat apa outputnya.
* Salin baris kode System.out.println("Hello World dari Java"); pada baris baru di dalam method main() dan ganti string di dalamnya dengan string lain yang Kamu suka.

**contoh**

class Main {

public static void main(String[] args){

System.out.println("Hello World dari Java");

}

}

Hello World dari Java

# Variabel di Java

Variabel adalah wadah yang diberi nama yang dapat kita isi dengan nilai dan dapat dimanipulasi. Sebelum dapat digunakan, variabel harus dideklarasikan terlebih dahulu.

tipedata namavariabel = nilai;

Bila hendak mendeklarasikan lebih dari satu variabel yang bertipe sama, pisahkan dengan tanda koma.

int a, b, c;

int a = 10, b = 10;

byte B = 22;

double pi = 3.14159;

char a = 'a';

Ada tiga jenis variabel di Java:

* Variabel lokal
* Variabel instance
* Variabel Class/Static

Variabel lokal adalah variabel yang dideklarasikan di dalam method atau blok. Variabel ini hanya dapat digunakan di dalam method tersebut. Variabel ini dibuat ketika method dipanggil, dan akan langsung dihapus setelah pemanggilan method selesai.

Pada pendeklarasian variabel lokal, kita harus menuliskan lengkap mulai dari tipe data, nama variabel dan juga nilai awalnya.

**Praktek**

* Pada editor terdapat class Test yang berisi method main() dengan satu buah variabel bertipe String. Jalankan kode dan lihat outputnya.
* Coba ganti nilai variabel pesan dengan namamu, lalu jalankan lagi kode program untuk melihat hasilnya.

public class Test {

public static void main(String args[]) {

String pesan = "Selamat Belajar";

System.out.println(pesan);

}

}

# Instance Variable

Instance variable adalah variabel yang dideklarasikan di dalam class, tetapi di luar method. Instance variable biasanya dibuat bila kita membutuhkan variabel yang akan digunakan oleh lebih dari satu method.

Berbeda dengan local variable, instance variable tidak mesti ditulis dengan nilai awal. Bila nilai awal tidak dideklarasikan, maka nilai default untuk instance variable adalah 0 bila bertipe integer, false bila bertipe boolean, dan null untuk referensi objek.

Instance variable dapat dipanggil langsung dengan namanya di dalam class, kecuali method static. Pada method static seperti main(), variabel dapat dipanggil melalui objeknya namaObjek.namaVariabel.

## Praktek

* Pada editor terdapat class Test dengan satu instance variable. Jalankan kode program untuk melihat outputnya.

public class Test {

String pesan = "Selamat Belajar";

public static void main(String args[]) {

Test tes = new Test();

System.out.println(tes.pesan);

}

}

OUTPUT PROGRAM

Selamat Belajar

# Class/Static Variable

Class variable atau dikenal juga dengan istilah static variable adalah variable yang dideklarasikan di dalam class di luar method. Sama seperti instance variable, class variable dibuat bila kita membutuhkan variabel yang digunakan di banyak method.

Perbedaan class variable dengan instance variable adalah, pada class variable kita menambahkan kata kunci static pada pendeklarasian.

static String pesan;

Class variable dapat dipanggil langsung di dalam class di dalam method, dan pada method static dapat dipanggil dengan menulis NamaClass.namaVariabel.

## Praktek

* Pada editor terdapat class Test dengan satu class variable. Jalankan kode dan lihat outputnya.
* Karena variabel pesan adalah class variable atau static variable, kita dapat memanggil langsung di dalam method main() tanpa harus membuat objeknya dahulu. Hapus deklarasi objek Test() di baris ke-5, dan ganti pemanggilan variabel di baris ke-6 menjadi Test.pesan. Jalankan kode dan lihat hasilnya.

public class Test {

static String pesan = "Selamat Belajar";

public static void main(String args[]) {

Test tes = new Test();

System.out.println(tes.pesan);

}

}

OUTPUT PROGRAM

Selamat Belajar

# Konstanta di Java

Konstanta pada pemrograman berfungsi mirip dengan variabel. Bedanya variabel dapat kita ubah isinya selama runtime, sedangkan konstanta tidak.

Di Java sebenarnya tidak ada term konstanta. Namun kita dapat membuat variabel yang bersifat seperti konstanta dengan menambahkan kata final pada deklarasi variabel sebelum menuliskan tipe data. Dan seperti konstanta umumnya, pada Java juga kita dianjurkan untuk menuliskan final variabel dengan menggunakan nama variabel menggunakan huruf kapital.

static final float PI = 3,14;

## Praktek

* Pada editor terdapat class Lingkaran dengan dua buah class variable radius dan PI. Jalankan kode dan lihat outputnya.
* PI pada kode adalah class variable dengan kata kunci final yang berarti ia berlaku sebagai konstanta, sehingga tidak dapat diubah sepanjang runtime. Pada baris ke-7 terdapat snippet untuk mencoba mengubah nilai konstanta PI. Hapus garis miring ganda di awal baris tersebut, dan jalankan kembali kode. Pesan error apa yang muncul?

class Lingkaran {

static int radius;

static final double PI = 3.14;

public static void main(String[] args){

// Lingkaran.PI = 3.14159;

Lingkaran.radius = 7;

double luas = (Lingkaran.radius \* Lingkaran.radius \* Lingkaran.PI);

System.out.println("Jari-jari: " + Lingkaran.radius);

System.out.println("Luas: " + luas);

}

}

OUTPUT PROGRAM

Jari-jari: 7

Luas: 153.86

# Tipe Data

Ada beberapa tipe data dasar yang wajib kamu ketahui bila ingin membuat sebuah aplikasi dengan bahasa pemrograman Java, diantaranya:

* **byte**, diisi dengan angka bulat bukan bilangan koma dengan rentang -128 hingga 127
* **short**, diisi dengan angka bulat bukan bilangan koma dengan rentang -32.767 hingga 32.767
* **int**, diisi dengan angka bulat bukan bilangan koma dengan rentang -2.147.483.648 hingga 2.147.483.647
* **long**, diisi dengan angka bulat bukan bilangan koma dengan rentang -9.223.372.036.854.775.808 hingga 9.223.372.036.854.775.807
* **float**, diisi dengan bilangan koma dengan rentang 32-bit IEEE 754 floating point
* **double**, diisi dengan bilangan koma dengan rentang 64-bit IEEE 754 floating point
* **boolean**, diisi dengan **true** dan **false** saja. Selain itu akan error
* **String**, diisi dengan rangkaian alfanumerik dan tanda baca yang diapit dengan tanda petik dua (")
* **char**, diisi dengan alfanumerik yang diapit dengan tanda petik satu (')

Kamu dapat membuat sebuah variabel di dalam method yang tentu saja akan menjadi variabel lokal di dalam method tersebut. Atau menjadikannya atribut suatu kelas, namun harus menginstansiasi atau membuat objeknya dahulu bila ingin menggunakan variabel tersebut.

Setelah membuat variabel, kamu dapat melakukan berbagai operasi seperti operasi aritmatika dan menampilkannya ke konsol.

## Praktek

* Jalankan kode di editor dan perhatikan outputnya.
* Buat variabel baru dengan tipe data yang kamu suka dan isi dengan nilai. Lalu amati output yang dihasilkan.
* Ganti nilai variabel z menjadi "10000000" (menggunakan kutip), lalu jalankan kode. Pesan error apa yang muncul pada output?
* Ganti nilai variabel pesan menjadi 200, lalu jalankan kode. Pesan error apa yang muncul pada output?

1

class Main {

2

​

3

public static void main(String[] args){

4

byte x = 100;

5

short y = 1000;

6

int z = 10000000;

7

double pi = 3.14;

8

String pesan = "Lorem ipsum sit dolor amet";

9

char nilai = 'A';

10

​

11

System.out.println(x);

12

System.out.println(y);

13

System.out.println(z);

14

System.out.println(pi);

15

System.out.println(pesan);

16

System.out.println(nilai);

17

}

18

}

JALANKAN KODE

 OUTPUT PROGRAM

100

1000

10000000

3.14

Lorem ipsum sit dolor amet

A

# Operator Relasional

Operator relasional atau perbandingan digunakan untuk membandingkan dua buah nilai atau variabel dan akan menghasilkan **true** dan **false**.

Operator relasional yang harus kamu ingat antara lain:

* == sama dengan (tanda = digunakan untuk mengisikan nilai pada variabel)
* != tidak sama dengan
* > lebih dari
* < kurang dari
* >= lebih dari sama dengan
* <= kurang dari sama dengan

Bila Kamu sudah pernah mempelajari operator perbandingan di matematika, Kamu sudah tidak akan asing lagi dengan operator di atas.

## Praktek

* Silahkan eksekusi kode di sebelah kanan dan perhatikan output yang akan dihasilkan.
* Ganti operator pada kode di baris ke-9 dengan !=, >, <, >= dan <= dan pelajari output dari setiap perubahan operatornya.
* Ganti nilai variabel x dan y dengan nilai yang sama, lalu cobakan lagi setiap operatornya dan lihat outputnya.

class Relasional {

3

public static void main(String[] args){

4

// definisi variabel

5

int x = 10;

6

int y = 15;

7

​

8

// menggunakan operator relasional

9

System.out.println(x == y);

10

}

11

}

JALANKAN KODE

 OUTPUT PROGRAM

false

# Operator Logika

Operator logika membandingkan dua pernyataan kondisi. Penggunaannya biasa digabungkan dengan operator relasional yang sudah dibahas sebelumnya. Fungsinya untuk membentuk paduan kondisi yang akan membuat sebuah aturan yang cukup kompleks.

Operator logika yang harus kamu ingat antara lain:

* && artinya **dan**, dimana kondisi bernilai benar atau true hanya jika kedua operand bernilai **true**. Bila salah satu atau kedua operand bernilai **false**, maka nilai kondisi menjadi **false**
* || artinya **atau**, dimana kondisi bernilai benar atau true jika salah satu dari kedua operand bernilai **true**. Bila kedua operand bernilai **false**, maka kondisi akan bernilai **false**
* ! artinya **negaSI**, dimana nilai variabel akan dibalik menjadi lawannya, misal **!true** berarti **false** dan **!false** berarti **true**

Bila kamu terbiasa dengan logika matematika, hal ini bukan hal yang sulit.

## Praktek

* Silahkan eksekusi kode di sebelah kanan dan lihat output yang akan dihasilkan
* Buatlah variabel dengan kasusmu sendiri dengan tipe data yang kamu suka dan coba tampilkan isi variabel yang berisi operator logika tersebut dengan menggunakan **System.out.println()**

1

class Logika {

2

​

3

public static void main(String[] args){

4

// definisi variabel

5

int x = 10;

6

int y = 15;

7

​

8

// menggunakan operator relasional

9

System.out.println( (x == y) || (x != y) ); // false || true

10

System.out.println( (x > y) && (x < y) ); // false && true

11

System.out.println( !(x >= y) ); // !false

12

}

13

}

JALANKAN KODE

 OUTPUT PROGRAM

true

false

true

# Operator Aritmatika

Di Java kamu dapat melakukan operasi aritmatika dasar seperti kali, bagi, tambah, kurang. Beberapa operator aritmatika tersebut antara lain:

* + untuk penjumlahan
* \* untuk pengalian
* - untuk pengurangan
* / untuk pembagian
* % untuk modulus (sisa bagi)
* ++ untuk menaikkan nilai sebanyak satu
* -- untuk menurunkan nilai sebanyak satu

Jika dua buah variabel berisi angka, maka operator aritmatika dapat diterapkan. Sekarang mari kita coba buat kode yang menggunakan operator aritmatika.

## Praktek

* Silahkan jalankan kode di sebelah kanan dan lihat outputnya
* Silahkan buat variabel kamu sendiri dan terapkan salah satu operator diatas terhadpa variabel yang kamu buat dan jalankan kodenya

class Aritmatika {

2

​

public static void main(String[] args){

// definisi variabel

int x = 10;

int y = 15;

int result = 0;

8

System.out.println("x: " + x);

System.out.println("y: " + y);

11

​

// menggunakan operator +

result = x + y;

System.out.println("x + y = " + result);

15

​

// menggunakan operator -

result = x - y;

System.out.println("x - y = " + result);

19

// menggunakan operator \*

result = x \* y;

System.out.println("x \* y = " + result);

23

// menggunakan operator /

result = x / y;

JALANKAN KODE

 OUTPUT PROGRAM

x: 10

y: 15

x + y = 25

x - y = -5

x \* y = 150

x / y = 0

x % y = 5

x setelah dinaikkan: 11

y setelah diturunkan: 14

# Operator Bitwise

Operator bitwise digunakan untuk menangani operasi bit pada suatu bilangan dalam representasi biner. Operator bitwise yang harus kamu ingat antara lain:

* & melakukan operasi AND terhadap dua bilangan biner
* | melakukan operasi OR terhadap dua bilangan biner
* ^ melakukan operasi XOR terhadap dua bilangan biner
* ~ melakukan operasi negasi terhadap bilangan biner
* >> n melakukan operasi penggeseran bit ke kanan terhadap bilangan biner sebanyak n bit
* << n melakukan operasi penggeseran bit ke kiri terhadap bilangan biner sebanyak n bit

Misalkan bilangan x = 60 yang dalam biner adalah 0011 1100, dan y = 13 yang dalam biner adalah 0000 1101, maka:

x & y

0011 1100

0000 1101

---------

0000 1100

x | y

0011 1100

0000 1101

---------

0011 1101

x ^ y

0011 1100

0000 1101

---------

0011 0001

~x

0011 1100

---------

1100 0011

x >> 2

0011 1100

---------

0000 1111

x << 2

0011 1100

---------

1111 0000

## Praktek

* Silahkan eksekusi kode di editor dan perhatikan output dari setiap operasi
* Buatlah variabel dengan kasusmu sendiri dengan tipe data yang kamu suka dan coba tampilkan isi variabel yang menggunakan operator bitwise dengan menggunakan **System.out.println()**

class Bitwise {

2

public static void main(String[] args){

4

// definisi variabel

int x = 60;

int y = 13;

8

// menggunakan operator &

System.out.println(x & y);

11

// menggunakan operator |

System.out.println(x | y);

// menggunakan operator ^

System.out.println(x ^ y);

​

// menggunakan operator ~

System.out.println(~ x);

19

// menggunakan operator >>

System.out.println(x >> 2);

​

// menggunakan operator <<

System.out.println(x << 2);

 OUTPUT PROGRAM

12

61

49

-61

15

240

# Operator Instanceof

Untuk memeriksa apakah suatu objek merupakan hasil instansiasi dari suatu class, kamu dapat menggunakan **instanceof**.

namaObjek instanceof NamaClass;

Kondisi akan menghasilkan nilai **true** bila namaObjek adalah benar hasil instansiasi dari class namaClass, dan menghasilkan nilai **false** bila namaObjek bukan hasil instansiasi dari class namaClass.

Operator ini berguna bila suatu saat kamu ingin memeriksa suatu objek sebelum melakukan operasi selanjutnya.

## Praktek

* Silahkan eksekusi kode di editor dan perhatikan output yang akan dihasilkan

class Message {

2

String pesan = "Halo Coders";

3

public static void main(String[] args){

5

Message tes = new Message();

System.out.println(tes instanceof Message);

System.out.println(tes.pesan instanceof String);

}

}

JALANKAN KODE

 OUTPUT PROGRAM

true

true