

Hands on Java for the First Time

Silver - Chapter 1 - Topic 2

Selamat datang di **Chapter 1 Topic 2**
online course **Back End Java** dari
Binar Academy!



Hello, World 🦋

Hello world banget nggak, nih. Pada topik ini kita bakal membahas Hello world secara lebih mendalam. Iya, kita bakal belajar tentang Hello world beneran.

Supaya pembelajaran Back End Java ini memberikan pengenalan yang menyeluruh, kamu bisa intip slides peta materi untuk lebih familiar dengan istilah baru yang akan ditemui. Yaudin, langsung aja kita kepoin~



Detailnya, kita bakal bahas hal-hal berikut ini:

- Hello World dengan Notepad dan IDE
- Java Program Structure
- Pengetahuan Dasar Method
- Konsep Java Syntax dan Convention



Hello, World!

Iya bener, di peta materi kita akan membahas **Hello World**.

Bukan seperti sapaan yang kita pakai di awal slides, Hello world punya arti berbeda di Back End Java.



Yang bukan sembarang sapaan~

Hello World adalah sebuah program yang akan menampilkan **pesan bertuliskan “Hello World!”** ke layar komputer Back End Engineer.

Biasanya muncul pas Back End Engineer baru banget set bahasa pemrograman yang digunakan.

Kalimat Hello World ini nggak hanya ada di Indonesia aja, tapi hampir di seluruh dunia juga menggunakan pesan ini. Iya, Jungkook sama Adam Levine juga pake.



Lalu, gimana caranya bikin Hello World?

Ibarat mau naik helikopter untuk menjemput idaman kita, pasti kita punya opsi jalan yang bisa dipilih, kan?

Bisa lewat jalan utama atau jalan alternatif, yang penting tujuannya bisa sampai.

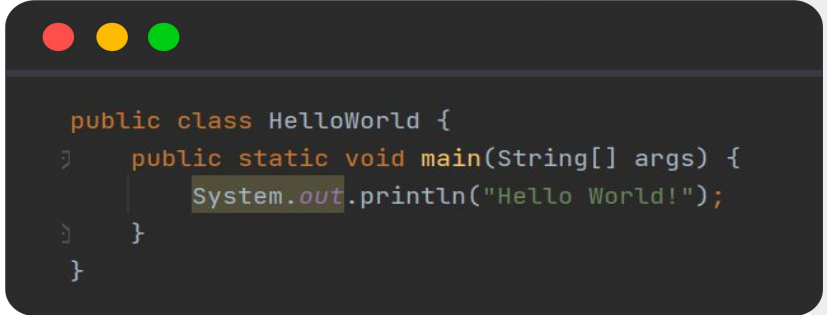
Begitu juga dengan Hello World. Untuk membuat Hello World menjadi nyata, ada beberapa tools yang bisa kita gunakan, diantaranya **Notepad** dan **IntelliJ**.



4. Selanjutnya, buka file Helloworld.java yang telah disimpan.
5. Pastikan JAVA_HOME udah terinstall.
6. Jalankan dua command berikut.
 - **Java HelloWorld.java**
 - **Java Hello World**

```
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hello World!");  
    }  
}
```

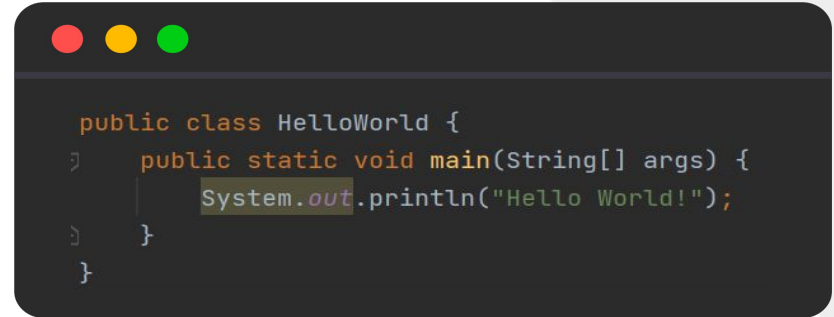

7. Kalau command udah diset, lanjut periksa apakah ada yang error.
8. Jika sudah berhasil, coba **ganti kata "Hello World!"** dengan kata lain dan jalankan command ke-2 (Java Hello World) tanpa menjalankan command ke-1



```
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hello World!");  
    }  
}
```

9. Coba **edit kode** dengan Notepad, **kurangi semicolon** (tanda titik koma) pada statement, kemudian **compile file** tersebut.
10. Selesai!

Jadi, apa yang bisa kita simpulkan dari kedua command tersebut? Coba deh kamu renungkan.

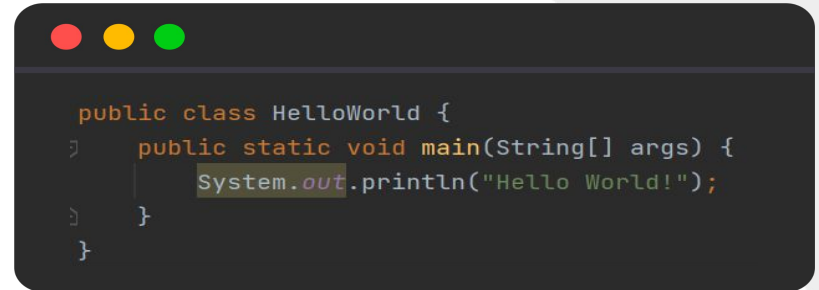


```
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hello World!");  
    }  
}
```

IntelliJ

Selain Notepad, kalau kamu mau buat Hello World pakai IntelliJ juga bisa, lho. Langkah-langkahnya, yaitu:

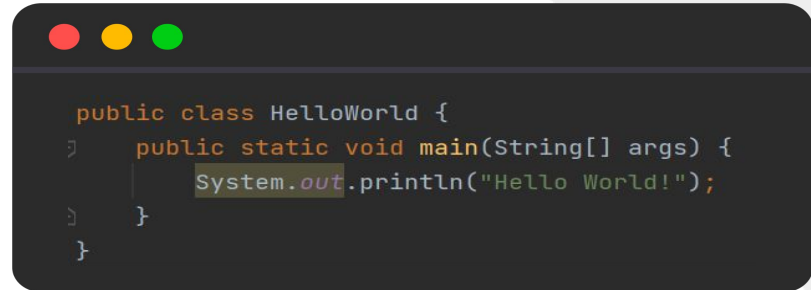
1. Buka IntelliJ sebagai **IDE** buat menjalankan program HelloWorld.
2. Buat New Project dengan Java.
3. Buat Class baru dengan nama **HelloWorld**. Udah ditulis namanya? Kita move ke langkah 4.



```
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hello World!");  
    }  
}
```

3. Tulis kode sesuai gambar di samping.
4. Jalankan kodenya.
5. Selesai!

Jadi, apa yang kamu rasakan ketika menuliskan kode pada IDE? Apa keuntungannya?



```
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hello World!");  
    }  
}
```

Setelah mengetahui tahapan dari dua tools, sekarang saatnya kita mengetahui proses dari pemrograman~



1. Penulisan program menggunakan editor yang menghasilkan file berformat **.java**
2. Proses kompilasi dengan command '**javac nama_file.java**' menghasilkan file berformat **.class**
3. Proses eksekusi program dilakukan dengan command '**java nama_file**'

Lebih lanjut tentang HelloWorld!, ada juga yang namanya **Java Program Structure**.

Apa tuh kaitan HelloWorld dan Java Program Structure? Iya, setelah ini kita berangkat langsung jelasin 🏃





```
public class HelloWorld {  
  
}
```

Supaya pemahaman kita terhadap HelloWorld makin ajib, kita harus tahu dulu struktur dari Java Program!

Di samping kita ada struktur yang didalamnya punya beberapa kata dengan warna berbeda. Struktur yang pertama dibahas, yaitu:

Class yang berupa HelloWorld.

Method yang berupa main

```
public static void main(String[] args) {  
    System.out.println("Hello World!");  
}
```

Method parameter
berupa args dengan tipe data Array of string

Kata '**System.out....**' berarti sebuah **statement** yang harus selalu diakhiri dengan semicolon (;)

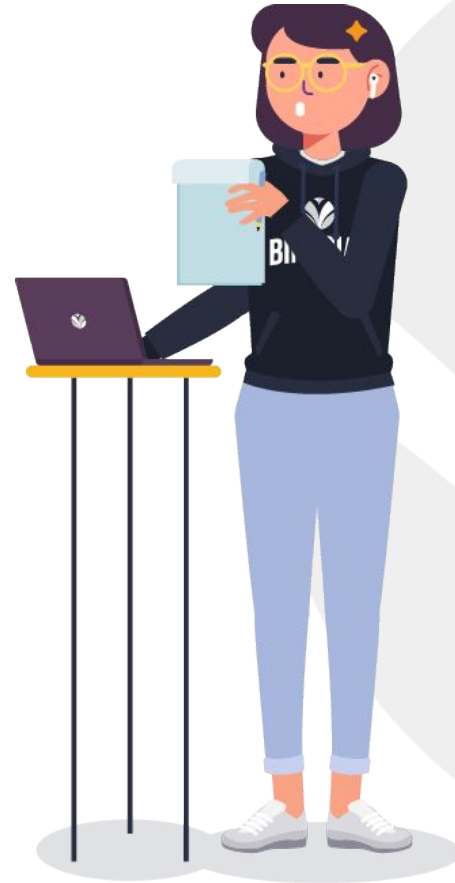
Method content berisi semua statement yang ada di dalam curly braces atau kurung kurawal

Supaya nggak lupa, mari kita cek istilah yang ada pada struktur Java Program!

- **Method main**

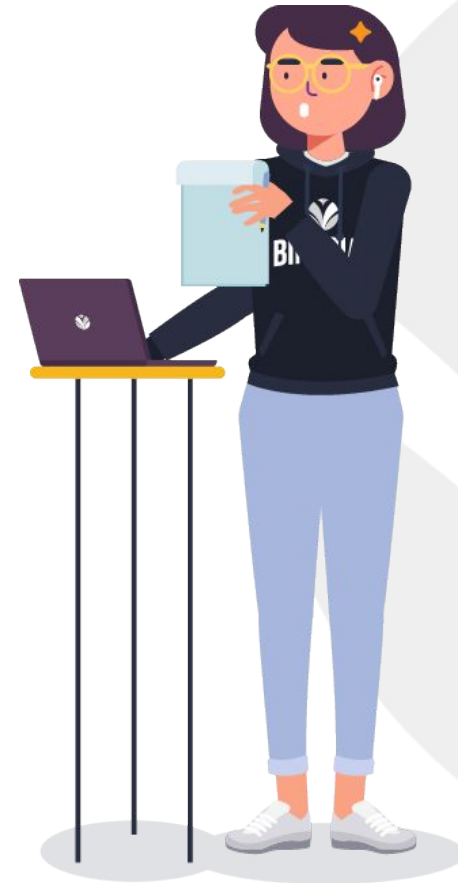
Merupakan blok program yang akan dieksekusi pertama kali dan wajib untuk dibuat.

Kalau nggak dibuat pertama kali, maka programnya nggak akan bisa dieksekusi. Ibarat kamu mau beli kopi di Starbun, eh lupa bawa uang, jadi gagal kan ngopi cantiknya?



- **Parameter**

Merupakan Input dari sebuah method ketika method dideklarasikan.



Ohiya, Java Program Structure juga punya berbagai kode program.

Kode program dalam Java Program Structure disebut dengan **Method**

Kayak apa tuh? mari kita cari tahu! 🚀



Method ini tuh mirip kaya jin di teko ajaibnya Aladdin. Coba deh, biar jin bisa keluar itu tuh Aladdin harus ngapain?

Betul, teko digosok secara berulang sebanyak tiga kali.

Nah, Method dari Java Program Structure bekerja kayak jin yang perlu dipanggil berulang-ulang untuk bisa membantu Back End Engineer. Lho, kenapa?



Karena eh karena~

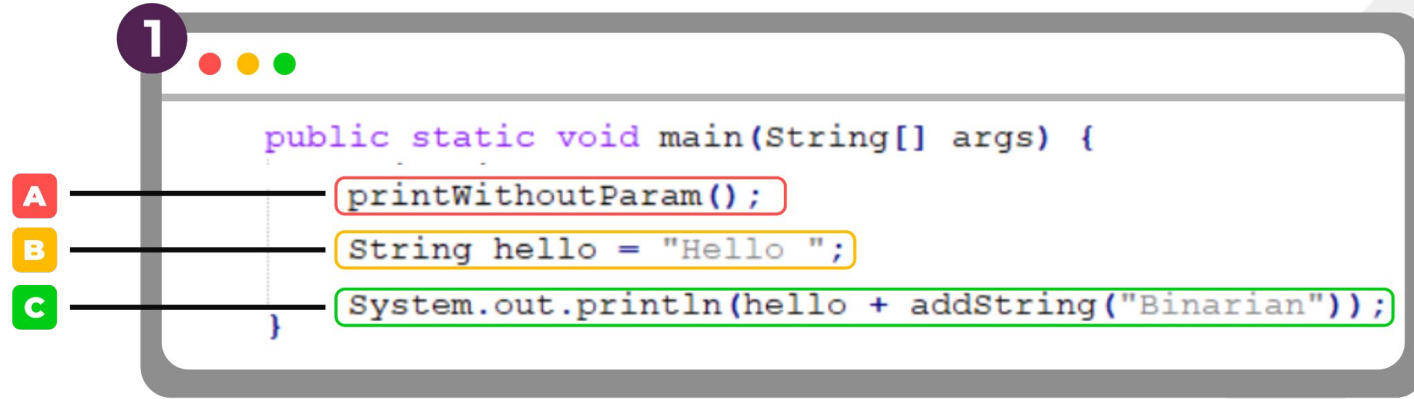
Method adalah sebuah **function** yang terdiri dari kumpulan kode yang bisa dipanggil secara berulang-ulang.

Kemampuan pengulangan ini membuat kita nggak perlu lagi menulis kode program untuk melakukan hal yang sama berkali-kali.



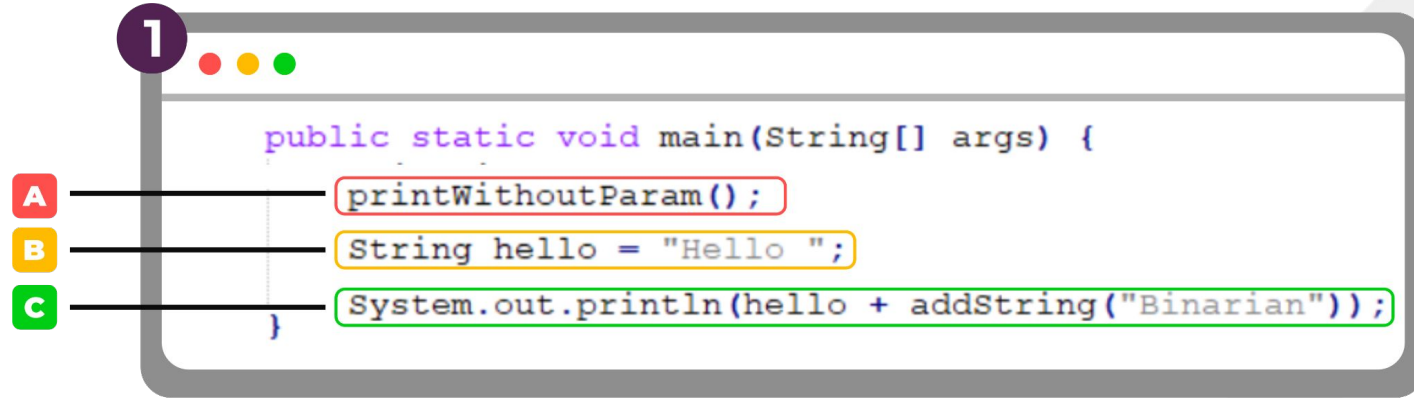
Sekarang, kita coba identifikasi 3 method di bawah, ya!

```
1 public static void main(String[] args) {  
    printWithoutParam();  
    String hello = "Hello ";  
    System.out.println(hello + addString("Binarian"));  
}  
  
2 private static void printWithoutParam() {  
    System.out.println("Hello Guys!");  
}  
  
3 private static String addString(String anyWord) {  
    return anyWord;  
}
```



Main method di atas punya 3 statement yang berbeda, yaitu:

- A** Yang pertama adalah panggilan untuk method **printWithoutParam**
- B** Kedua yaitu deklarasi variable yang bertipe **data String**



- C** Terakhir, menampilkan **output pada console** dengan variable hello yang dideklarasikan di line kedua dan pemanggilan method `addString` dengan argument "Binarian".

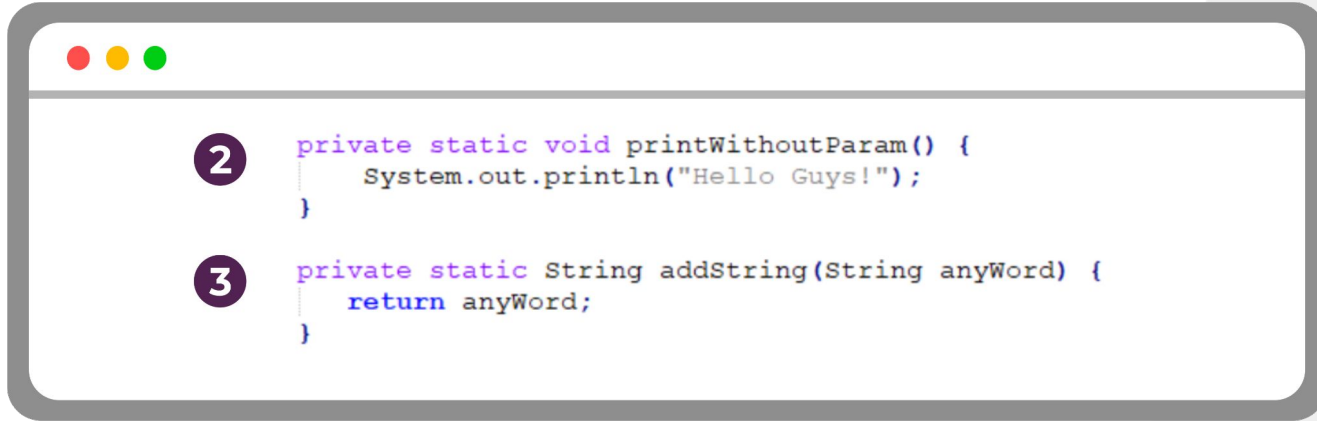
Argument yang dimaksud bukan sebuah alasan untuk memperkuat pendapat, ya. Tapi, argument adalah value untuk memenuhi parameter ketika sebuah method **diinvoke**.


```
public static void main(String[] args) {  
    printWithoutParam();  
    String hello = "Hello ";  
    System.out.println(hello + addString("Binarian"));  
}  
  
private static void printWithoutParam() {  
    System.out.println("Hello Guys!");  
}  
  
private static String addString(String anyWord) {  
    return anyWord;  
}
```

Selanjutnya, coba ubah urutan pendeklarasian method di atas sesuka hati kamu dan lihat apa yang terjadi!

Bimsalabim!

Hampir serupa tapi tak sama~

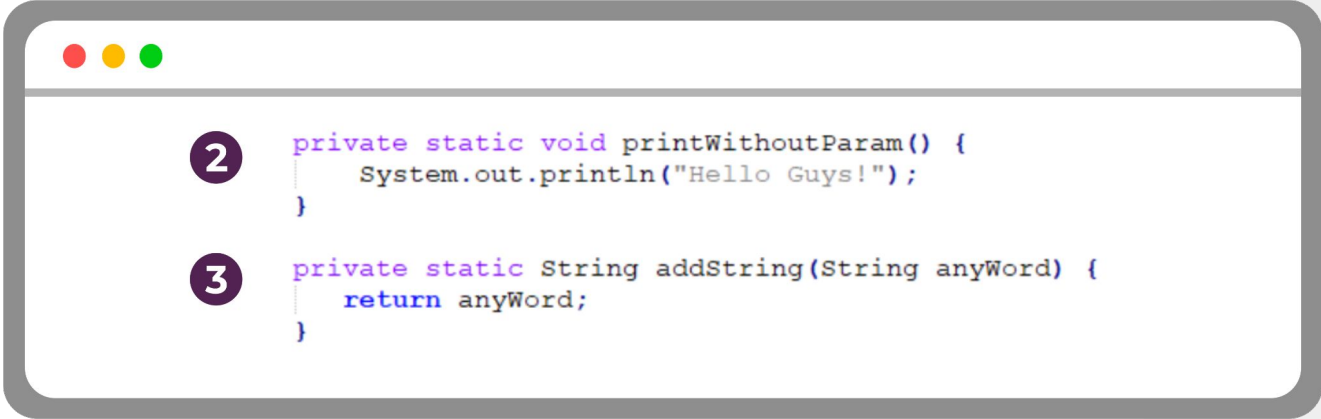


```
2 private static void printWithoutParam() {  
    System.out.println("Hello Guys!");  
}  
  
3 private static String addString(String anyWord) {  
    return anyWord;  
}
```

Dari method di atas, kita tahu kalau ada dua deklarasi method yang punya modifier private dan static.

1. **Method printWithoutParam** yaitu method yang nggak punya parameter dan return value.
2. **Method addString** yaitu method yang punya parameter tipe data String dan return value berupa tipe data String.

Yuk, kupas lebih dalam..



```
2 private static void printWithoutParam() {  
    System.out.println("Hello Guys!");  
}  
  
3 private static String addString(String anyWord) {  
    return anyWord;  
}
```

Return artinya **mengembalikan sebuah value ketika method udah dijalankan.**

Ibarat cowok yang mau nyatakan cinta ke cewek, si cewek pasti punya dua kemungkinan untuk merespon. Antara dia juga punya perasaan atau emang dia nggak punya perasaan sama sekali.

Begitu juga dengan method, ada yang punya return dan ada juga yang nggak punya.

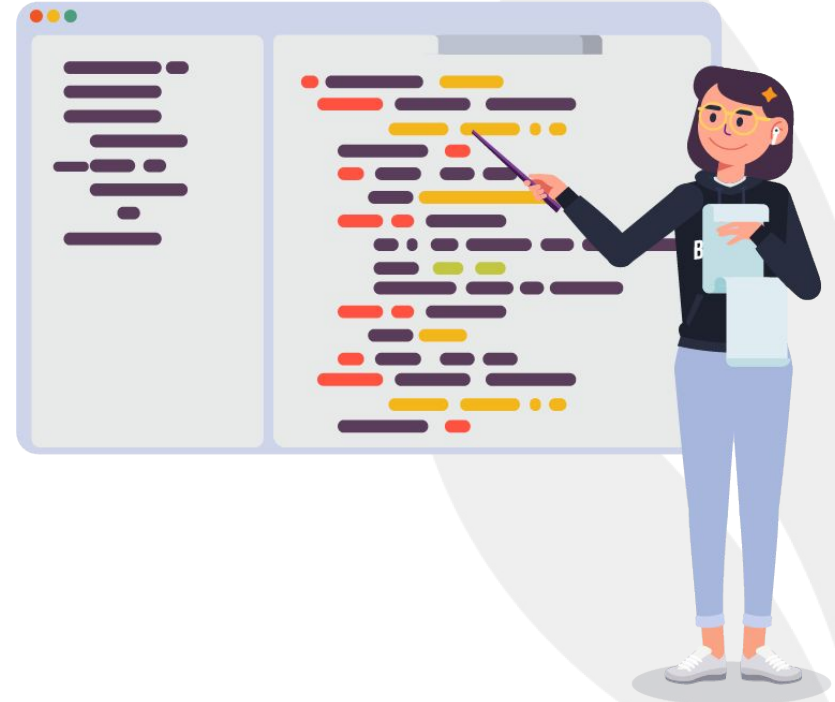
Emang bedanya di mana?

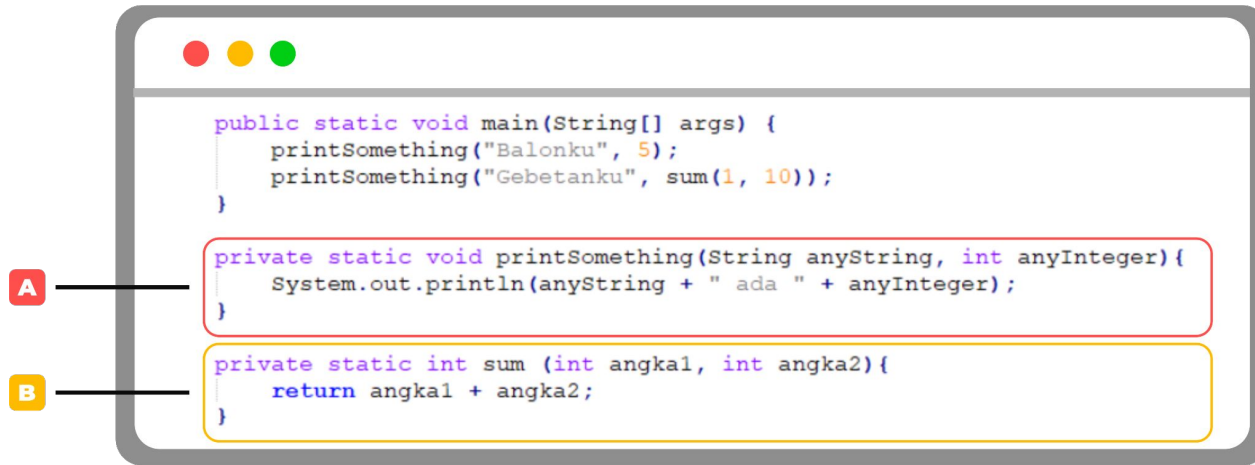
- Method yang punya **return** ditandai dengan **tipe data di sebelah nama method**. Kayak orang pacaran yang lagi duduk berdua~

Contohnya: `String addString(string anyWord)`

- Method yang nggak punya return ditandai dengan keyword **void** di sebelah nama method. Void adalah keyword sebagai penanda bahwa **method tersebut tidak memiliki return value**. Ibarat cewek yang ketika nolak perasaan cowok pakai kata-kata “maaf, ya”

Contohnya: `void printWithoutParam()`



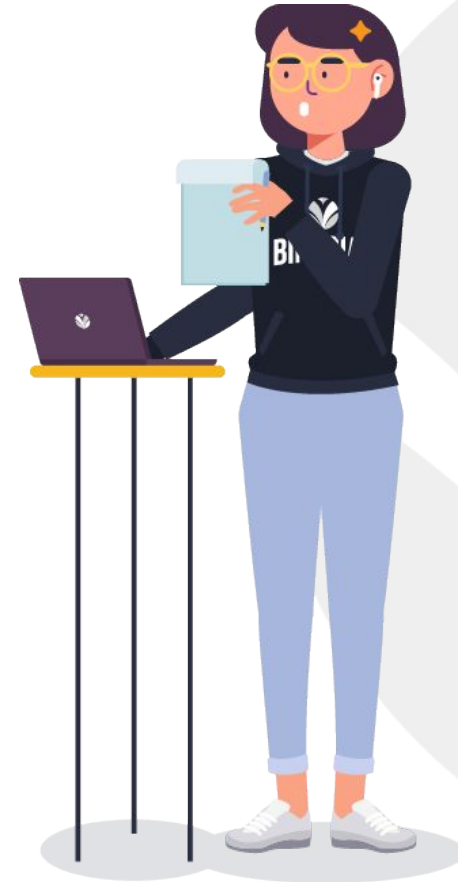


Kita intip implementasinya pada gambar di atas, terdapat dua method yaitu **printSomething** dan **sum**

- A** Method printSomething dan sum memiliki lebih dari satu parameter
- B** Pada content method main statement kedua, return dari method sum akan menjadi argument kedua pada method printSomething

Biar nggak lupa, sekarang kita simpulin yuk!

1. Sebuah method boleh menggunakan multiple parameter
2. Sebuah method hanya boleh memiliki sebuah return value



Selain method, program juga punya aturannya sendiri yang disebut sebagai Syntax.

Jadi, selamat memasuki materi **Konsep Java Syntax dan Convention!**



Hati-hati di setiap kondisi...

Syntax adalah aturan baku yang harus dipatuhi.

Tujuannya, biar sebuah source code bisa dimengerti oleh komputer.

Kalau susunan code yang dibuat nggak mengikuti aturan tersebut, bisa-bisa program yang udah dibuat nggak berjalan dengan semestinya.

Contohnya kayak lalu lintas, nih. Kalau kita nerobos lampu merah, yang ada jalanan malah makin nggak karuan, kan?



Berikut adalah syntax yang udah kita pelajari!

1. Dalam suatu program Java, untuk melakukan eksekusi harus terdapat **main method**.
2. Eksekusi dari kode yang ada dalam suatu block yang sama harus dilakukan secara **berurutan**. Hal ini berlaku untuk pemrograman 1 thread.



3. Setiap statement harus diakhiri dengan **semicolon (;)**
4. Berbagai cara untuk **mendeklarasikan method atau variable.**



Jangan puas sampai sini dulu sob!
Karena masih banyak lagi aturan
yang harus dipenuhi~

Selain dari 4 hal tadi, masih ada juga aturan yang bersifat **conventional** atau bersifat **kesepakatan** (nggak tertulis ataupun diatur).



Dengan mengikuti **convention** yang ada, kita bisa melakukan code lebih baik karena meningkatnya readability pada kode yang udah dibuat.

Tapi, tenang aja, gengs. Untuk convention lainnya akan kita pelajari di materi Clean Code di chapter 7 ya!



Serupa tapi tak sama...

Bukan cewek aja lho yang kalau dicuekin suka sensitif. Java juga sama!

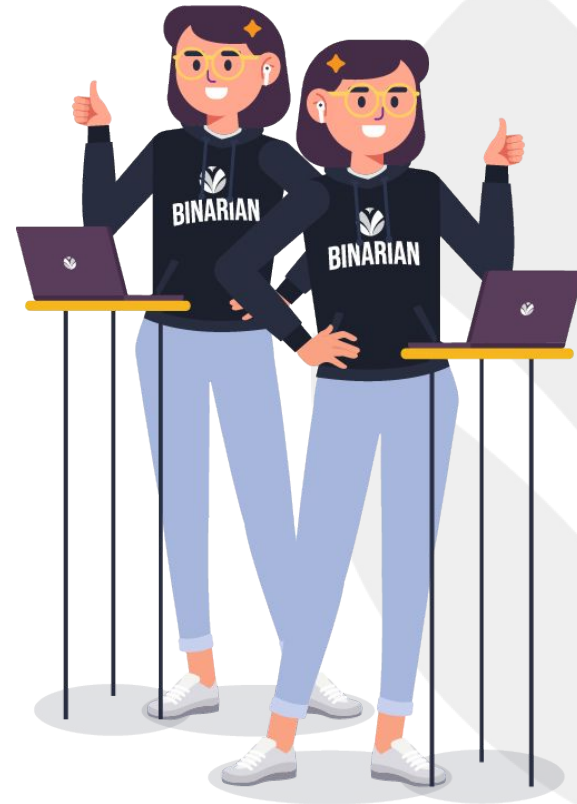
Yang harus diperhatikan dari Java adalah Java itu sifatnya case sensitive, yaitu keadaan di mana huruf besar dan huruf kecil punya arti yang beda.



Contohnya gini, nih:

- **myClass** akan berbeda dengan **myclass** dan berbeda pula dengan **MyClass**.

Walaupun kalimatnya kelihatan sama, tapi artinya berbeda karena cara penulisannya. Begitu juga dengan arti atau **identifier** lain seperti variable, method juga bersifat case sensitive.



Dalam hal penamaan, Java juga udah punya beberapa aturan, lho!

1. Identifier terdiri minimal **satu karakter**.
2. Identifier yang diberikan oleh pemrogram **nggak boleh sama dengan keyword** yang ada di Java.

Misalnya, memberikan nama variable dengan nama int juga.



Ada keran, ada tutup. Dua aturan? nggak akan cukup!

3. Lanjut nih, dimulai dengan huruf atau underscore (garis bawah) atau tanda (\$).

Sebisa mungkin harus dimulai dengan huruf karena mungkin identifier dengan awalan underscore dan (\$) dipake untuk pemrosesan internal dan file import. **Nggak boleh pake angka di awal identifier, ya!**



4. Karakter **berikutnya bisa berupa huruf atau angka 0 sampai 9**. Simbol-simbol seperti '+' dan spasi nggak bisa digunakan
5. **Nggak boleh pake space** pada pembuatan identifier.



Ada lagi, nih! Keyword yang nggak boleh dipakai secara identik!

boolean	double	import	protected	throw
break	else	instanceof	public	throws
byte	extends	int	return	transient
case	false	interface	short	true
catch	final	long	static	try
char	finally	native	strictfp	void
class	float	new	super	volatile
continue	for	null	switch	while
default	if	package	synchronized	
abstract	do	implements	private	this

naik vespa sama afgan,
jangan lupa dicatet gan



Karena dilarang itu nggak enak, berikut adalah contoh nama variable yang diperbolehkan! Yeayy~

- @2var
- _status
- tanggal
- jumlahBarang
- nama_kecil
- final_test
- int_float

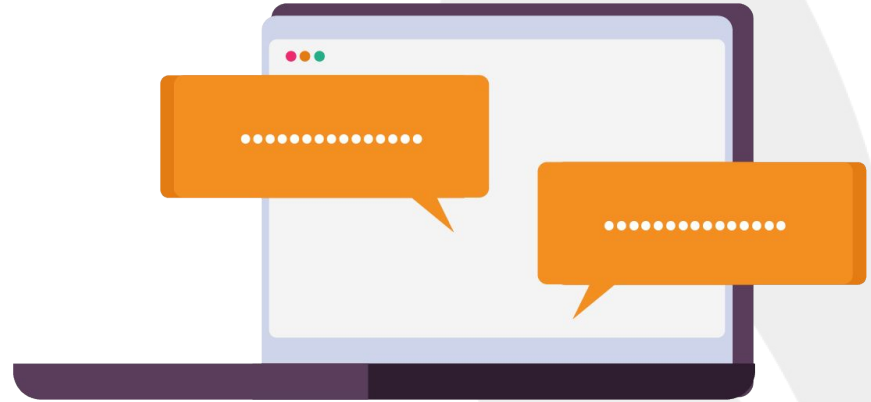
Mantep ga tuh



Bukan cuma di sosmed, program juga punya istilah comments, lho!

Tapi, comments yang ini artinya bukan komentar tentang sesuatu ya, gengs!

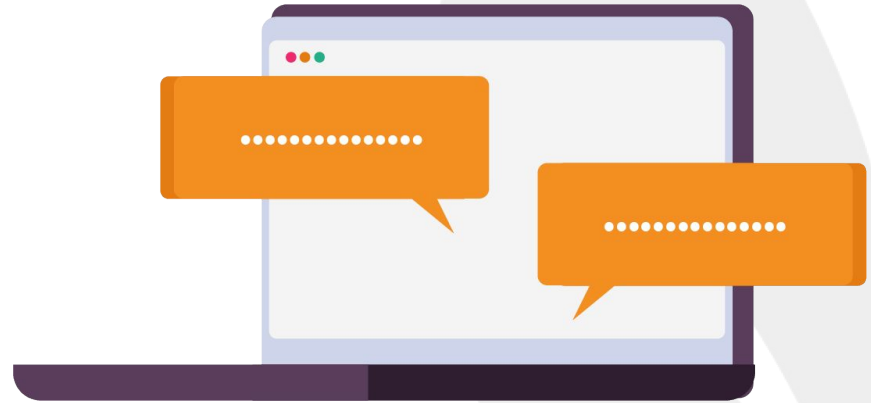
Comments adalah line yang nggak bakal dieksekusi sebagai program.



Comments punya beberapa fungsi, yaitu:

1. untuk **menonaktifkan line of code**, dan juga
2. **ngasih keterangan tambahan pada code** biar mudah terbaca.

Yuk, kita intip cara untuk melakukan comments!



Berikut cara melakukan comments!

1. Untuk comments yang bersifat **single line**, lebih baik kita menggunakan //

contohnya:

```
System.out.println("Hello World"); // This is a comment
```

1. Untuk comments **multiple line**, maka sebaiknya menggunakan /* */

contohnya:

```
/* The code below will print the words Hello World  
to the screen, and it is amazing */  
System.out.println("Hello World");
```



Gimana nih praktik buat Hello Worldnya pake Notepad dan IntelliJ? Apakah seru dan menyenangkan?

Menurut kamu, apa kelebihan dan kekurangan dari dua tools tersebut ketika kamu praktik membuat Hello World? Dan tools mana yang lebih nyaman kamu gunakan?



Nah, selesai sudah pembahasan kita di topic 2 ini.

Selanjutnya, kita bakal bahas tentang **Java Data Types & Variable.**

Penasaran kayak gimana? Cus langsung ke topik selanjutnya~

