

# Java Introduction and Tools Preparation

**Silver** - Chapter 1 - Topic 1

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



### Hai teman-teman, selamat datang! 🎉

Sebagai awal dari perjalanan kita buat eksplor tentang Back End Java, chapter 1 ini mengajak kamu kenalan sama dasar syntax pada pemrograman Java.

Untuk lebih spesifiknya, kita mengelaborasi tentang **Java Introduction, Data Types and Variable**. Kalau gitu, yuk langsung aja kita kepoin~



**Detailnya, kita bakal bahas hal-hal berikut ini:**

- Pengenalan Back End
- Back End Engineer skill set dan day to day activities
- Pengenalan Back End tools
- Pengantar Java
- 3 komponen pada platform Java (JDM, JRE, JDK)
- IDE dan Terminal
- Set up Tools (Java, Maven, IntelliJ)
- Apache Maven Basic



Karena tak kenal maka tak sayang,  
sebelum kamu terjun jadi Back End  
Engineer, mari kita kenalan dulu sama  
yang namanya **Back End**.

Pernah denger?

Tenang~ abis ini kita jelasin 🌟

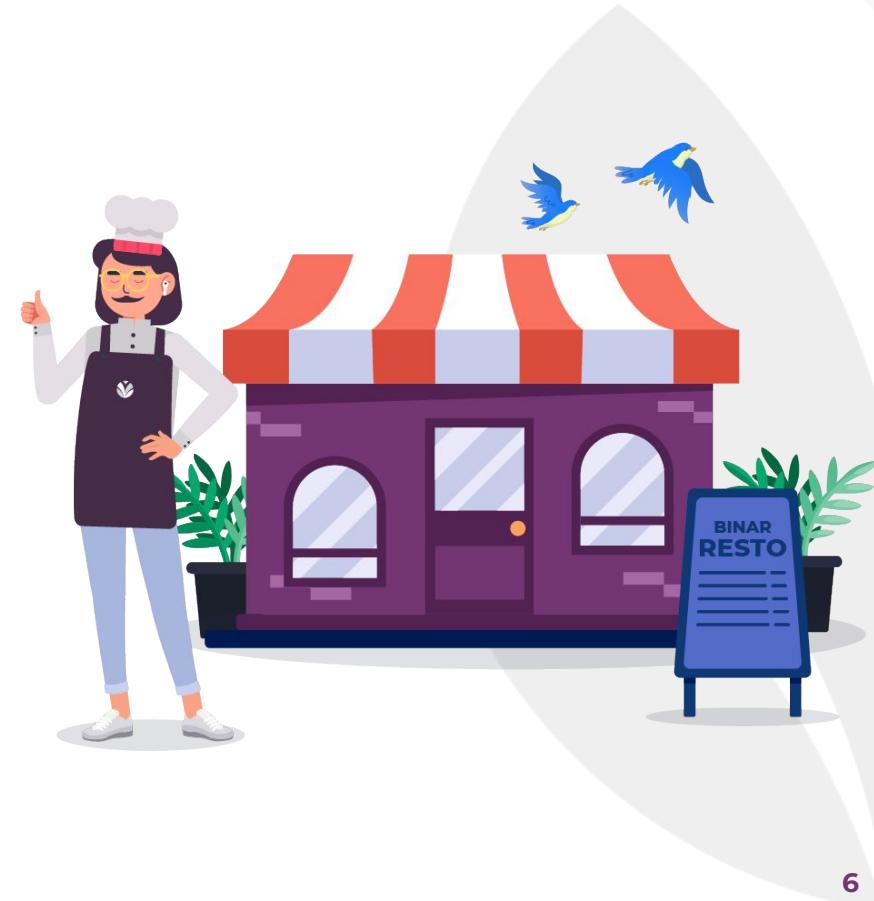


## Back End? Apaan tuh?

Sabar, sabar~ Biar lebih paham, kita coba jelaskan pakai analogi restoran, ya!

Restoran memiliki ruang depan untuk kita duduk as customer dan juga dapur sebagai tempat memasak pesanan kita.

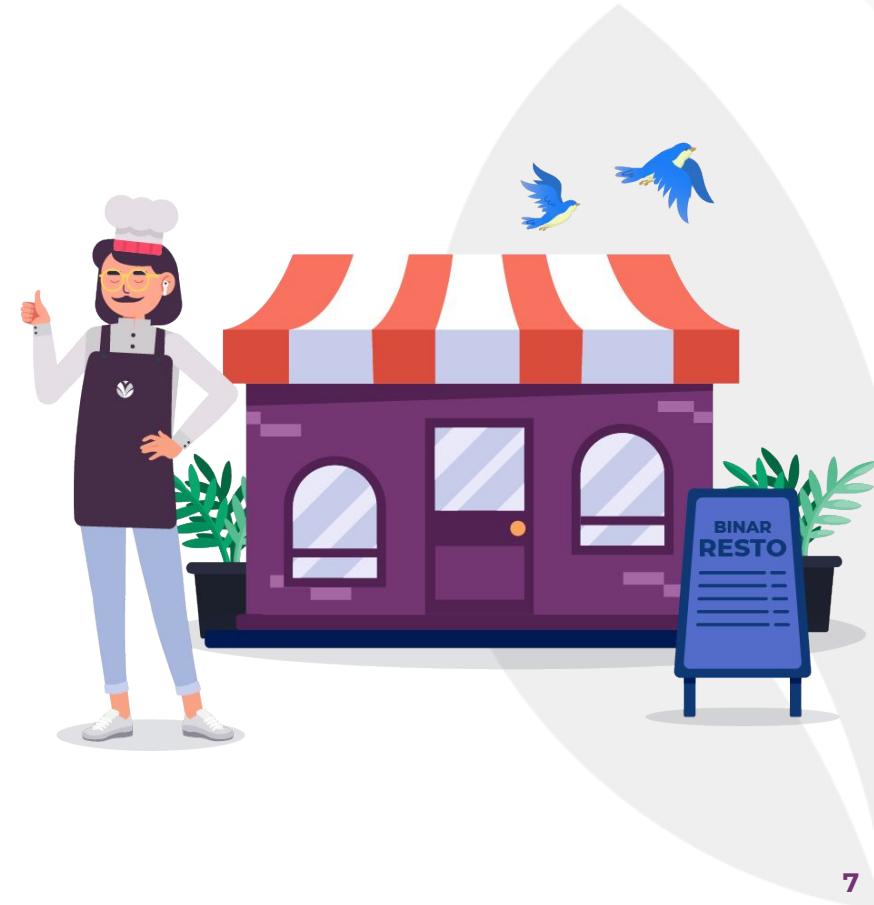
Pas kamu duduk di ruang depan, kamu bisa tahu nggak di dapur lagi pakai resep dan bahan apa aja? Enggak kelihatan, kan.

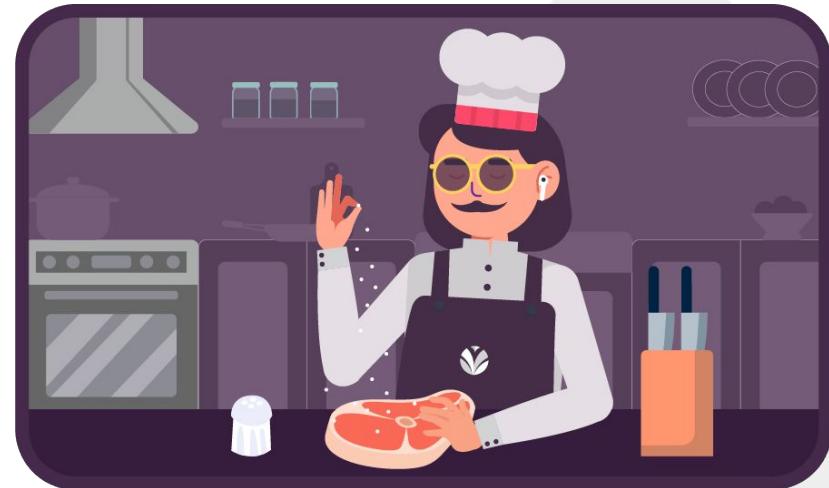
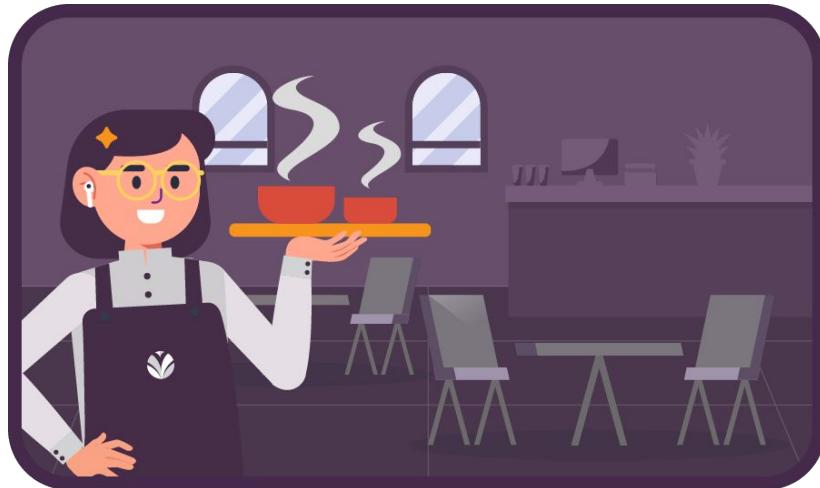


Dari situ kita tahu dong bahwa ada bagian dari restoran yang memang bisa kita lihat dan bagian yang tidak bisa kita lihat.

Terlepas dari bisa atau nggak-nya kita melihat, pesanan kita tetap dalam proses pembuatan biar mateng.

Nah, sekarang anggap aja **sebuah aplikasi adalah sebuah restoran**. Aplikasi punya ruang depan dan dapur untuk memproses pesanan kita.





Suasana, interior ruangan, pelayan yang ramah, atau semua yang bisa dilihat oleh pelanggan merupakan **front end**-nya.

Sedangkan resep masakan, teknik memasak, cara mendapatkan bahan, atau semua yang nggak bisa terlihat oleh pelanggan bisa dibilang sebagai **back end**-nya.

Dari analogi tadi, kita bisa tahu bahwa kualitas restoran yang bagus sangat tergantung dari pihak di belakang layar.

Kalau kita menerapkan analogi ini ke aplikasi, pihak penting di balik layar ini disebut sebagai **Back End Engineer**.



Back End Engineer ini tugasnya berkaitan dengan sisi **server**, baik itu mengurus server, bikin skema database, sampai eksekusi fitur di software yang sesuai sama [flowchart](#) yang udah disepakati.

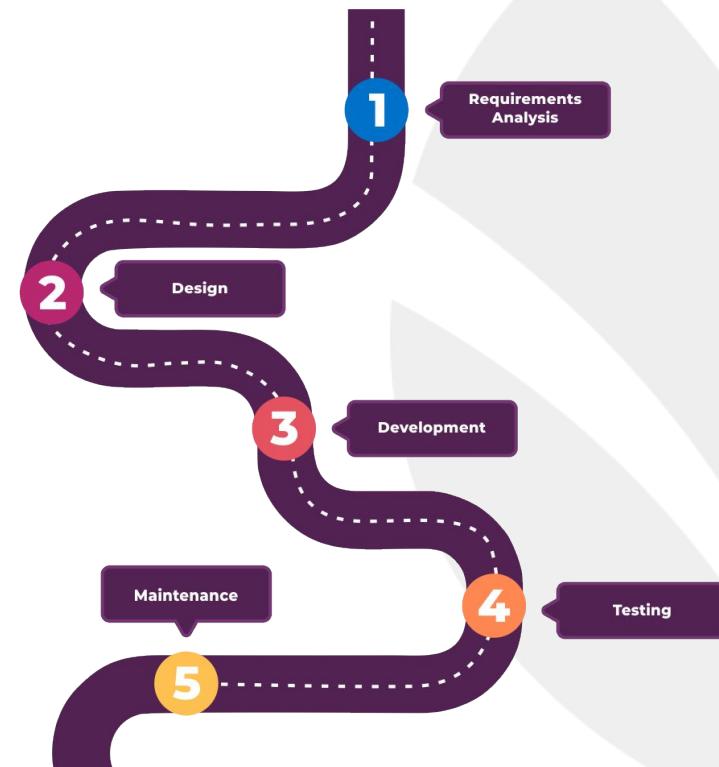
Masih asing sama istilah-istilah di atas? Tenang~ kita bakal belajar semuanya dari dasar, kok. 😊



### Sama kayak masak mie instan, bikin software juga ada prosesnya, lho

Bedanya, **proses bikin software ini disebut sebagai SDLC atau Software Development Life Cycle yang isinya ada 5 tahapan.**

SDLC merupakan sebuah siklus dari berbagai proses yang terjadi dalam pembuatan software. Siklus ini terjadi secara berulang-ulang.



Sebagaimana udah disebutkan sebelumnya, ada lima tahapan SDLC yang harus dilalui. Lebih lengkapnya kayak gini:

- 1. Requirement Analysis:** proses untuk menentukan ekspektasi dari pembuatan fitur baru. Bisa juga berupa modifikasi serta definisi dari penyelesaian fitur tersebut.
- 2. Design:** proses untuk mentranslate requirement yang udah dikumpulkan di tahap sebelumnya menjadi spesifikasi teknikal.



3. **Development:** proses pembuatan code.
4. **Testing:** proses test secara end to end.
5. **Maintenance:** proses pengelolaan aplikasi setelah deployment aplikasi dilakukan.



Ohiya, selain lima tahapan tadi, SDLC juga punya peran-nya sendiri. Mirip kayak aktor atau aktris yang suka main film 😊

Iya, meskipun aktor/aktris ini cuma satu orang, tapi dia bisa berperan sebagai tokoh antagonis dan protagonis. Nah, SDLC juga punya peran, yaitu :

## 1. Product Owner

Merupakan pemilik product. Product owner bertanggung jawab dalam pengelolaan backlog dan product delivery.



## 2. Product Manager

Bertanggung jawab untuk melakukan riset serta pengembangan dari produk.



### 3. Technical

Khusus untuk peran Technical ini diisi oleh orang-orang yang terdiri dari:

- Back End Engineer      • SRE/Dev Ops
- Front End Engineer     • Tester/Software Quality Team
- Mobile Engineer        • Business Analyst

Kita bahas satu-satu yuk.



Kalau slide sebelumnya seputar kerangka aja, sekarang kita menuju lebih spesifik ke tanggung jawab orang didalamnya, yaitu:

- **Back End Engineer** bertanggung jawab untuk men-develop system dari server side.
- **Front End Engineer** bertanggung jawab untuk men-develop tampilan dari suatu web.
- **Mobile Engineer** bertanggung jawab untuk mendevelop aplikasi mobile.



- **Site Reliability Engineer (SRE)/Dev Ops** bertanggung jawab untuk mempersiapkan environment agar proses development, deployment dan release bisa berjalan dengan baik.
- **Tester** bertanggung jawab untuk memastikan Software Quality.
- **Business Analyst** bertanggung jawab untuk menerjemahkan requirement yang ada dari product owner ke functional document.





Selanjutnya, supaya gambaran kita lebih clear, ada video yang bisa dijadikan referensi. Video ini membahas contoh struktur tim yang biasa digunakan di perusahaan teknologi dengan Spotify tribe model. Kamu bisa klik [Struktur Team di Perusahaan Teknologi](#)

Wiih, kita udah kenal istilah SDLC. Mulai dari peran sampai orang-orang yang ada di dalamnya. Dapet progress!

Selanjutnya, ada istilah baru yang perlu dipelajari. Kita move on ke **tugas Back End Engineer**. Geser yuk slide-nya~



Menjadi seorang Back End Engineer artinya belajar banyak istilah baru. Selain SDLC, kamu bakal ketemu **keyword** kayak yang ada di bawah ini :

Server Side Applications

API

REST

RESTful

Stateless Service

Stateful Service

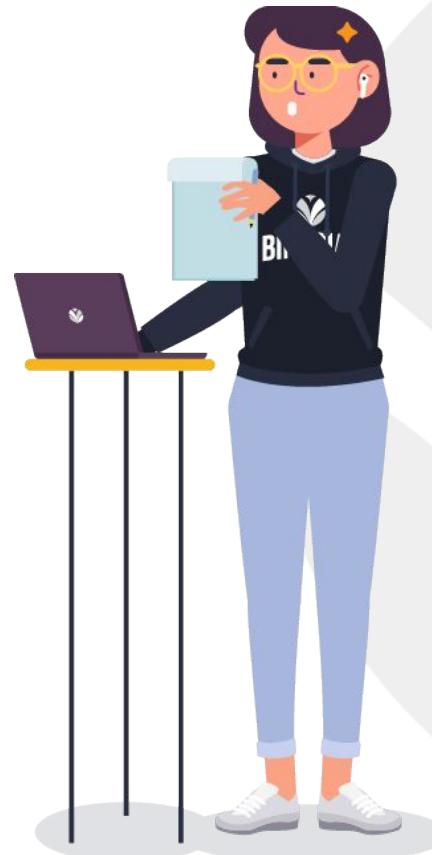
Database

Unit Test

Microservice

Monolith

UML Diagram



### Biar pelan asal paham~

Banyak banget kan istilah yang berhubungan sama Back End Engineer?

Eits, nggak perlu khawatir. Seiring dengan perjalanan belajar kita di course Back End Java, pelan-pelan kita bakal belajar semuanya sampai kamu familiar.

Jadi, nggak perlu overthinking duluan ya, gengs!



**“Terus, Back End Engineer kerjanya ngapain aja?”**

Nah, kembali ke laptop, si Back End Engineer punya beberapa **tugas yang saling berkaitan**. Alias nggak bisa dipisahin satu sama lain.

Kayak hati kamu sama doi~



### Berikut tugas dari Back End Engineer!

- Melakukan analisis terhadap requirement yang diterima.
- Membuat desain system pada server side sesuai dengan requirement.
- Berkolaborasi dengan front end engineer, (cont.)

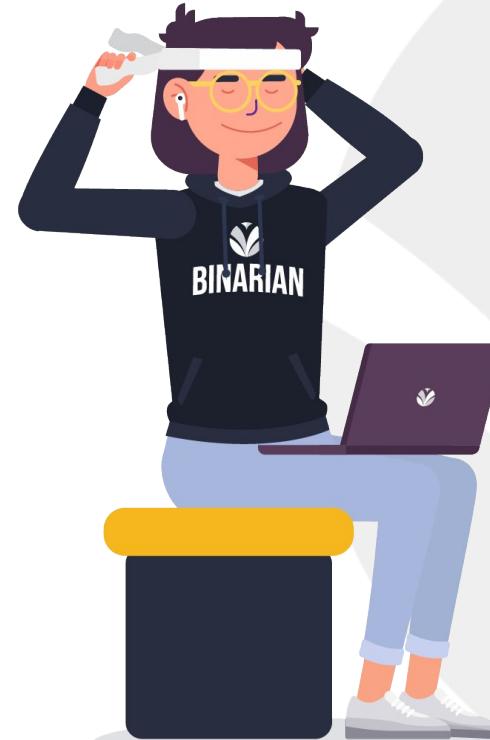


- Men-develop server side system (service atau aplikasi yang berjalan dari sisi server) sesuai dengan desain yang sudah dibuat.
- Menyelesaikan bugs yang berkaitan dengan back end.



**Eits, masih ada part terakhir nih untuk tugas dari Back End Engineer!**

- Memastikan system yang sudah dibuat berjalan dengan baik hingga ke production.
- Membuat dokumentasi teknikal.
- Having fun with team.



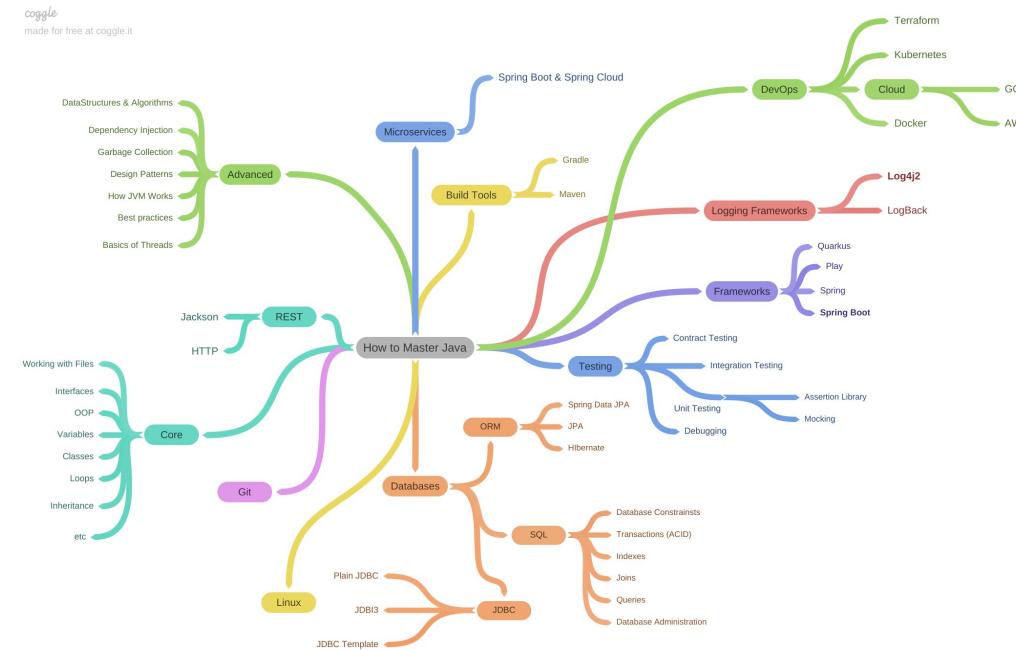
**“Kalau gitu, kita belajarnya mulai dari mana?”**

Biar makin paham, gambar pada halaman berikutnya adalah roadmap materi yang nantinya bakal kita pelajari bareng-bareng di course Back End Java.

Mari kita buka tirai putih setelah ini~



# Java Introduction and Tools Preparation



Bisa kamu intip dulu roadmap di atas. Ada poin-poin yang bisa membantu kita mastering Java secara spesifik. Penjelasan lebih detailnya, kamu bisa cek pada video di bawah ini.

[\*\*How To Master Java - Java for Beginners Roadmap\*\*](#)

A little progress is a progress. Wajar banget kalau kita belum familiar sama istilah baru secepat sihir 🧙

Buat mempermudah belajar, kita lihat dulu yuk isi materi yang akan dipelajari di **Back End Java ini.**



## Dari tulisan, jadi penasaran~

Gimana?

Udah makin penasaran sama isi course Back End Java?

Nah, dari yang tadi udah dijelasin, nantinya kita bakal berpetualang mengarungi materi Back End Java yang isinya, kayak disamping ini nih~



Sebelum lanjut belajar, jangan lupa **download 3 tools di bawah ini** dulu, nih!

Tujuannya biar proses PDKT kita sama Back End Java bisa berjalan lancar tanpa hambatan~



Java Development Kit 8



Apache Maven versi 3.8.4



IntelliJ Idea Ultimate Edition  
untuk **IDE (tools bahasa pemrograman)**

Namanya juga PDKT sama Back End Java,  
kita harus tahu dulu nih latar belakang  
alias konsep dasarnya.

Java?

Apa tuh?



### Sebelum ke Java-nya, kita kenalan dulu yuk sama penciptanya~

Java diciptakan sama **James Gosling**, developer dari Sun Microsystems pada tahun **1991**.

Java adalah bahasa pemrograman berorientasi object murni yang dibuat berdasarkan kemampuan terbaik bahasa pemrograman object sebelumnya seperti C++, Ada dan Simula.

Gampangnya, Java adalah bahasa pemrograman general purpose untuk mengembangkan sebuah aplikasi atau software.



● **James Gosling**  
● Java Programming Language Founder

### Waw kopi yang berubah jadi bahasa program?!

Awalnya, James Gosling menamakan bahasa pemrograman sebagai Oak.

Lalu diganti jadi nama Java karena Java merupakan jenis kopi yang paling banyak dikonsumsi oleh engineer di Sun Microsystems.

Yup, betul banget! **Java adalah sebuah kopi yang berasal dari pulau Jawa, Indonesia!**



**Berakit-rakit ke hulu, berenang-renang ketepian. Dirakit aja dulu, diakuisisi kemudian~**

Jangan salah! Java juga punya slogan yang keren banget. Bunyinya yaitu:

**“Write Once, Run Anywhere”**

Yang artinya kode yang udah dijalankan di satu platform nggak perlu lagi dikompilasi ulang untuk di tempat lain.

Berhubung canggih banget nih, pada tahun 2010 Java diakuisisi oleh Oracle. Emejing!~



**Kalau abis ngomongin konsep,  
nggak pas rasanya kalo nggak  
sambil bahas karakteristik, ya kan?**

Ibarat mau nembak cewek, pasti setiap cowok punya karakteristik idealnya masing-masing, kan? Entah itu yang zodiaknya libra atau mirip Lisa Blackpink.

Java juga punya karakteristik yang mesti kita tahu.



Karakteristiknya tuh, kayak:

- **Java sangat berorientasi pada object atau OOP.** Jadi implementasi programnya bisa jadi lebih baik.
- **Arsitekturnya kokoh dan pemrograman yang aman.** Kayak motor yang pake kunci ganda! Aman pokoknya~

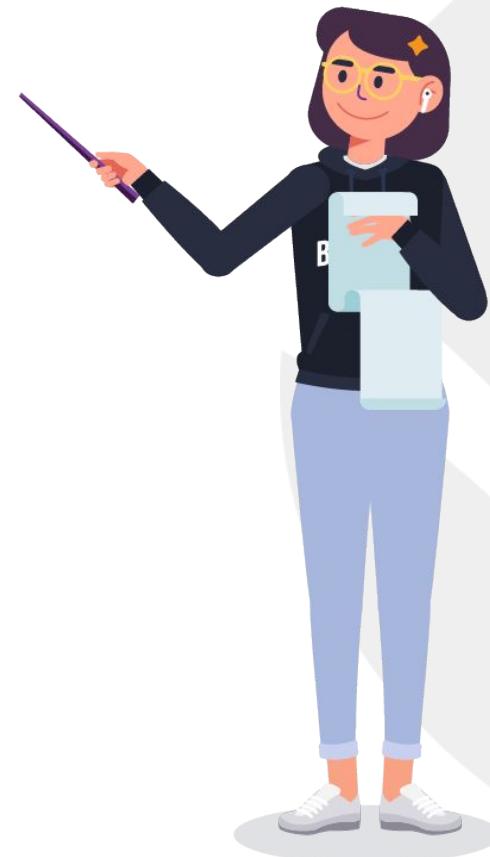


## Masih ada lagi bosque~

- **Tidak mudah “hang”** karena ada Garbage Collector yang mengumpulkan object yang tidak terpakai.

Selain itu juga ada Exception yang ngebantu buat menangani masalah yang terjadi.

“Pokoknya, apapun masalahmu, aku hadir menjadi pahlawanmu!” kata Java

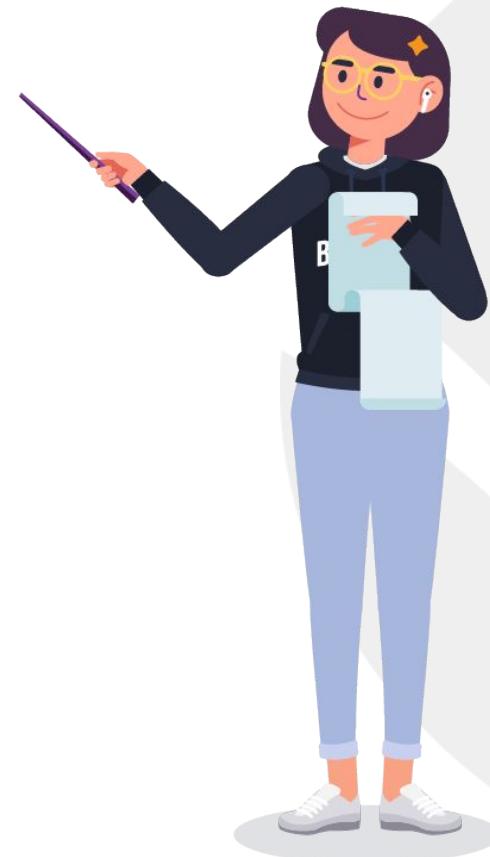


### Oke, yang terakhir nih!

- Program Java dijalankan dengan [interpreter](#) melalui **Java Virtual machine (JVM)**.

Hal ini yang bikin [source code](#) Java yang udah dikompilasi jadi Java bytecodes bisa dijalankan di platform yang berbeda-beda.

Satu titik, dua koma. Sekali klik, bisa dipakai di platform berbeda! Cakepnya mana dulu nih~



### Di mana ada karakteristik, di situ ada kelebihan...

Kalau ditanya, "kenapa kamu prefer naik ojek online?"

Pasti jawabnya karena banyak keuntungan, kan? karena ada voucher potongan harga, nggak ribet, nggak sulit dicari, dan lain-lain.

Nah, Java juga bukan cuma sekedar program yang nggak punya keuntungan, lho.

Nggak percaya? Nih, buktinya!



### Berikut adalah kelebihan Java yang wajib kita tahu!

#### 1. Object Oriented

Java merupakan bahasa pemrograman berorientasi object. Jadi, **semua yang ada di java bersifat object**. Sehingga dapat dengan mudah diperluas karena udah didasarkan pada Object.



## 2. Simple

Pemrograman Java dirancang biar mudah dipelajari. Jadi, kita bisa dengan **mudah menguasai Java kalau udah paham konsep dasar OOP**.

Tapi walau simple tetap nggak bisa langsung bimsalabim jadi master Java ya guys ya.....



Nggak cukup cuma dua,  
kelebihannya masih banyak lagi~

### 3. Portable

Yes, mirip kayak arti dalam bahasa Indonesia yang mudah dibawa, **program Java bisa dijalankan di berbagai platform** asalkan JVM untuk platform tersebut tersedia.



## 4. Garbage Collection

Program Java bisa buang sendiri “sampah-sampah” yang nggak penting.

Artinya, program nggak perlu menghapus object yang dialokasikan di memori karena **dealokasi memori dilakukan secara otomatis**.

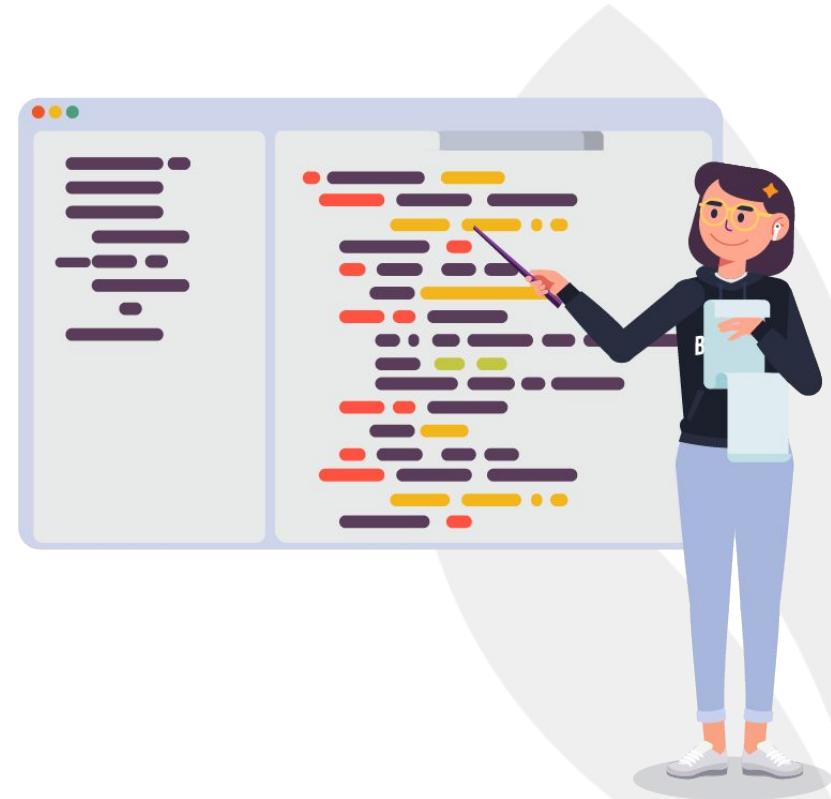
Efeknya, program Java bebas dari permasalahan pengelolaan memori yang membagongkan.



## 5. Architecture Neutral

Pada dasarnya, **Java dirancang biar bisa kerja di berbagai arsitektur prosesor dan berbagai sistem operasi.**

Jadi, Java kerja dengan interpreter Java untuk mengubah program Java menjadi byte-code sebelum dijalankan oleh suatu mesin tertentu.



## 6. Robust

Berangkat dari hasil interpreter Java yang memeriksa seluruh akses sistem yang dilakukan program, maka program Java nggak bakal bikin sistem jadi crash. Jadi, kalau ada masalah serius, ya harus diomongin baik-baik.

Eh salah,

Maksudnya kalau ada masalah serius, **program Java bakal bikin pengecualian (exception)** yang bisa ditangani dan dikelola sama program tanpa berisiko bikin macet system.



## 7. Secure

Sistem Java itu canggih banget, lho. Apalagi kalau tentang mengelola memori.

Java nggak cuma bisa memverifikasi semua akses ke memori, tapi juga menjamin kalau nggak bakal ada virus yang “membonceng” di program yang lagi berjalan.



Masih lanjutan dari Secure, nggak bakal ada yang membonceng Java karena Java nggak mendukung pointer. Jadi, **program nggak dapet akses ke area sistem di mana ia nggak dapet otorisasi.**

Ibarat rumah nih, udah dikunci, dirantai, terus digembok pula. Pokoknya keamanan tetap nomor 1.



## 8. Platform Independent

Program Java mendukung metode native code.

**Native code merupakan proses yang memungkinkan pemrogram menulis fungsi dalam bahasa lain.**

Biasanya berupa C++ yang bisa dieksekusi secara lebih cepat karena doi langsung berjalan di hardware yang bersangkutan. Dibandingkan yang ditulis dalam Java yang berjalan di atas JVM.

Metode native code dikaitkan secara dinamis (dynamically linked) ke program Java atau dikaitkan dengan program saat runtime.



### 9. Multithreaded

**Dengan fitur multithreaded, memungkinkan Java buat nulis program yang bisa melakukan banyak tugas secara bersamaan.**

Fitur ini memungkinkan developer buat membangun aplikasi interaktif yang bisa berjalan dengan lancar.

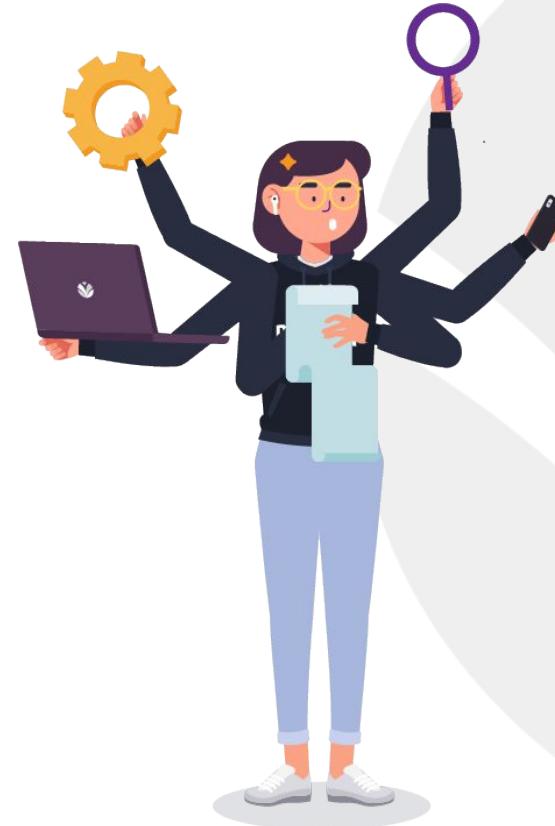


### Kelebihan terakhir nih, sob!

#### 10. Interpreted

**Kode byte Java diterjemahkan dengan cepat ke instruksi mesin asli dan nggak disimpan di mana pun.** Sehingga, proses development jadi lebih cepat dan analitis karena proses menghubungkan adalah proses tambahan dan ringan.

Karena ngafalin satu tempat lebih mudah daripada banyak tempat.



# LANJUTTT



Udah paham tentang Java, selanjutnya kita bakal kenalan sama istilah yang lainnya, yaitu **Java (JVM, JRE, JDK)**.

Biar nggak penasaran, langsung aja yuk kita bahas!

Karena akan ngebahas JVM, JRE, dan JDK, kita harus tahu dulu nih kalau mereka sebenarnya bagian dari Platform Java!

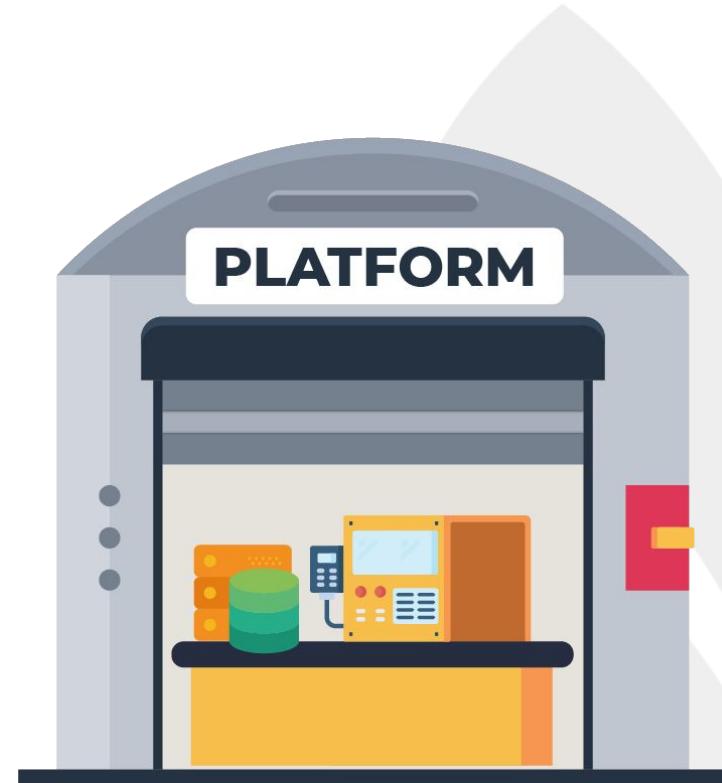
Iya, bener! Itu dia kenapa kita mempelajari tiga sekawan ini.



Berhubung JVM, JRE, JDK adalah bagian dari platform Java, kita jelasin dulu nih definisi platform.

Gampangnya, **platform adalah tempat hardware dan software berjalan**.

Biasanya platform dibuat berdasarkan nama sistem operasi yang digunakan, misalnya kayak Windows, Linux, atau Solaris.



Dengan kata lain, untuk menjadi platform yang utuh, diperlukan pondasi platform dan software. Si pondasi dan software ini disebut komponen pendukung berdirinya platform.

Disinilah kehadiran JVM, JRE, JDK merubah dunia platform Java.



### Masih nggak percaya? Nih penjelasannya~

Kayak slogan mie instan yang bunyinya “sarimin isi dua~”, platform Java memiliki dua komponen, yaitu:

#### 1. Java Virtual Machine (Java VM)

Yaitu pondasi untuk platform Java yang bisa dipakai di berbagai platform hardware.



### 2. Java Application Programming Interface (Java API)

Yaitu kumpulan komponen software yang siap pakai (ready-made software components) buat berbagai keperluan. Udah kayak makanan aja ya, siap saji.



Masuk ke materi inti tentang platform Java, ternyata ada 3 komponen pentingnya nih gais!

### 1. JVM atau Java Virtual Machine (JVM)

Sering dianggap sebagai **jantung** dari bahasa pemrograman Java.



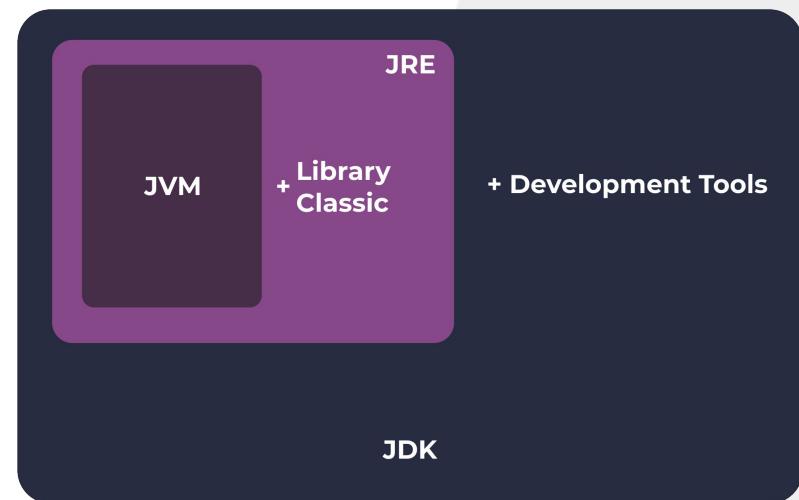
### 2. JRE atau Java Runtime Environment (JRE)

Merupakan **implementasi** dari **JVM**. JVM ini yang memberikan platform buat mengeksekusi program-program Java.

### 3. JDK atau Java Development Kit (JDK)

merupakan **komponen inti** dari Java.

Masih bingung? Kalau gitu, yuk kita cek penjelasannya!

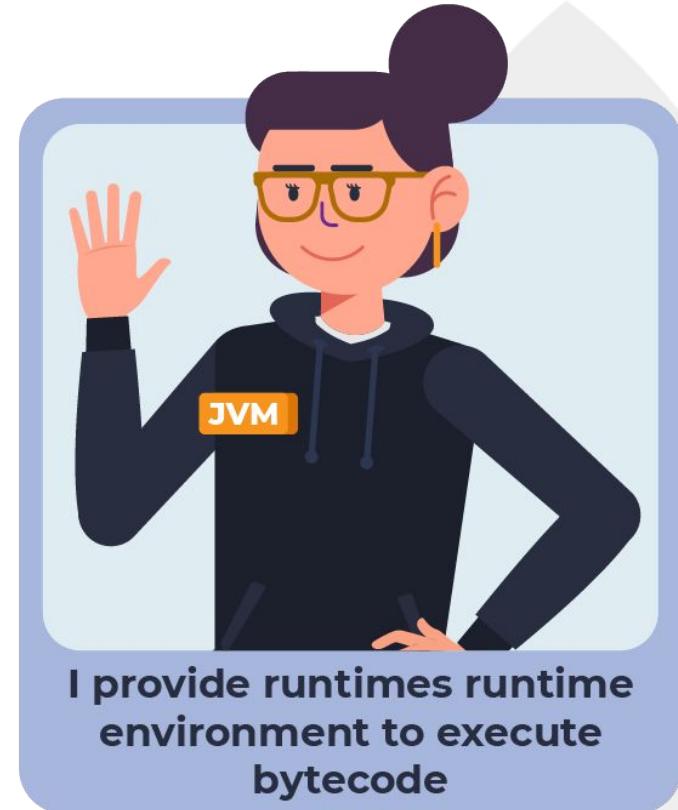


### Sekarang waktunya kita kenalan sama JVM alias Java Virtual Machine!

Java Virtual Machine (JVM) adalah mesin yang **menyediakan lingkungan runtime** buat menjalankan kode dalam aplikasi Java.

JVM punya tugas buat mengolah code yang udah di-compile sehingga bisa dijalankan.

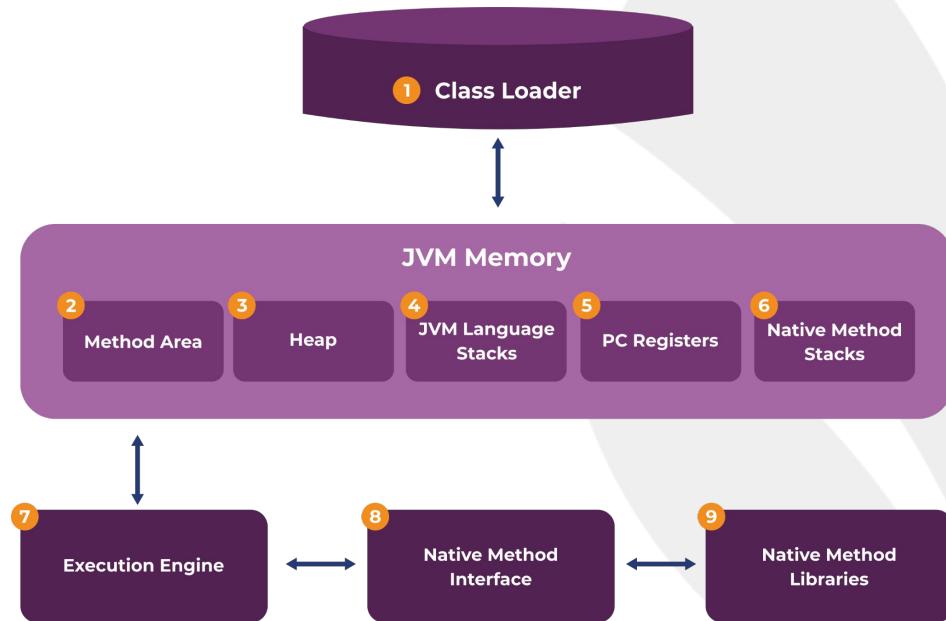
Selain itu, JVM merupakan bagian dari JRE dan JDK yang bakal dibahas di slide selanjutnya.



## “Emang JVM isinya apa saja?”

JVM sendiri terdiri dari beberapa hal, yaitu:

1. Class Loader
2. JVM Memory
3. Execution Engine
4. Java Native Interface



### Setelah ada bayangan tentang JVM, selanjutnya kita bahas tentang JRE

JRE (Java Runtime Environment) merupakan **paket Java yang dibutuhkan untuk menjalankan aplikasi Java**. Paket ini isinya ada JVM dan library-library Java.

“Terus dipakainya ketika melakukan apa?”

JRE bisa dipake pas kita mau menjalankan aplikasi Java, tanpa harus menulis code. Jadi kita cuma perlu JRE aja.



### “Kalau JDK artinya apa dong?”

JDK adalah singkatan dari Java Development Kit, yaitu sebuah Software Development Kit (SDK) untuk Java.

JDK itu mirip sama JRE. Bedanya, JDK adalah sebuah paket software Java yang digunakan untuk melakukan kompilasi dari kode Java menjadi byte code.



### Selain itu, JDK berisi apa aja?

JDK juga punya open source yang bernama **OpenJDK** dan **OracleJDK**.

Tapi, untuk yang OracleJDK punya ketentuan lain, yaitu buat penggunaan pribadi gratis, sedangkan untuk enterprise diwajibkan untuk membayar 2.5\$ per user per bulan.

Kurang keren apa coba JDK ini?



**Pasti dari lubuk hati yang paling  
dalam, kamu mau nanya  
“Sebenarnya hubungan antara  
mereka bertiga itu apa, sih?”**

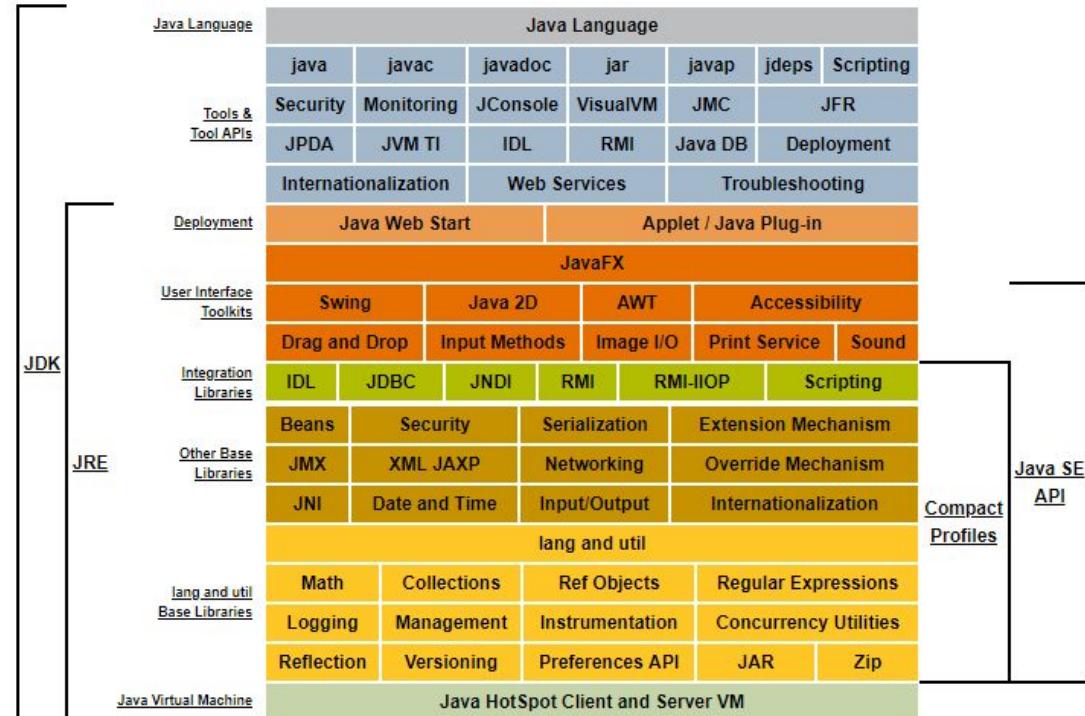
Ini nih yang ditunggu-tunggu. Sebenarnya, mereka bertiga itu best friend 4ever, guys. Yuk kita lihat dari gambar setelah ini!



## Perhatiin baik-baik gambarnya ya, Chingu!

Dari gambar ini, kamu bisa melihat apa aja sih yang ada di dalam Java, serta bagaimana cara kerjanya

Description of Java Conceptual Diagram



Materi Java



JDK, JRE, JVM



Sekarang kita akan mulai mengarungi konsep **IDE** dan **Terminal**. Meluncurrr~



### Aha! aku punya IDE!

Eh, tapi IDE yang dibahas bukan ide yang kita punya pas merencanakan kerja kelompok itu. Beda jauh sama IDE yang ada di KBBI~

IDE dalam konsep Java merupakan singkatan dari Integrated Development Environment, yaitu **software yang menyediakan berbagai fitur untuk software development.**



### “Terus, IDE tuh fungsinya buat apa sih?”

Pertanyaan bagus! Okey, biar makin paham, kita pakai contoh kasus ya.

Sabrina adalah seorang software developer. Pekerjaannya sehari-hari melakukan coding hingga pengetesan code tersebut.

Dari hulu hingga hilir, Sabrina butuh banyak banget tools seperti text editor, code library, compiler sampai tool buat mengetes softwarenya.



Dalam proses pengembangan software, code harus dipindah dari satu tools ke tools lainnya. Ini menyebabkan lamanya waktu dan banyaknya tenaga yang harus dikeluarkan.

Di sinilah **IDE hadir buat menggabungkan semua fungsi di atas menjadi satu.**

Nggak perlu lagi deh tuh pake banyak tools buat banyak tujuan. Cukup satu saja, semua sudah terintegrasi tanpa repot-repot lagi. Canggih, kan?



Bukan cuma itu aja. Paling nggak, IDE punya fitur-fitur keren kayak gini nih!

1. **Interpreter**, berfungsi buat menerjemahkan kode ke byte code.
2. **Build Automation tools**, buat compile suatu project.

Gampangnya, fitur ini bisa dipakai buat nyusun suatu project. Project dating aku sama kamu, misalnya. Eh, bukan, bukan.



### Ada lagi, nih...

3. **Debugger**, buat melakukan tracing atau pencarian terhadap suatu bug.

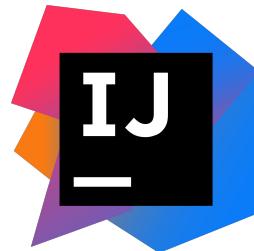
Iya kayak kalo kamu ngilang, aku pasti cari deh!

4. **Source Code Editor**, berfungsi buat menulis code.

Kode dari kamu juga bisa, kok.



Berikut adalah 4 macam IDE yang biasa dipake di Java Development, gengs!



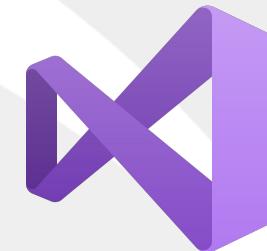
IntelliJ IDE



Eclipse



Net Beans



Visual Studio Code

### Jangan lupa install IntelliJ ya, Sob!

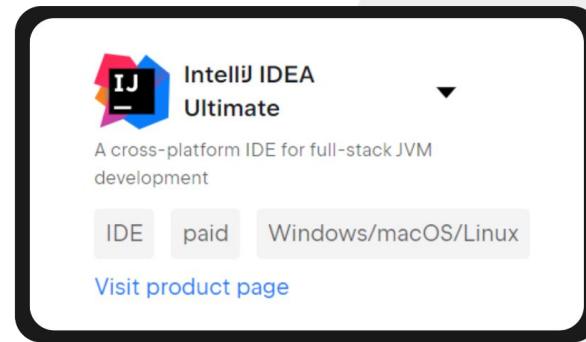
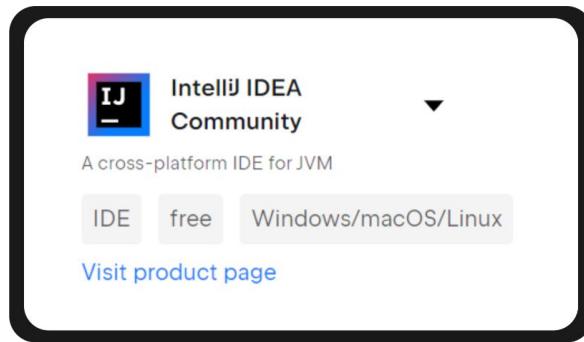
"Lho kok gitu? kenapa nggak yang lain aja?"

Jangan ngambek dulu, dong. Kita pake IntelliJ karena IntelliJ punya kelebihan dibanding IDE yang lain, yaitu:

1. **Intelligent Code Editor**, mendukung code completion dan memudahkan user buat melakukan refactor atau desain setelah coding.
2. **Build Code**, mendukung user buat melakukan build secara otomatis dan mendukung banyak build tools.



Oh iya, IntelliJ ini ada dua versi, yaitu **Community Edition** dan **Ultimate Edition**. Bedanya apa? yuk kita simak~



### Ultimate Edition

Versi full version dan berbayar. Tersedia juga trial version selama 1 bulan. Ultimate Edition sebenarnya bisa digunakan secara gratis selama 1 tahun dengan menggunakan email kampus. Yuk gercep pake email kampus!

### Community Edition

Versi gratis dan hanya memiliki fitur yang sedikit

### Selain 4 IDE tadi, sebenarnya ada IDE lain yang punya fitur tambahan, lho!

Fitur tambahan ini bisa dipakai buat men-develop Spring Framework.

IDE tersebut yaitu [Spring Tool Suite](#), yang pada dasarnya hampir serupa dengan Eclipse.



Oke, sekarang waktunya kita persiapkan  
Instalasi Tools (Java, Maven, IntelliJ)

Are you ready? Go!



## Saatnya kita install tools~

Setelah kita membahas mengenai konsep Java dan IDE, saatnya kita coba install tools yang nantinya bakal dipakai selama belajar di course Back End Java. Yeay!

Seperti yang udah dibahas sebelumnya, kita bakal melakukan instalasi pada 3 tools berikut, yaitu **Java**, **Maven** dan **IntelliJ**.

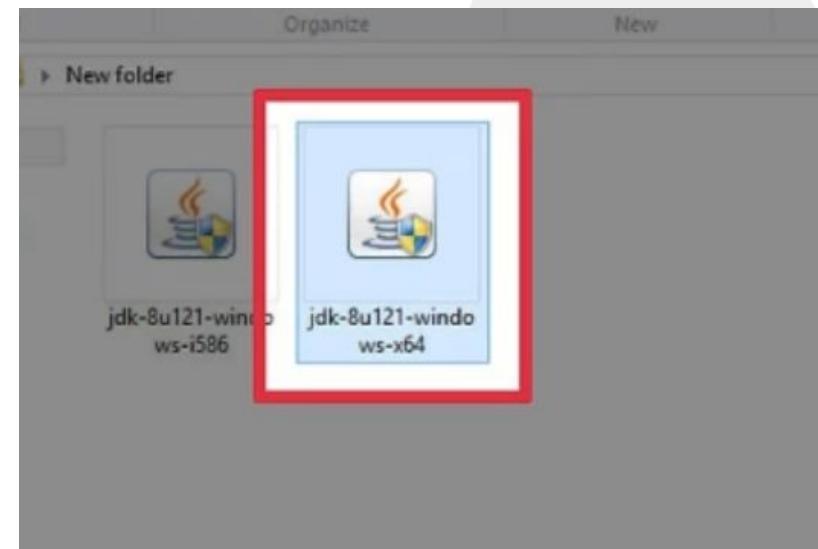
Berangkatt~



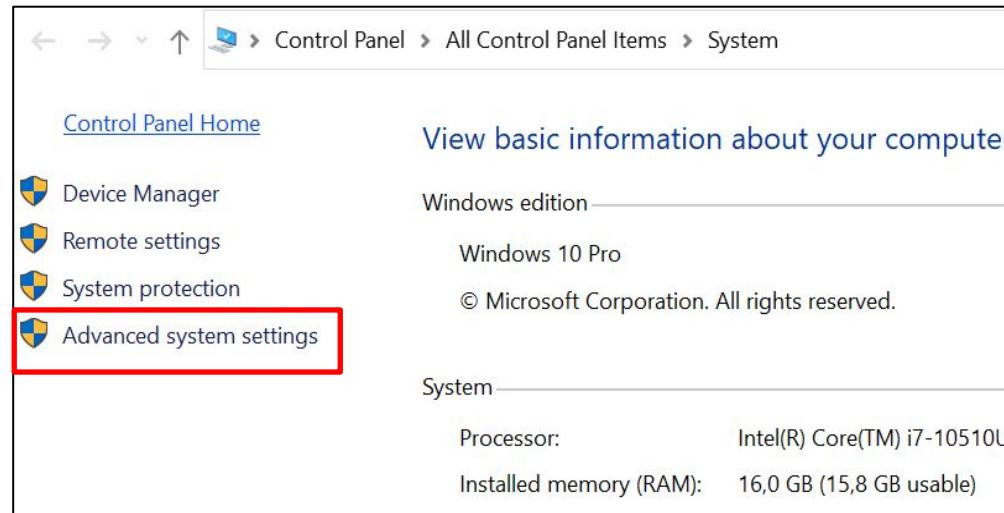
## Instalasi Java

Hal yang perlu disiapkan pertama, yaitu perangkat yang akan kamu gunakan ya! Selanjutnya ikuti step by step berikut, ya!

1. **Download Java SE Development Kit** untuk Java 8 atau yang terbaru (untuk version Java 17) [di sini](#). Pilih sesuai dengan OS yang terinstall.
2. **Double click pada file** yang udah di download.



3. Lakukan instalasi file.
4. Tekan tombol **windows + X**, nanti bakal muncul menu di bagian kiri bawah. Terus **pilih System** pada menu tersebut. **Pilih Advanced system settings**.



## 5. Terus pilih Environment Variables.

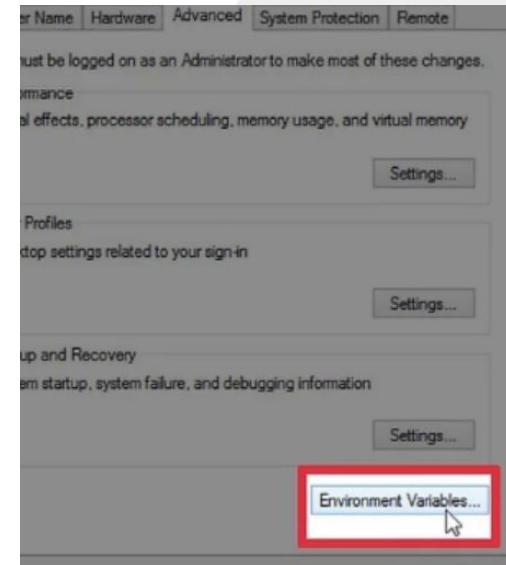
## 6. Set JAVA\_HOME

Pilih New pada System variable.

Masukkan variable name dengan JAVA\_HOME dan variable value dengan lokasi file Java sesuai dengan tempat Java terinstall seperti :

c:\ Program Files \ Java \ jdk1.8.0\_xx \ bin

(tetapi ganti bagian “8.0\_xx” dengan versi SDK yang diinstall, nggak perlu diubah). Lalu Klik “OK.”

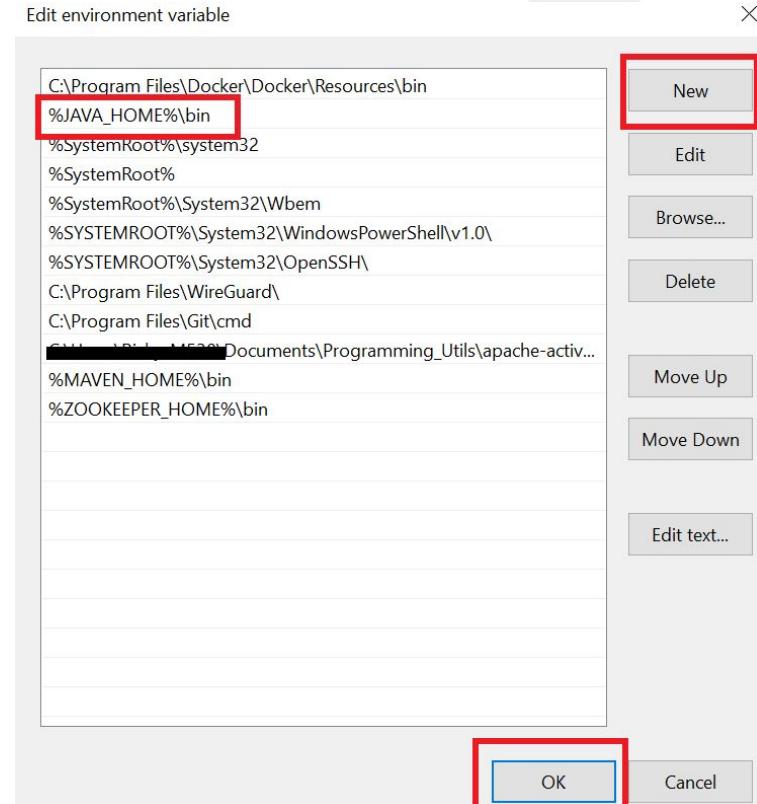
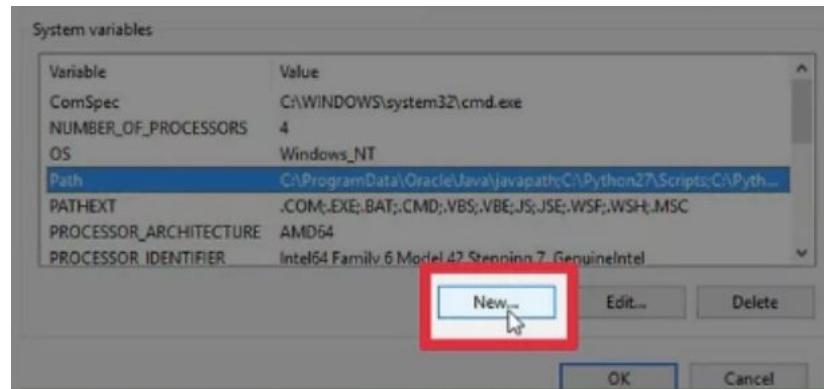


# Java Introduction and Tools Preparation

## Untuk Versi Windows 10 :

Pada bagian System Variables, klik 2x opsi Path lalu pilih New. Tambahkan:

%JAVA\_HOME%\bin, lalu klik "OK"





```
java version "1.8.0_301"
Java(TM) SE Runtime Environment (build 1.8.0_301-b09)
Java HotSpot(TM) 64-Bit Server VM (build 25.301-b09, mixed mode)
```

7. **Buat memastikan apakah Java sudah terinstall dengan benar, buka terminal cmd.** Lalu ketik **java -version**. Kalau berhasil, maka informasi versi pada Java akan muncul seperti gambar berikut

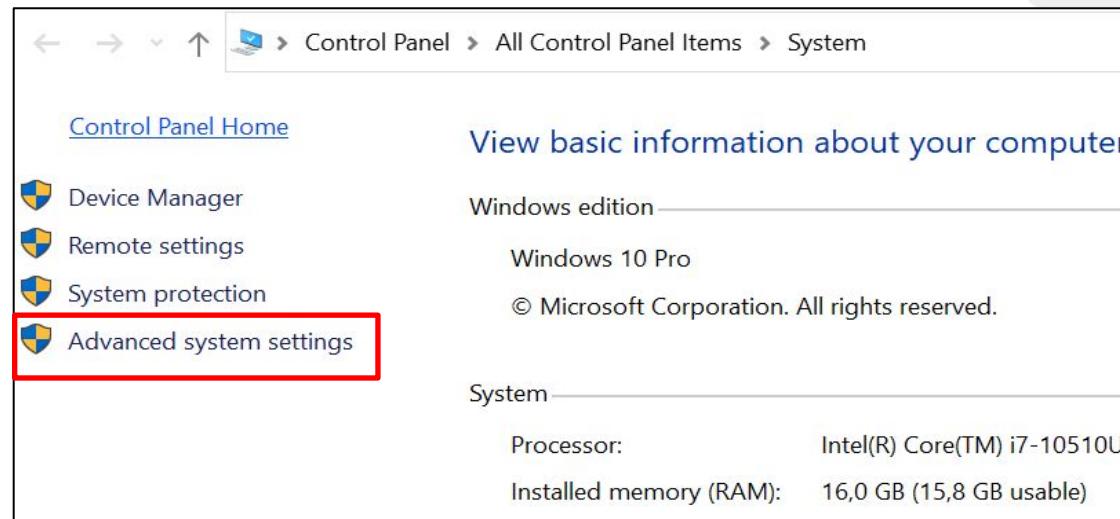
## Instalasi Maven!

Selanjutnya kita bakal mulai install Maven. So, langsung aja yuk kita install!

1. **Download apache Maven** di link yang bisa kamu akses [di sini](#). Buat Windows, pilihlah binary file dengan extension zip.
2. **Extract file zip** hasil download. Letakkan folder hasil extract ke direktori yang diinginkan.

	Link
Binary tar.gz archive	<a href="#">apache-maven-3.8.4-bin.tar.gz</a>
Binary zip archive	<a href="#">apache-maven-3.8.4-bin.zip</a>
Source tar.gz archive	<a href="#">apache-maven-3.8.4-src.tar.gz</a>
Source zip archive	<a href="#">apache-maven-3.8.4-src.zip</a>

3. **Tekan tombol windows + X**, nanti bakal muncul menu di bagian kiri bawah. **Pilih System** pada menu tersebut. **Pilih Advanced system settings**.

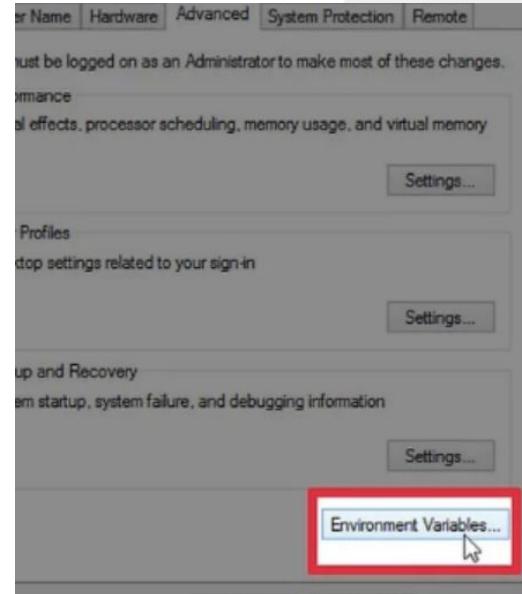


## 4. Pilih Environment Variables

## 5. Set MAVEN\_HOME

Pilih New pada System variable. Masukkan variable name dengan MAVEN\_HOME dan variable value dengan lokasi file apache maven seperti:

C:\Users\any\_user\Downloads\apache-maven-3.8.1-bin\apache-maven-3.8.1

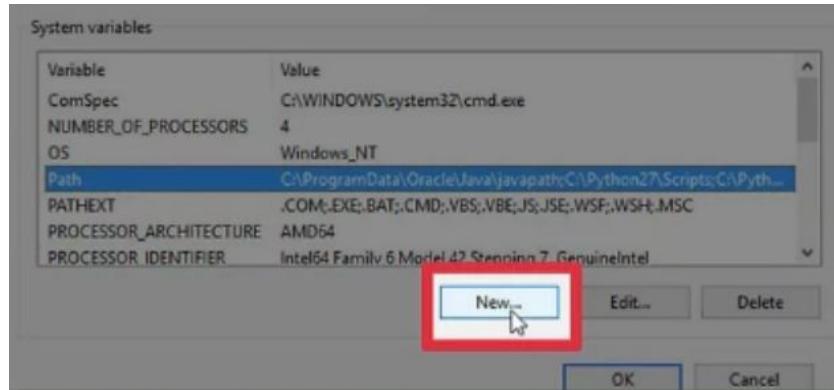


# Java Introduction and Tools Preparation

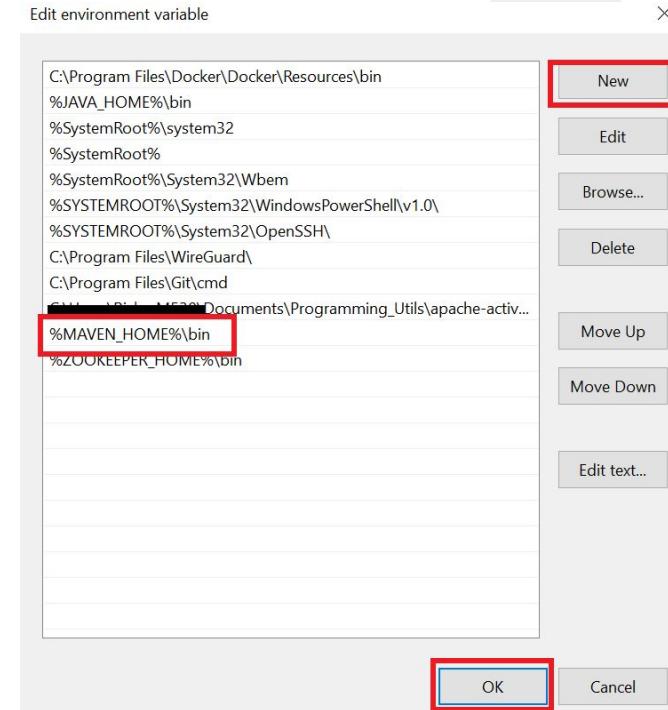
## Untuk Versi Windows 10 :

Pada bagian System Variables, klik 2x opsi Path lalu pilih New. Tambahkan:

%MAVEN\_HOME%\bin, lalu klik "OK".



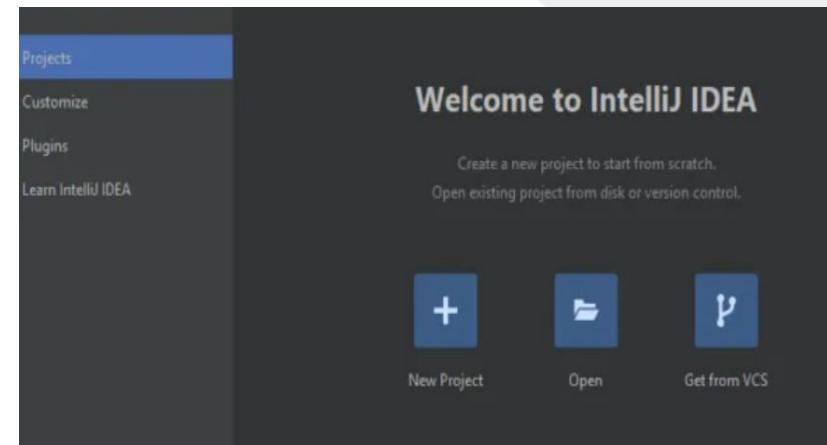
**Buka terminal dan ketik mvn -version** buat memastikan jika Apache Maven telah terinstall.



## Instalasi IntelliJ Idea Ultimate Edition

Tools terakhir yang wajib di install nih!

1. **Download IntelliJ** di link yang bisa kamu akses [di sini](#) dan sesuaikan dengan OS kamu.
2. **Lakukan proses instalasi** dengan setting default.
3. Setelah proses instalasi selesai, **buka IntelliJ Idea** untuk memastikan apakah udah terinstall dengan baik.





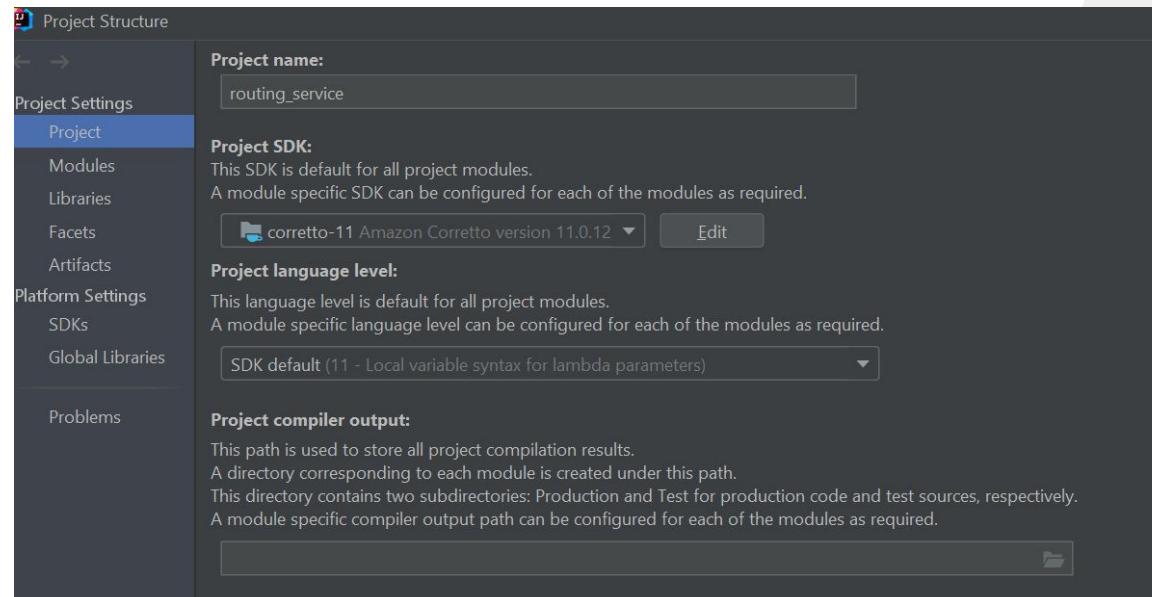
### Catatan nih, Sob!

Sebenarnya kalau udah melakukan instalasi IntelliJ, kita bisa mengatur SDK dan Maven berdasarkan project kita.

Tapi, buat mempelajari dasar pemrograman Java, kita bakal pake command terminal buat memperkuat pemahaman tentang kompilasi java.

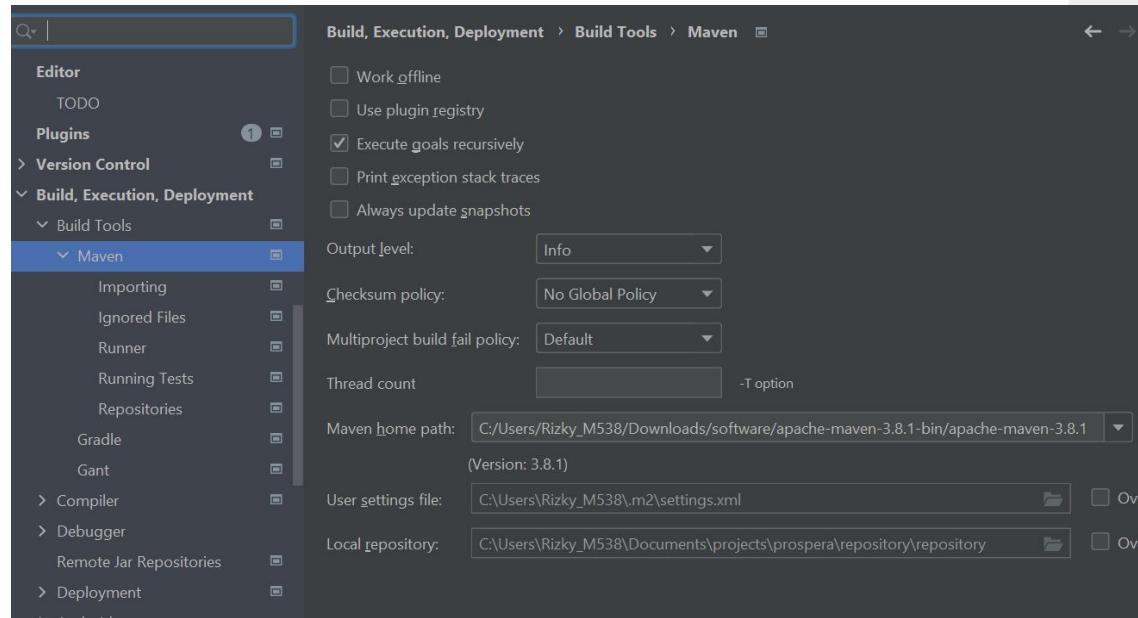
## Berikut cara untuk mengatur SDK

File > Project Structure > Project Settings > Project



## Kalau ini tampilan buat mengatur Maven Home di IDE~

File > Settings > Build, Execution, Deployment > Maven



Okey, sekarang waktunya kita mengulik Maven lebih detail.

Pada Maven, ada Build Tool Automation yang dinamai **Maven Apache**.

Apa itu Build Tool Automation? Tenang, selain belajar definisinya kita coba juga pelajari cara implementasinya, ya!



### Belum kenal berarti belum sayang~

Kenalin, ya, ini namanya Apache Maven yang merupakan **Build Tool Automation** untuk Java.

Build Tool Automation ini berupa struktur project dan berbagai command Maven.

Apache Maven saat ini widely used atau banyak digunakan pada Java development.

Selain Maven, build tool automation lainnya dari Java salah satunya adalah Gradle. Dokumentasi dari Apache Maven tersedia [di sini](#)

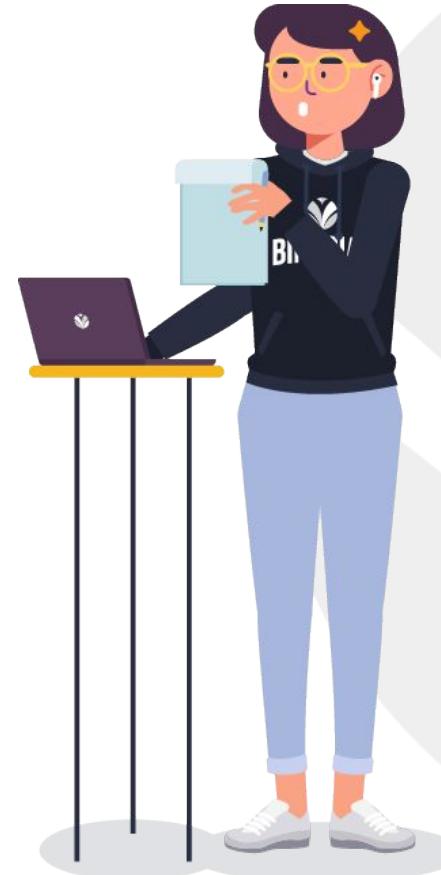


### Kalau pakai Maven, kita bisa ngelakuin 3 hal kaya gini nih~

Bukan sembarang program, Maven punya keuntungan yang bisa kita manfaatin, lho.

1. Kompilasi source code dengan mudah.
2. Melakukan Unit Testing.
3. Menambahkan library yang dibutuhkan.

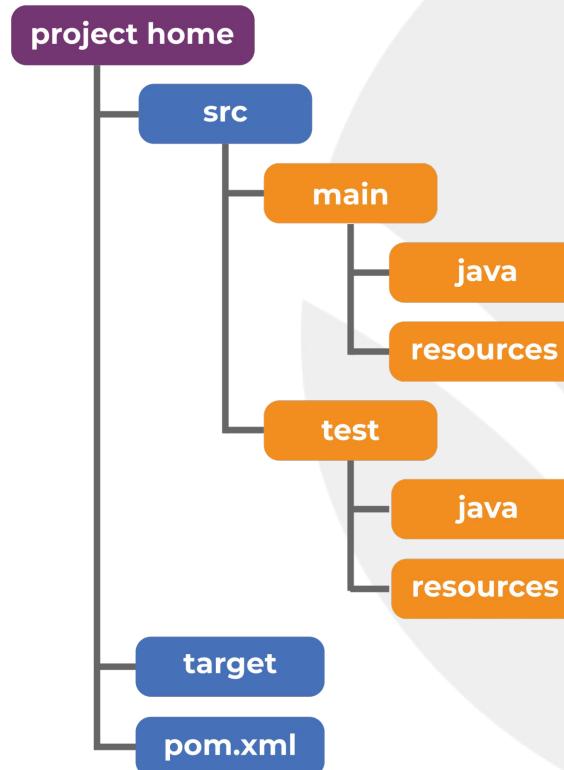
Kalau pakai command-command Maven, kita bisa pakai command line atau pun menggunakan IDE.



## Di mana ada program, di situ ada strukturnya!

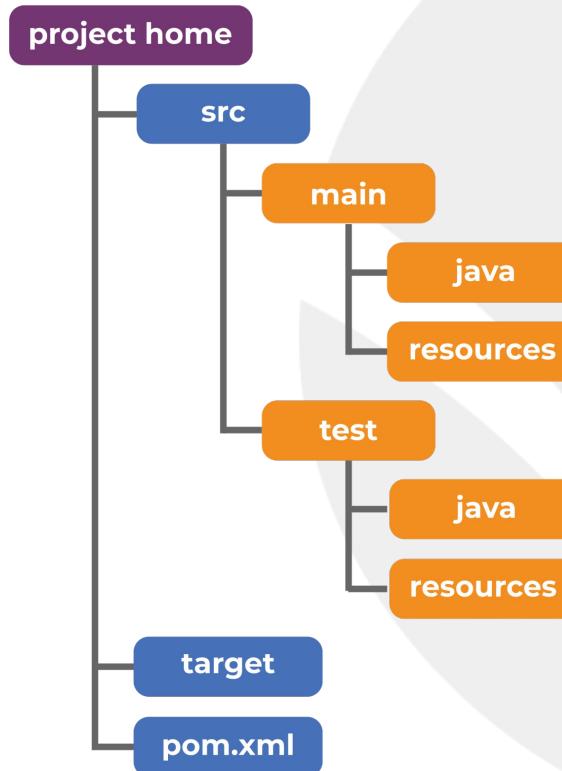
Di samping adalah gambaran dari struktur Maven Project. Yang pasti nggak ribet, nggak bikin pusing~

- **src** merupakan direktori source code dimana kita bisa menulis business logic.
- **test** merupakan direktori buat menulis unit testing.
- **java** merupakan direktori buat menyimpan semua code dalam java, sedangkan **resource** buat menyimpan file-file yang bakal ikut dicompile bareng project.



- **target** adalah hasil compile yang berupa JAR, WAR atau .class.
- **pom.xml**, yaitu file buat ngatur kompilasi project .pom yang merupakan singkatan dari Project Object Model. POM ini berbentuk file xml.

Selain struktur, ada juga nih yang harus kita pahami dari Maven Project. Geser dulu yuk slide-nya~



## Jalan-jalan naik ojek. Yuk kita bahas Tag yang ada di Maven Project~

Nggak nyambung? Biarin aja, yang penting sekarang kita bakal cari tahu beberapa Tag yang perlu diketahui.

1. **<dependencies>** yaitu Tag untuk menambahkan library. Salah satu library yang bisa kita cari yaitu mvn repository yang bisa kamu akses [di sini](#).

Contoh untuk nambahin library kayak gambar di samping. Jangan lupa letakkan di dalam tag <dependencies> ya



```
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-actuator</artifactId>
<version>2.6.3</version>
</dependency>
```

### 2. <build>

Tag yang digunakan untuk mengatur struktur dari direktori. Jadi, buat ngubah struktur dari project, kita bisa mengatur segala tag yang ada di dalam tag <build>.

Misalnya kita mau menambahkan file resource, maka kita harus melakukan deklarasi pada tag ini.



**Nggak sampe situ aja, kita juga bisa meng-compile aplikasi dengan command atau perintah, lho!**

Berikut adalah daftar maven commands yang sering digunakan:

1. **mvn compile**, buat mengcompile code.
2. **mvn package**, buat menghapus hasil compilation sebelumnya.
3. **mvn install**, buat memasang library yang dibutuhkan agar terbaca oleh project.
4. **mvn test**, buat menjalankan unit testing.



- |   |                                     |
|---|-------------------------------------|
| 1 | <input checked="" type="checkbox"/> |
| 2 | <input checked="" type="checkbox"/> |
| 3 | <input type="checkbox"/>            |
| 4 | <input type="checkbox"/>            |



Sipp deh! Pembahasan maven commands tadi sekaligus jadi materi terakhir kita di chapter 1 ini.

Gimana perasaan kamu? Seruu? Masih belum familiar?  
Nggak papa, kok. Namanya juga belajar pelan-pelan~

**So, apa nih gengs yang kamu dapatkan setelah kenalan dengan Back End Java?**

Kalau Sabrina sih jadi paham bahwa menjadi seorang Back End Engineer itu bukan sekedar ngoding aja, tapi juga butuh hard skill dan soft skill untuk mendukung dalam bekerja.

Kalau menurut kamu, apa aja sih keterampilan yang penting dimiliki oleh seorang Back End Engineer?



Yuhuuu! Kamu udah berhasil  
menamatkan Chapter 1 Topic 1 🎉

Selanjutnya, kita bakal bahas tentang  
Hands on Java for the First Time.

Penasaran kayak gimana? Cus langsung  
ke topik selanjutnya~

