**Pengantar Java**

## **Apa itu Java?**

Java adalah salah satu bahasa pemrograman yang populer saat ini. Java adalah bahasa yang “general purpose, concurrent, class-based, dan object-oriented” tutur penciptanya, **James Gosling.**Dengan Java, developer bisa membuat aplikasi yang bersifat WORA (write once, run anywhere), artinya cukup sekali kita menuliskan kode maka hasilnya yang berupa aplikasi, dapat berjalan di banyak platform..

Compiling di Java mengubah Java code menjadi Java bytecode. Dengan menggunakan Java bytecode, kita lebih mudah memproses porting ke platform lainnya via Java Virtual Machine (JVM)

Sintaks yang digunakan Java berasal dari C dan C++ dengan lebih sedikit fasilitas low-level jika dibandingkan dengan keduanya. Java memiliki automatic garbage collector yang berfungsi untuk mengelola memori secara otomatis. Automatic garbage collector akan secara otomatis membebaskan memori yang sudah tidak digunakan, dengan tujuan menghindari kebocoran memori (memory leaks). Otomasi ini mengurangi beban developer dalam menuliskan kode untuk membebaskan memori secara manual. Inilah yang membedakan Java dengan C dan C++ (contoh metode free di C dan metode delete di C++)

## **Mengapa Java?**

Selain fitur portability-nya, Java adalah bahasa yang mature (dewasa) dengan library yang relatif lengkap. Pada 2016, Java menjadi salah satu bahasa pemrograman yang paling banyak digunakan di dunia. Selain itu, Java juga merupakan bahasa resmi untuk membuat aplikasi Android. Intinya Java dipandang sebagai bahasa yang akan selalu relevan dengan perkembangan di masa yang akan datang.

Untuk mengetahui lebih lanjut tentang Java Anda dapat melihat tautan berikut ini:

* [Go Java](https://go.java/)
* [Introduction to Java](https://www.oracle.com/java/technologies/introduction-to-java.html#943)

## **Tools**

Sebelum berperang, kita harus siapkan senjatanya terlebih dahulu. Demikian halnya dengan programming. Sebelum coding maka kita harus memastikan tools apa yang akan kita pakai. Tools pertama yang terpenting adalah Java Development Kit (JDK).

### **Java Development Kit (JDK)**

Java Development Kit adalah tools yang digunakan seorang developer dalam proses developing, debugging, dan monitoring suatu aplikasi Java. Di dalam JDK juga terdapat Java Runtime Environment yang digunakan untuk menjalankan suatu aplikasi Java.

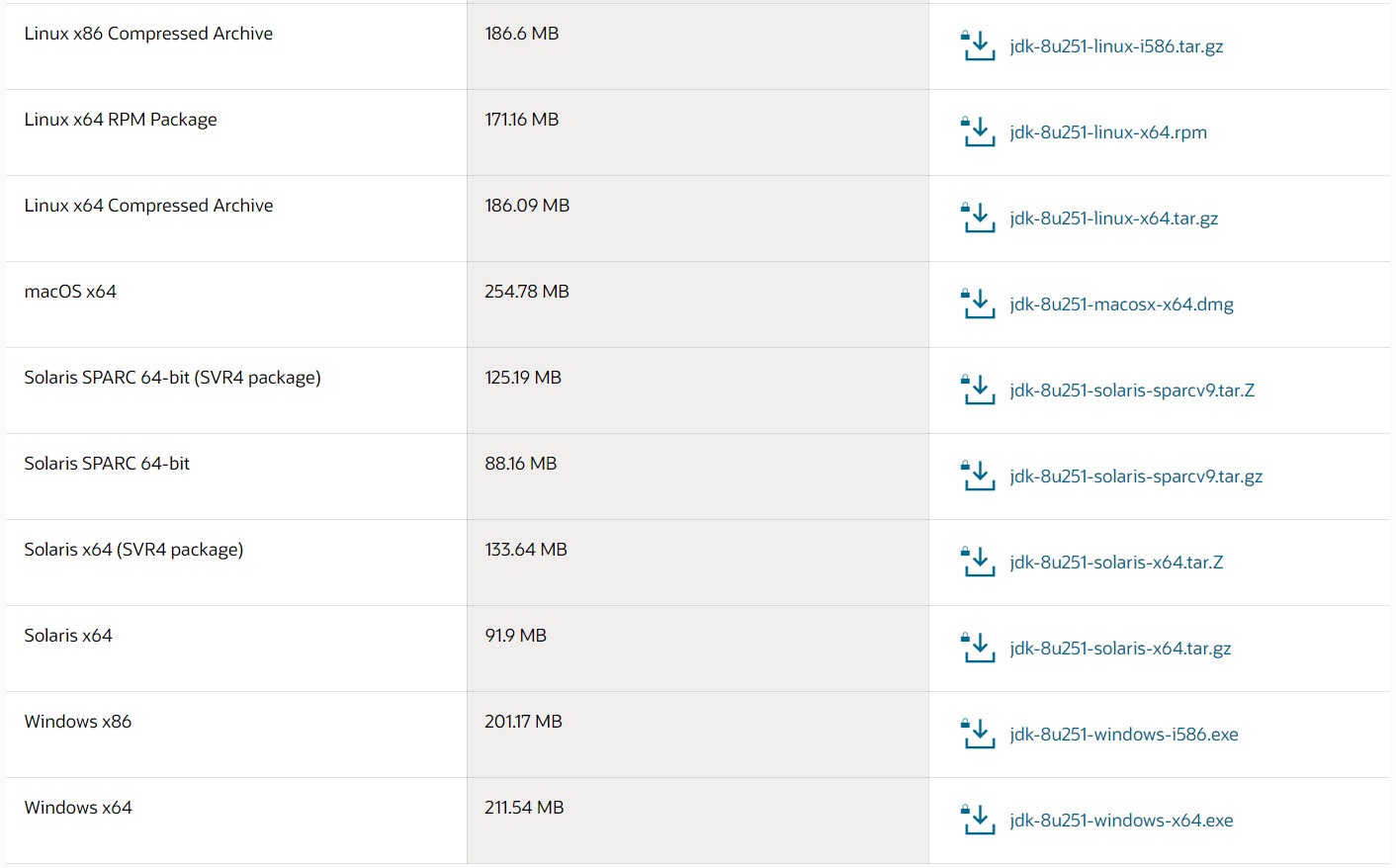
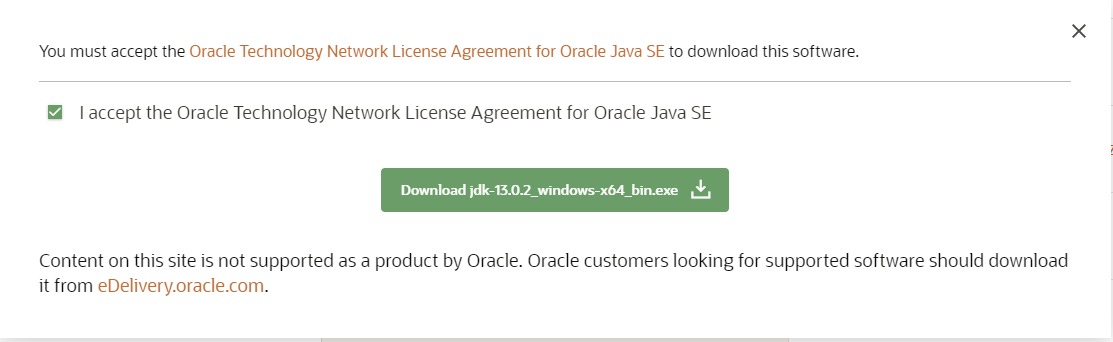
#### **How to install JDK**

Instal Java Development Kit yang bisa kita dapatkan pada tautan berikut:

* <https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html>

Biasanya muncul pertanyaan, "Apakah JRE cukup?" Tidak, JRE adalah Java Runtime Environment yang berfungsi sebagai Virtual Machine untuk menjalankan program Java. Sedangkan JDK merupakan Java SE Development Kit, di mana JRE juga terdapat di dalamnya. Dan yang lebih penting adalah di dalamnya terdapat compiler dan tools untuk membuat dan compile program.

Sederhananya JRE untuk menjalankan program, sedangkan JDK untuk membuat program. Mari kita mulai dengan proses instalasi dari JDK dari Oracle.

1. Langsung saja buka tautan di atas menggunakan browser Anda. Pilihlah link download yang sesuai dengan OS yang Anda pakai.  
   [](https://www.dicoding.com/academies/60/tutorials/1791?from=1788)
2. Jangan lupa untuk mencentang Accept License Agreement dan klik tombol Download.  
   [](https://www.dicoding.com/academies/60/tutorials/1791?from=1788)
3. Anda akan diminta untuk login terlebih dahulu, silakan mendaftar dulu jika Anda belum memiliki akun.
4. Setelah proses mengunduh selesai, langsung install ke gawai Anda dan ikuti petunjuknya sampai selesai.

#### **Update PATH Environment Variable (Opsional)**

Anda dapat menjalankan JDK tanpa pengaturan enviroment PATH atau secara opsional mengaturnya agar dapat menjalankan berkas java ((javac.exe, java.exe, javadoc.exe, dan sebagainya) dari directory manapun. Jika Anda tidak mengatur variabel PATH, maka Anda perlu menentukan jalur lengkap/lokasi ke berkas yang dapat dieksekusi setiap kali Anda menjalankannya, seperti:

1. C:\> "C:\Program Files\Java\jdk1.7.0\bin\javac" MyClass.java

"C:\Program Files\Java\jdk1.7.0\bin\javac" merupakan lokasi dari JDK yang biasanya akan secara default ada. Di sini contoh versi yang digunakan adalah jdk 1.7.0. Di dalamnya ada folder bin dan javac.  
  
Lalu bagaimana cara mengaturnya? Caranya mudah kok, yakni:

1. Klik **Start**, kemudian pilih **Control Panel**, dan pilih **System**.
2. Kemudian klik **Advanced** dan pilih **Environment Variables**.
3. Tambahkan lokasi folder **bin** dari JDK untuk **variabel PATH**di Variabel Sistem. Berikut ini adalah nilai tipikal untuk variabel PATH:
   1. C:\WINDOWS\system32;C:\WINDOWS;**C:\Program Files\Java\jdk1.7.0\bin**

Tentu sesuaikan dengan lokasi dari java yang ada dalam PC Anda.

1. Jika sudah Klik OK atau Apply untuk mengakhirinya.
2. Untuk memastikan apakah gawai sudah mengetahui tentang infomasi jdk, Anda bisa menjalankan perintah java -version (Windows) atau javac -version (Linux/MacOS).

Tentu beda OS akan mempengaruhi cara pengaturan PATH Environment Variable.

## **Develop, Debug, dan Monitor**

Apa itu develop, debug dan monitor? Develop adalah proses penyusunan baris per baris kode sampai menjadi satu aplikasi utuh yang memiliki suatu fungsi. Tujuan akhir yang ingin dicapai akan mempengaruhi kompleksitas proses development. Tak peduli betapa rumitnya proses development, kesalahan mungkin terjadi. Kesalahan ini kita sebut sebagai bug, yakni  eror yang dapat membuat aplikasi kita tidak dapat berjalan seperti keinginan. Di sinilah muncul istilah debug, yakni proses mencari dan menghilangkan eror tersebut di dalam aplikasi untuk kemudian memperbaikinya. Dan terakhir, monitor adalah proses untuk mengamati jalannya aplikasi dari yang kita buat. Biasanya ini berhubungan dengan performa aplikasi dengan environment di mana ia dijalankan. Intinya, JDK adalah tools wajib yang harus digunakan oleh seorang developer Java untuk membuat suatu aplikasi dari nol hingga selesai.

## **Integrated Development Environment (IDE)**

Untuk membuat suatu aplikasi Java sebenarnya cukup dengan teks editor dan command line. Editor digunakan untuk menuliskan baris kodenya sementara command line digunakan untuk menjalankan perintah Java. Akan tetapi, saat ini sudah ada IDE yang membantu tidak hanya dalam proses developing, bahkan hingga optimasinya.

Biasanya di dalam IDE terdapat source code editor, build automation tools, dan debugger. Fitur lainnya adalah code completion yang sangat berguna terkait efisiensi penulisan kode. IDE juga biasanya dilengkapi dengan version control system yang bermanfaat untuk versioning suatu aplikasi. Beberapa IDE juga memiliki tools untuk membuat tampilan (layout) dalam Graphical User Interface (GUI), sehingga kita bisa membuat suatu tampilan dengan cara drag-and-drop.

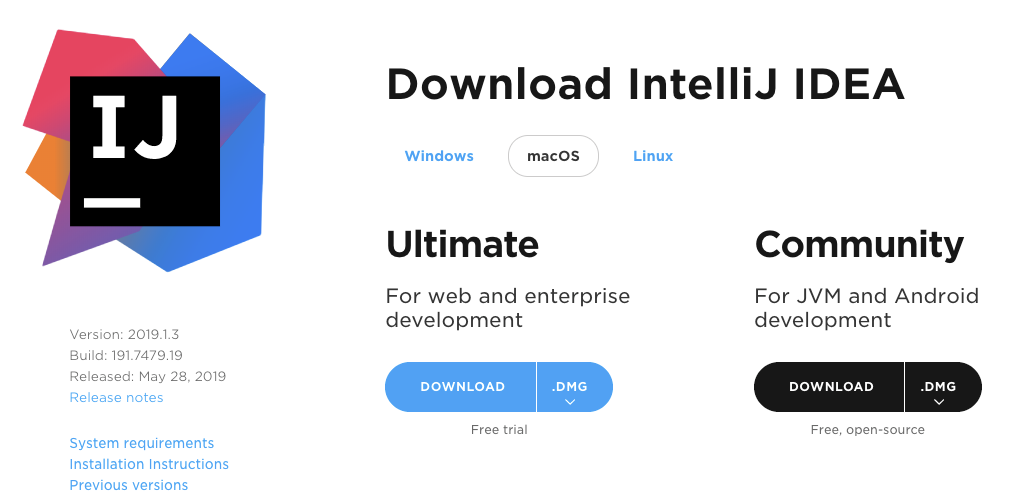
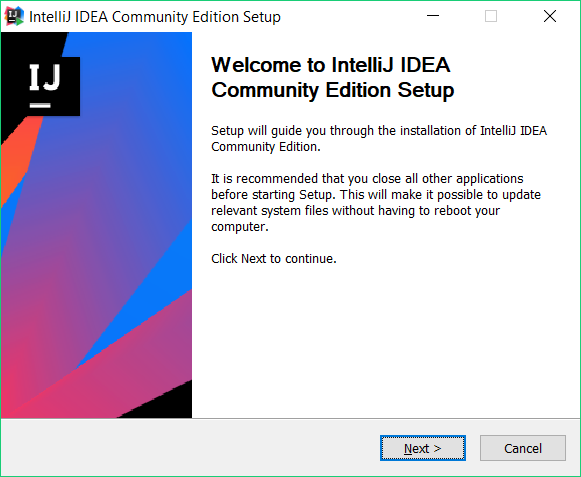
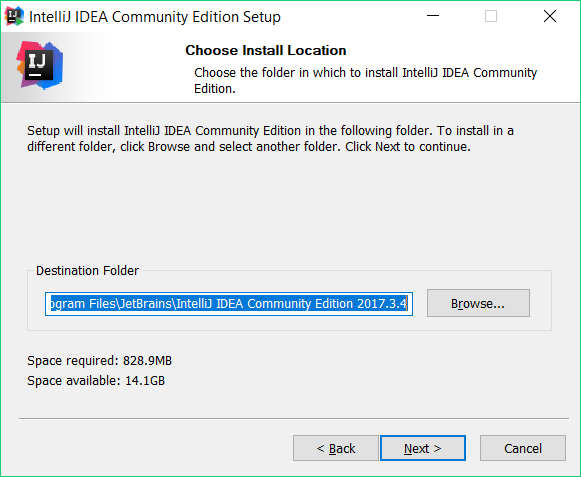
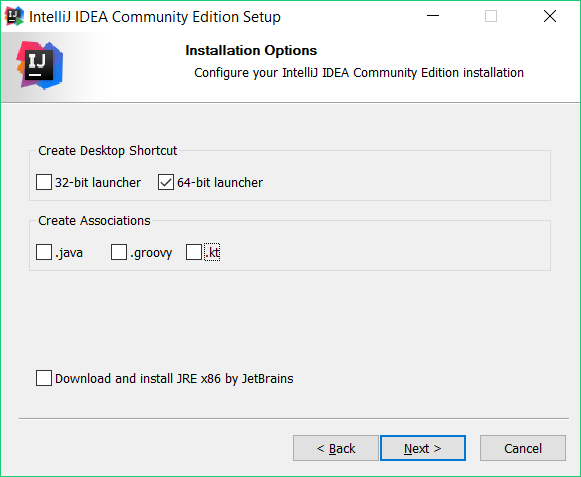
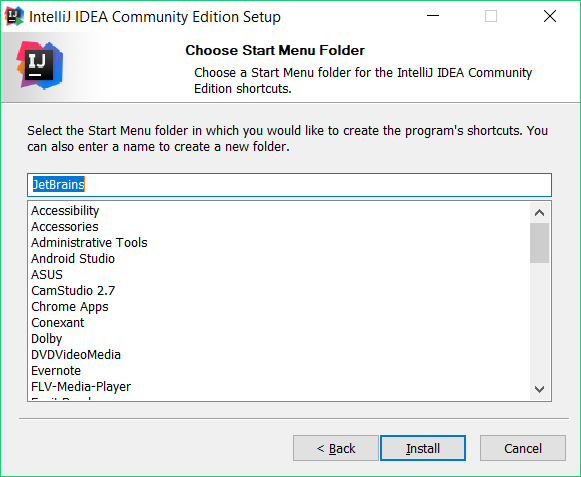
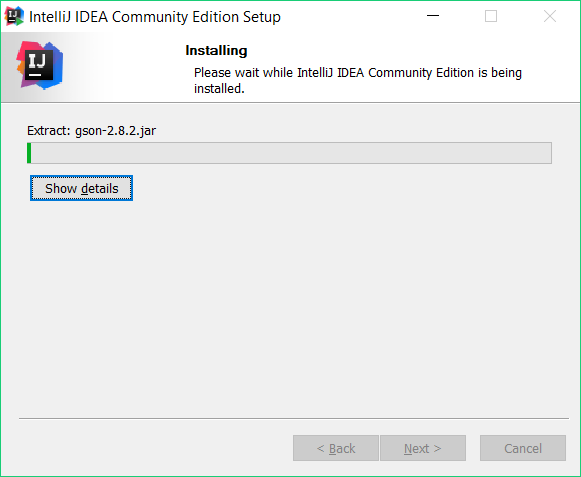
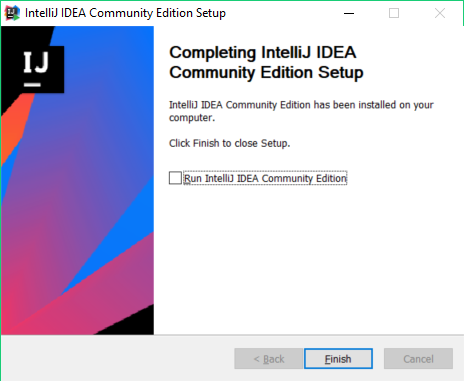
IDE saat ini sudah menjadi tools wajib untuk memaksimalkan produktivitas seorang developer dalam membuat suatu aplikasi. Seperti perumpamaan sebelumnya, IDE adalah senjata developer. Semakin canggih dan tingginya kita menguasai senjata tersebut, semakin ampuh juga efeknya.

Tools yang akan digunakan di dalam kelas ini adalah :

* **OpenJDK**, yakni free open source di bawah lisensi **GNU**General Public License.
* **Intellij**, adalah IDE untuk Java development yang dikembangkan oleh **JetBrains**

Namun demikian, untuk mengikuti kelas ini, penggunaan Intellij dan OpenJDK tidaklah bersifat wajib. Anda boleh menggunakan tools Java lainnya, tetapi lebih disarankan memakai tools bahasan materi ini. Tujuannya, untuk mengurangi potensi kesalahan teknis atau lainnya.

### Cara Install Intellij

1. Unduh Intellij di tautan ini <https://www.jetbrains.com/idea/download/>. (Anda bisa menggunakan versi Community)  
   [](https://www.dicoding.com/academies/60/tutorials/1791?from=1788)
2. Ikuti langkah-langkah di bawah ini untuk proses instalasinya.
3. Jalankan installer yang sudah Anda unduh:  
   [](https://www.dicoding.com/academies/60/tutorials/1791?from=1788)
4. Pilihlah lokasi di mana Anda hendak menyimpan aplikasi tersebut.  
   [](https://www.dicoding.com/academies/60/tutorials/1791?from=1788)
5. Centang pilihan launcher sesuai dengan versi OS yang Anda gunakan (32 atau 64) bit. Perintah tersebut membuat launcher nampak di desktop. Anda pun bisa dengan mudah mengakses Intellij.  
   [](https://www.dicoding.com/academies/60/tutorials/1791?from=1788)
6. Untuk start menu biarkan pilihan default-nya tetap intact.  
   [](https://www.dicoding.com/academies/60/tutorials/1791?from=1788)
7. Tunggu sampai prosesnya selesai.  
   [](https://www.dicoding.com/academies/60/tutorials/1791?from=1788)
8. Saat selesai, klik Finish.  
   [](https://www.dicoding.com/academies/60/tutorials/1791?from=1788)

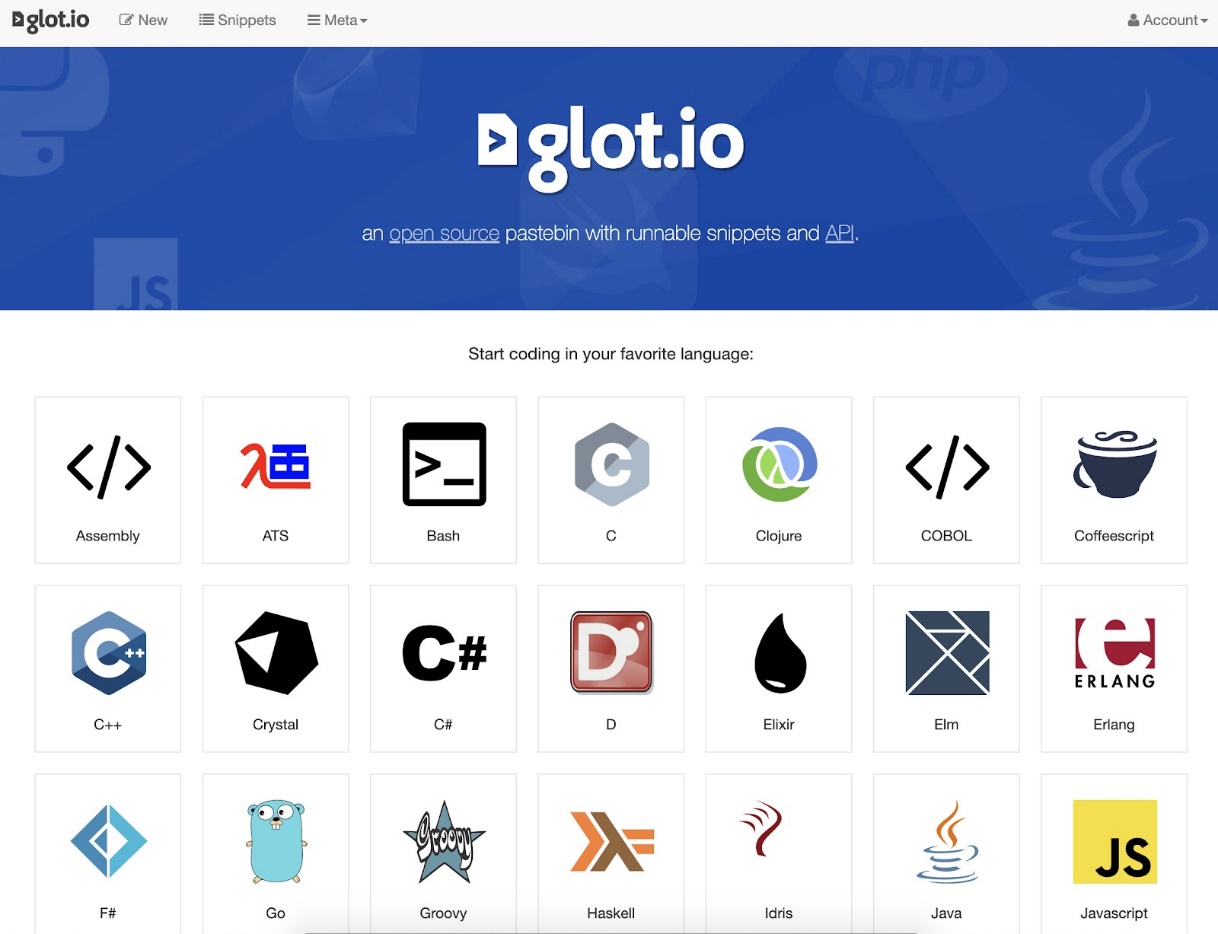
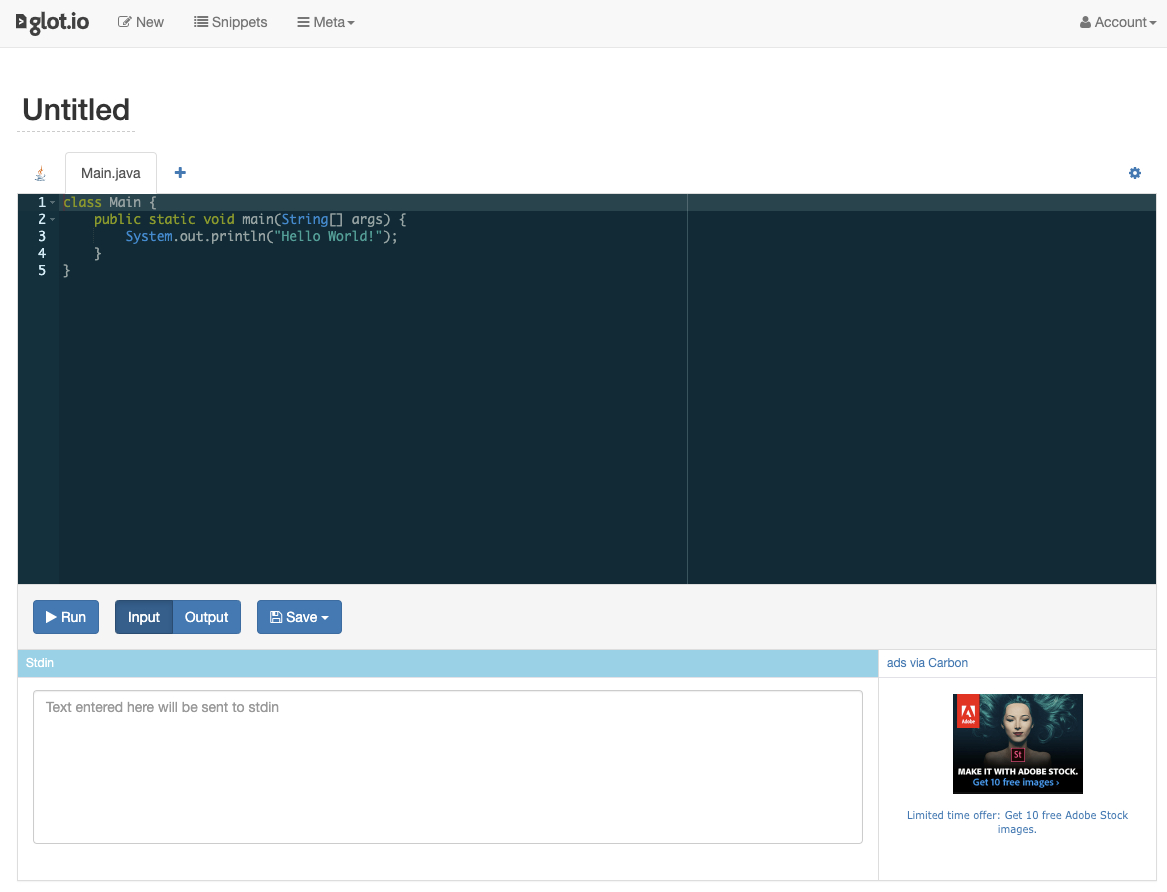
Untuk petunjuk pemakaian yang resmi dari Intellij, silakan lihat tautan ini <https://www.jetbrains.com/help/idea/meet-intellij-idea.html>.

### **IDE Online**

Jika Anda mengalami kesulitan, Anda bisa menggunakan glot.io dan repl.it.

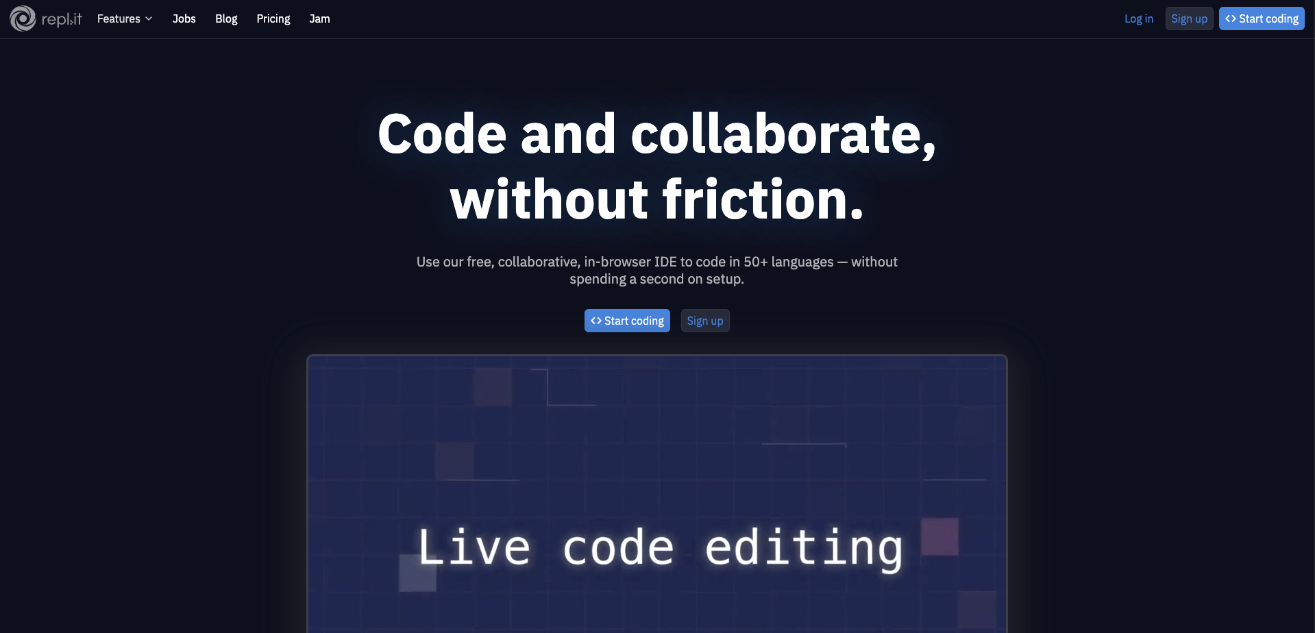
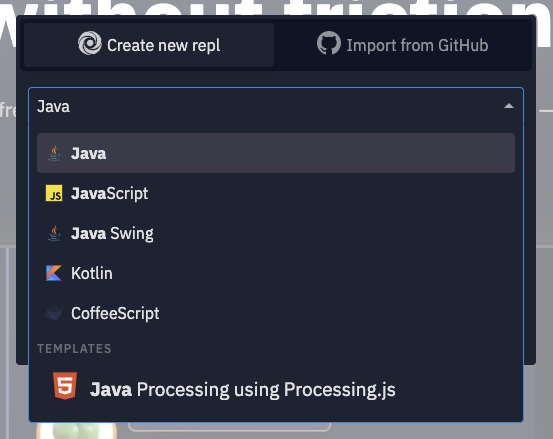
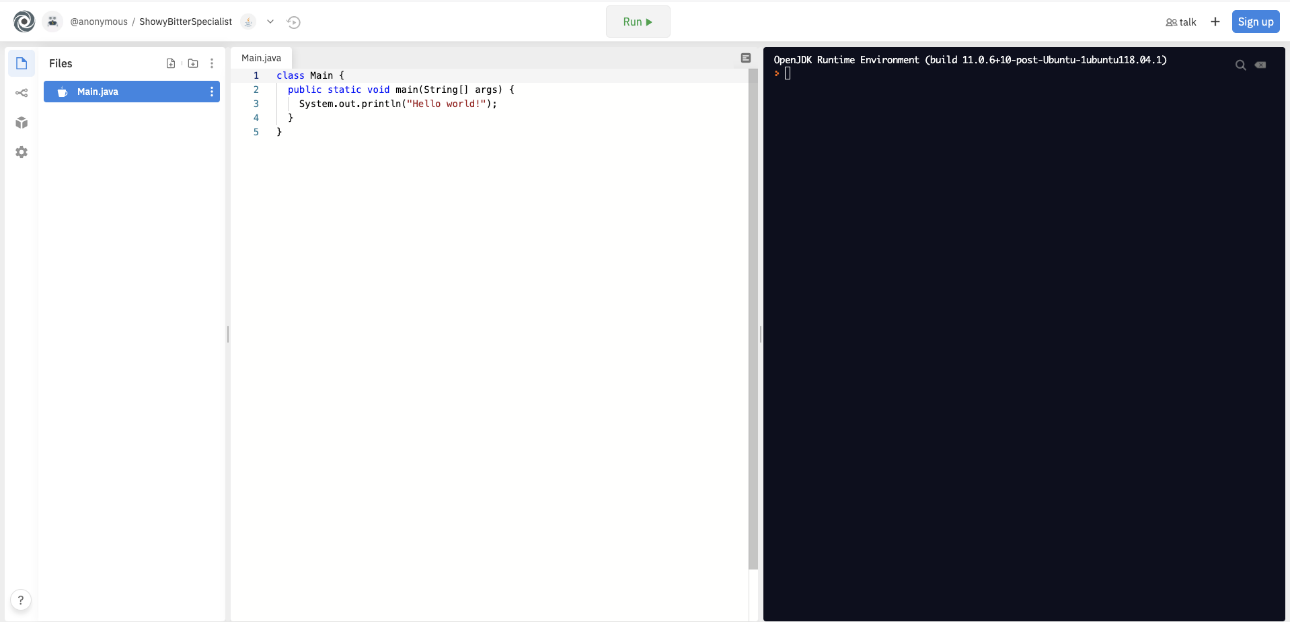
#### **Glot.io**

Untuk glot.io, silakan ikuti cara-cara berikut:

1. Pertama bukalah halaman <https://glot.io/>.  
   [](https://www.dicoding.com/academies/60/tutorials/1791?from=1788)
2. Selanjutnya, pilih bahasa Java karena Anda akan mempelajari bahasa ini.  
   [](https://www.dicoding.com/academies/60/tutorials/1791?from=1788)
3. Selanjutnya, Anda bisa **ubah nama proyek** dan**nama berkas** yang Anda buat dan tentunya Anda bisa memulai menuliskan baris kode di dalamnya.  
   [](https://www.dicoding.com/academies/60/tutorials/1791?from=1788)Terdapat beberapa tombol pada gambar di atas, di antaranya:
   1. tombol **Run** digunakan untuk menjalankan kode,
   2. tombol **Input** digunakan untuk memasukkan teks ke dalam program yang dijalankan,
   3. tombol **Output** digunakan untuk menampilkan hasil dari program yang dijalankan,
   4. dan tombol **Save** digunakan untuk menyimpan berkas yang sudah dibuat.
4. Anda juga bisa membuat beberapa berkas di dalam proyek ini dengan menekan tombol add (+), seperti gambar di bawah:  
   [](https://www.dicoding.com/academies/60/tutorials/1791?from=1788)

#### **Replit**

Sedangkan untuk repl.it, silakan ikut cara-cara berikut:

1. Pertama, buka halaman [repl.it](https://repl.it/).  
   [](https://www.dicoding.com/academies/60/tutorials/1791?from=1788)
2. Selanjutnya klik tombol **Start coding**, kemudian pilih bahasa **Java**.  
   [](https://www.dicoding.com/academies/60/tutorials/1791?from=1788)Setelah itu, klik tombol **Create repl**.
3. Nah dengan begitu, Anda akan disuguhkan halaman untuk menuangkan kode-kode Java Anda.  
   [](https://www.dicoding.com/academies/60/tutorials/1791?from=1788)Di bagian kiri merupakan stuktur project, bagian tengah merupakan tempat menuangkan kode Java, dan bagian kanan merupakan tempat input/output program.

Jadi seperti itulah beberapa tools yang bisa Anda gunakan untuk membangun program Java, baik IDE offline (Intellij Idea) maupun IDE online (Replit dan Glot.io).

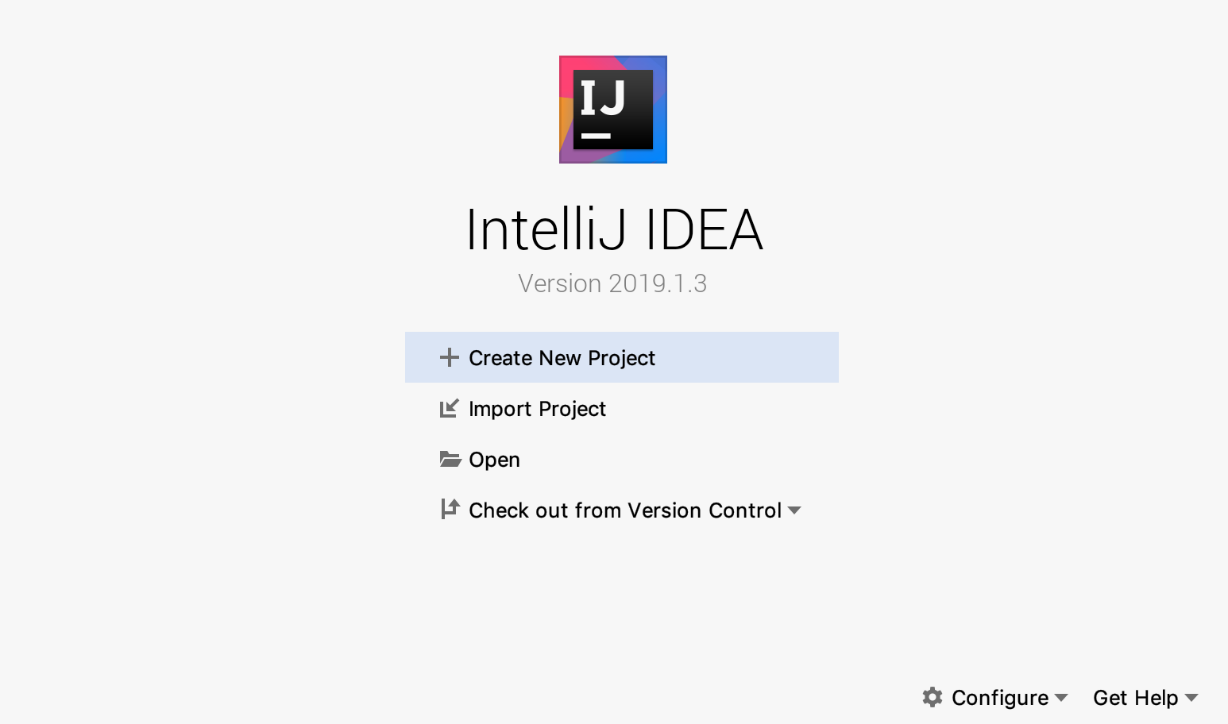
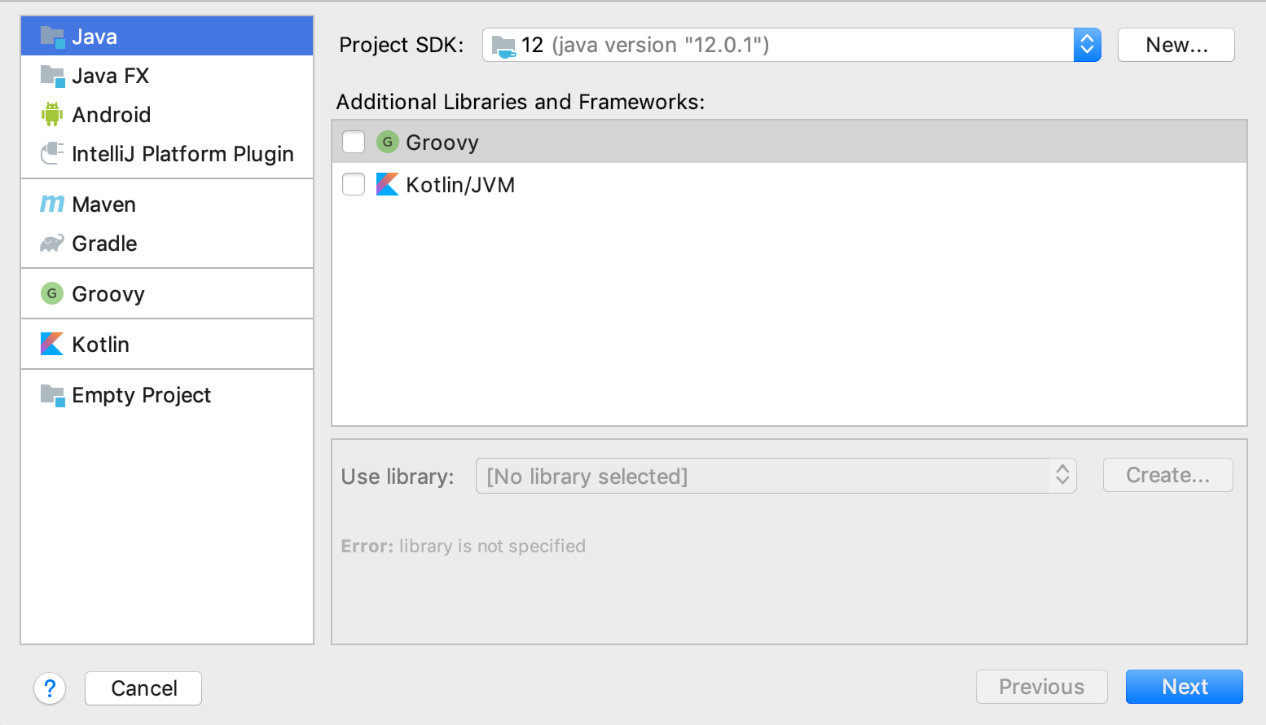
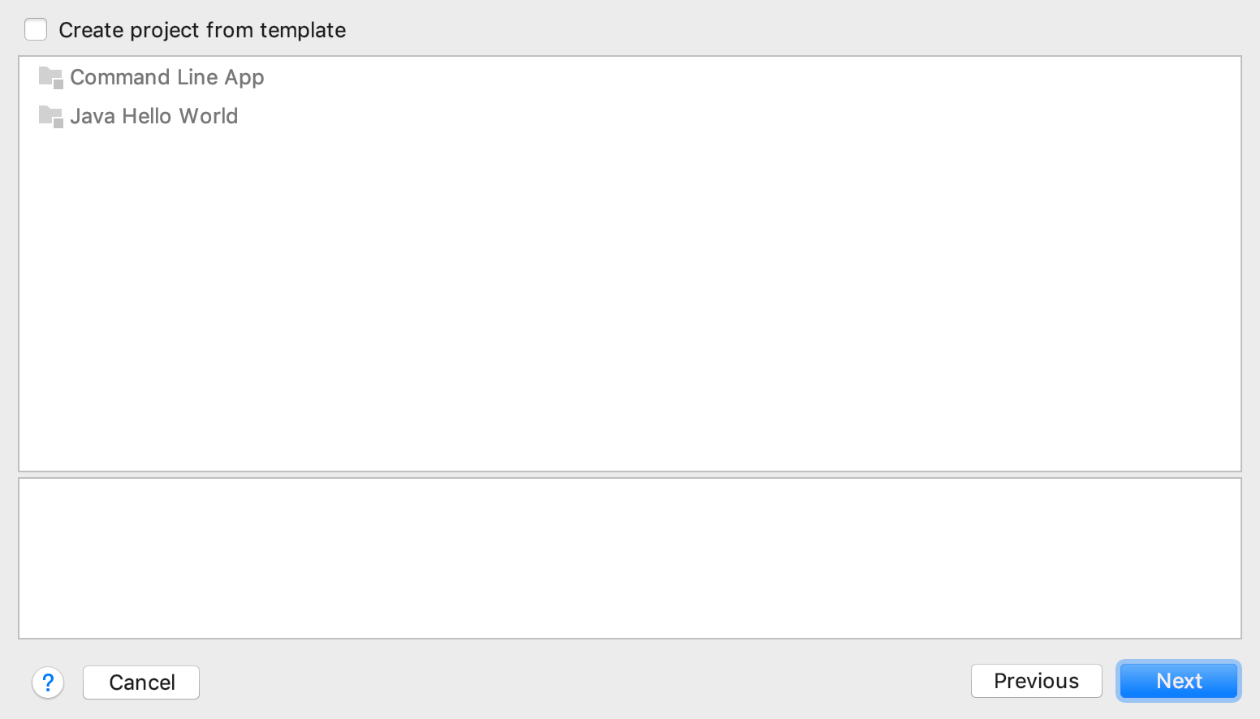
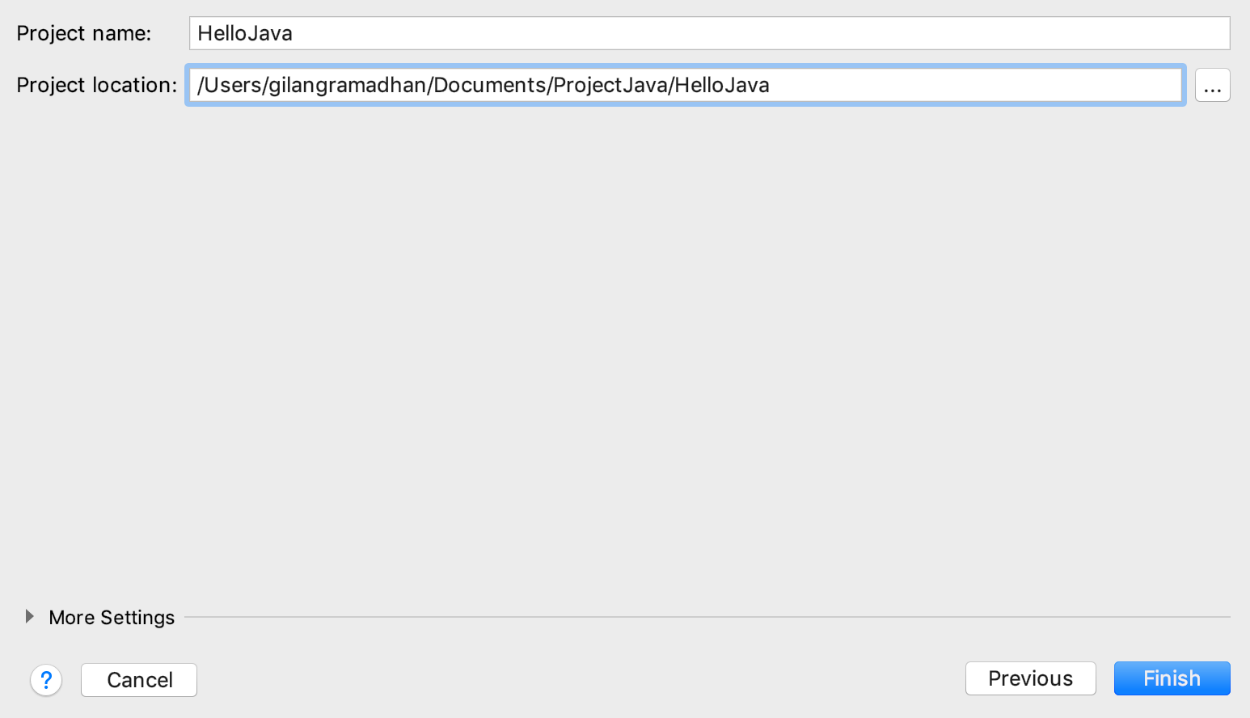
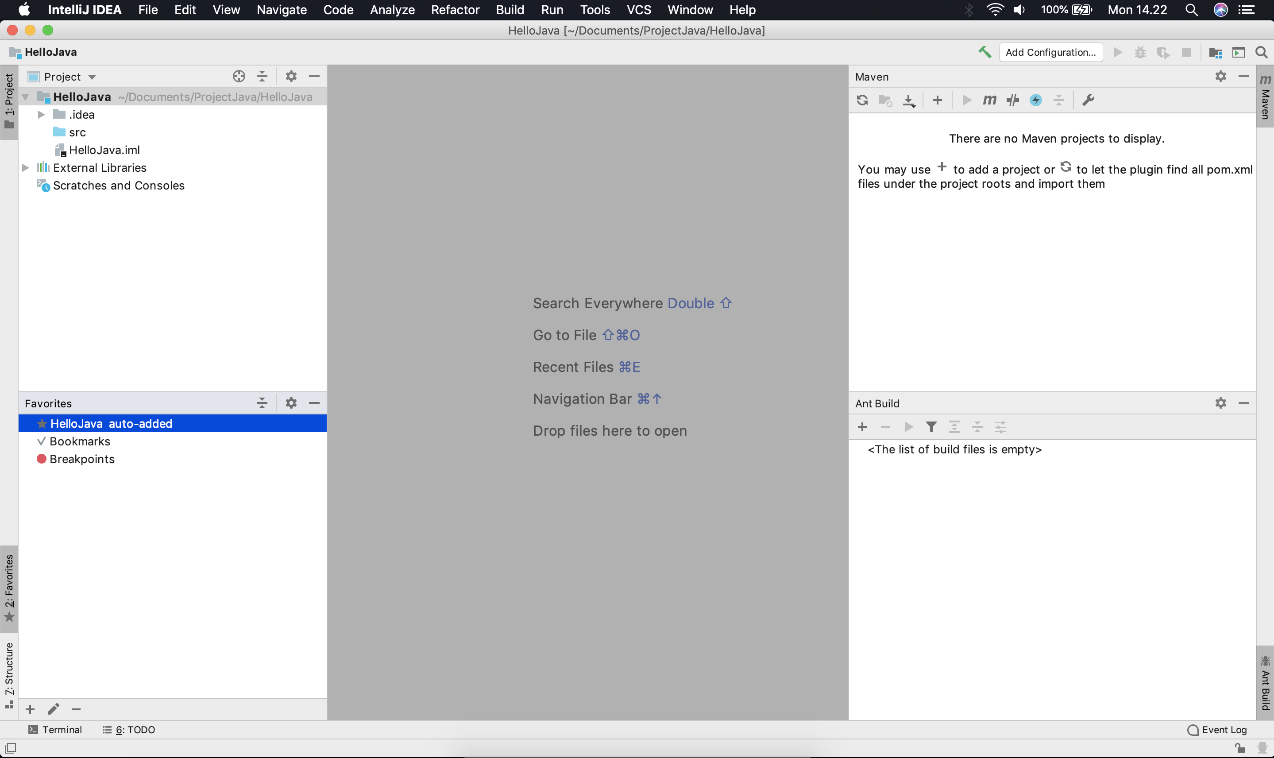
## **Membangun Project Pertama Kali**

### **Tujuan**

Setelah semua tools sudah siap, mari kita buat aplikasi sederhana yang disimpan dalam sebuah file Hello.java. Kita akan menuliskan teks **Hello World!** di layar seperti ini:

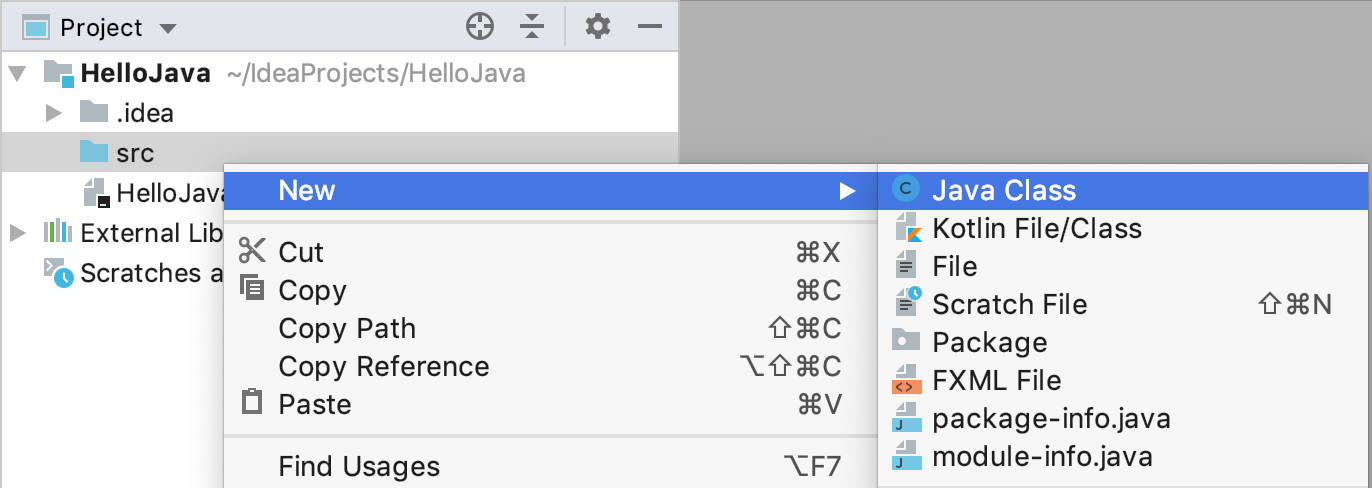
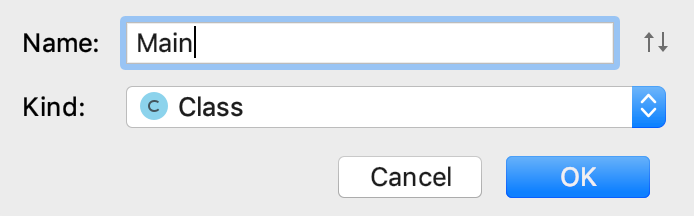
### **Codelab Membuat Proyek Baru**

Ikuti langkah-langkah berikut ini:

1. Buka aplikasi IntelliJ. Akan muncul window setup wizard seperti ini. Pilihlah **Create New Project**.  
   [](https://www.dicoding.com/academies/60/tutorials/1860?from=1791)Di dalam tampilan awal **IntelliJ**, terdapat beberapa pilihan menu, di antaranya:
   1. **Create New Project:**untuk membuat project baru.
   2. **Import Project:**untuk memasukan project yang telah di buat sebelumnya.
   3. **Open:**untuk Membuka folder yang sudah berisi aplikasi yang telah di buat.
   4. **Chek out from Version Control:**digunakan untuk berkolaboratif dengan teman atau rekan kerja Anda.
2. Kemudian pilih **Next**.  
   [](https://www.dicoding.com/academies/60/tutorials/1860?from=1791)
3. Jika terdapat pilihan **Create project from template**, abaikan dan pilih **next**.  
   [](https://www.dicoding.com/academies/60/tutorials/1860?from=1791)
4. Ganti **Project location** ke directory folder yang Anda inginkan. Berilah **Project name**-nya **“HelloJava”.**  
   [](https://www.dicoding.com/academies/60/tutorials/1860?from=1791)
5. Tampilan utama aplikasi adalah seperti ini.  
   [](https://www.dicoding.com/academies/60/tutorials/1860?from=1791)

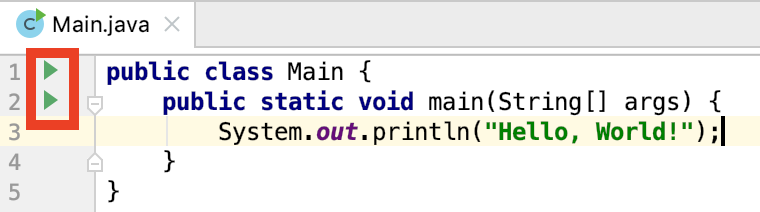
### **Codelab Membuat Kelas Baru**

Mari kita buat berkas baru dengan cara:

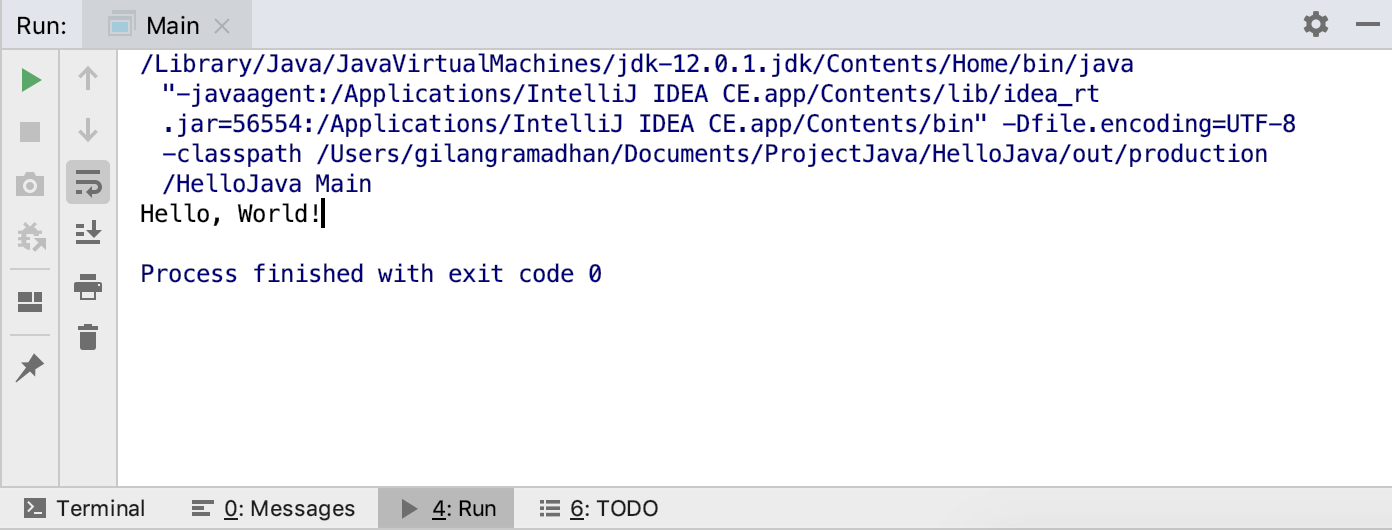
1. Klik kanan di bagian **src** kemudian pilih **New → New Java Class**. Lihatlah gambar di bawah ini:  
   [](https://www.dicoding.com/academies/60/tutorials/1860?from=1791)
2. Setelah itu akan muncul dialog seperti gambar di bawah ini, berilah nama Main.  
   [](https://www.dicoding.com/academies/60/tutorials/1860?from=1791)
3. Tambahkan kode berikut ke dalam kelas Mainyang telah kita buat.
   1. public class Main {
   2. **public static void main(String[] args) {**
   3. **System.out.println("Hello, World!");**
   4. **}**
   5. }

### **Codelab Menjalankan Kelas Main**

Anda bisa menjalankan Aplikasi dengan cara berikut:

1. Tekanlah tanda run dengan **warna hijau** di sebelah kiri editor. Lihat gambar di bawah ini:  
   [](https://www.dicoding.com/academies/60/tutorials/1860?from=1791)
2. Setelah itu akan muncul log pada bagian bawah. Anda juga dapat melihat output-nya.

|  |
| --- |
| Hello, World! |

[](https://www.dicoding.com/academies/60/tutorials/1860?from=1791)

Selesai sudah materi introduction. Di sini kita sudah membahas banyak hal mengenai Java, JDK, IDE dan juga latihan membuat aplikasi **Hello World**. Jika Anda ingin mengetahui arti dari kode di atas Anda dapat membacanya pada tautan [ini](https://docs.oracle.com/javase/tutorial/getStarted/application/index.html).

**Pengantar Java Dasar**

Demi menaklukkan medan pemrograman dengan Java, kita perlu memahami terlebih dahulu seperti apakah Java itu. Bagaimana strukturnya? Bagaimana menggunakan fungsi *basic programming* yang ia miliki? Oleh karena itu kita perlu tahu pemrograman dasar Java yang dikemas dalam materi berikut:

1. **Struktur Dasar**  
   Ini adalah materi paling mendasar dari Java yang wajib Anda kuasai.
2. **Tipe Data**  
   Setiap tipe data digunakan untuk menampung konten yang berbeda. Delapan (8) tipe data dalam Java yang akan kita pelajari adalah int, float, char, long, double, boolean, byte dan short.
3. **String**Selain data primitif, *string* merupakan pokok dasar yang perlu Anda kuasai.
4. **Operator**Operator meliputi operasi aritmatika dasar seperti penambahan, pengurangan, perkalian, dan pembagian. Selain aritmatika dasar, terdapat juga operator-operator lain seperti *assignment*,*unary, equality and relational,*dan*condition.*
5. **Fungsi Input Output**  
   Proses *input* dan *output* merupakan hal yang penting agar kita dapat berinteraksi dengan program. Di sini kita akan belajar beberapa komponen yang dapat menerima *input* dari *user* dan komponen yang dapat menampilkan data ke *user*.
6. **Array**Array adalah kumpulan data yang dimasukkan ke dalam satu objek. Di materi ini kita akan belajar tentang pembuatan *array* dan aplikasinya di dalam suatu program.

# Struktur Dasar

## **Package**

Kita telah menginstal tools yang diperlukan dan membuat program sederhana yang menampilkan '**Hello, World!'** di bagian Pendahuluan. Kini kita akan mempelajari struktur dasar dari program Java. Setiap class di program Java diletakkan dalam sebuah package, yakni mekanisme penempatan/penamaan class agar lebih terstruktur atau modular.

Mekanisme package sangat diperlukan. Salah satunya untuk membedakan class dengan nama yang sama. Misalnya institusi Apache membuat class StringUtils. Begitu juga  dengan institusi Spring yang membuat class StringUtils. Lalu bagaimana kita membedakan keduanya jika kita ingin menggunakan kedua library tersebut secara bersamaan? Karena pada praktiknya hampir setiap aplikasi Java dibuat dengan library dari eksternal. Yup, coding Java sama dengan bermain di dunia open-source.

### **Langkah penamaan package:**

* Umumnya menggunakan nama domain institusi (dengan penulisan di balik) pembuat program tersebut.
* Sertakan nama program/aplikasi. Ikuti dengan nama modul-modulnya.
* Beri pemisah dengan tanda titik.

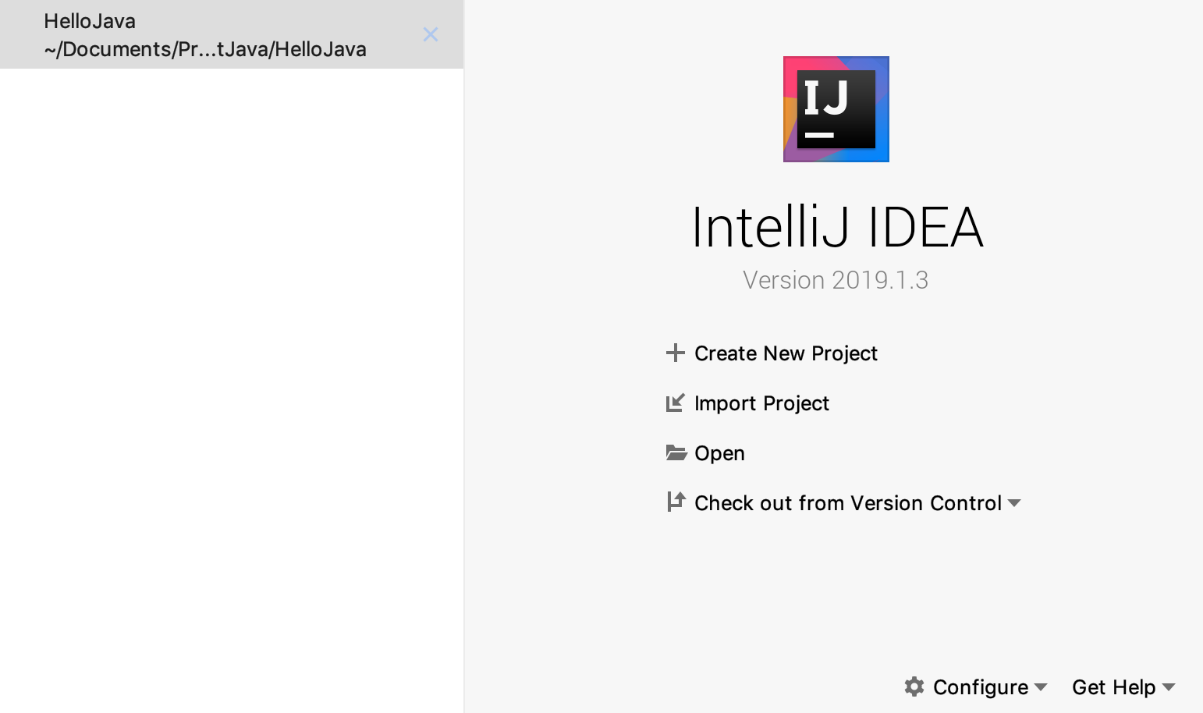
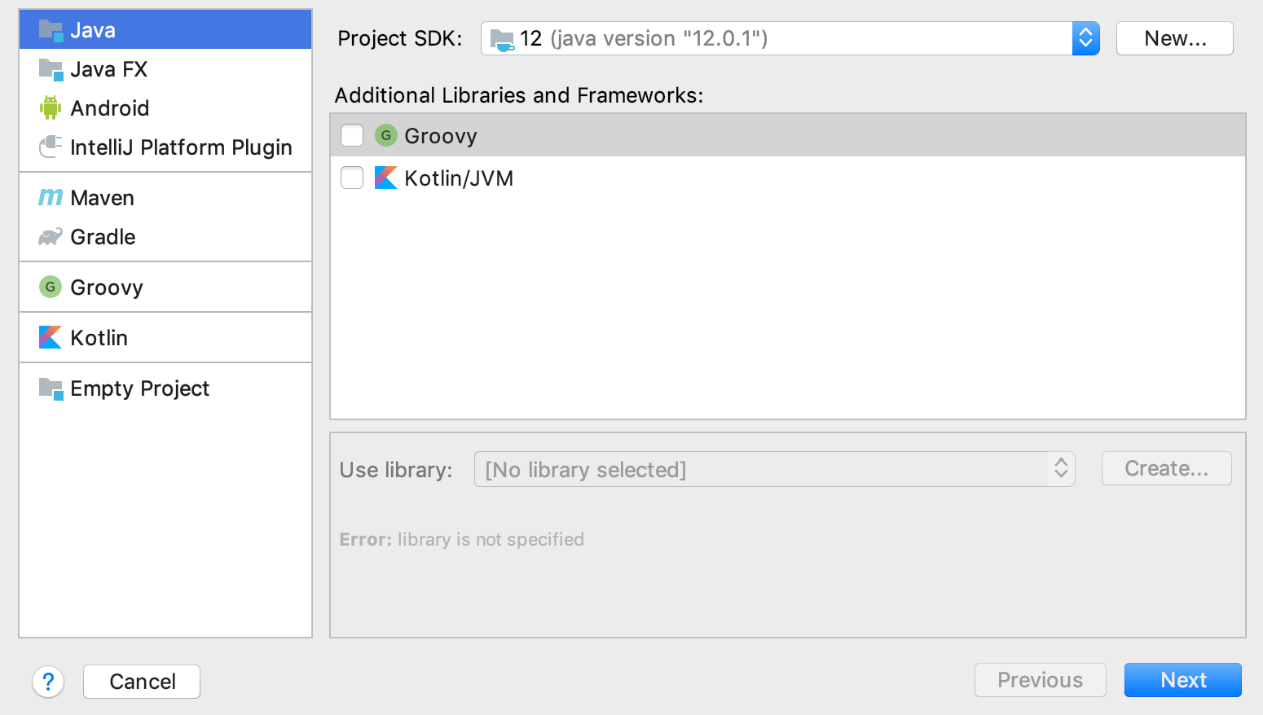
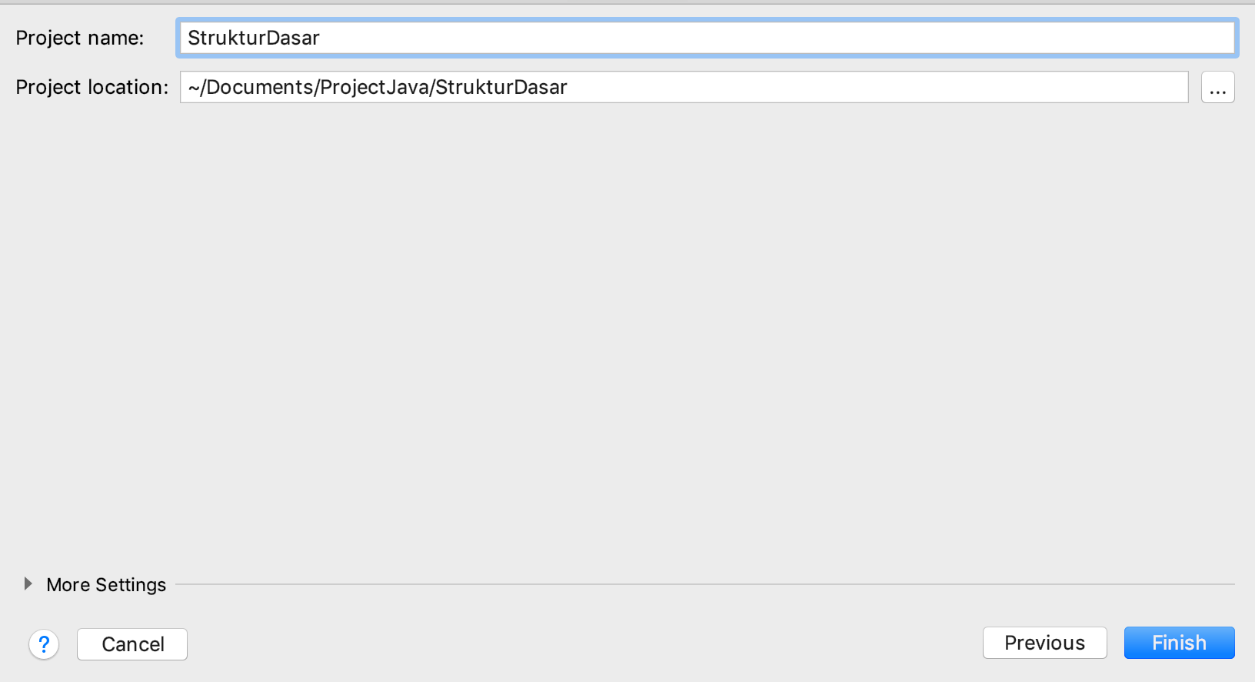
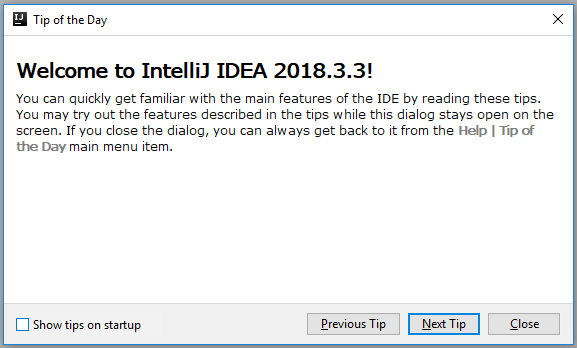
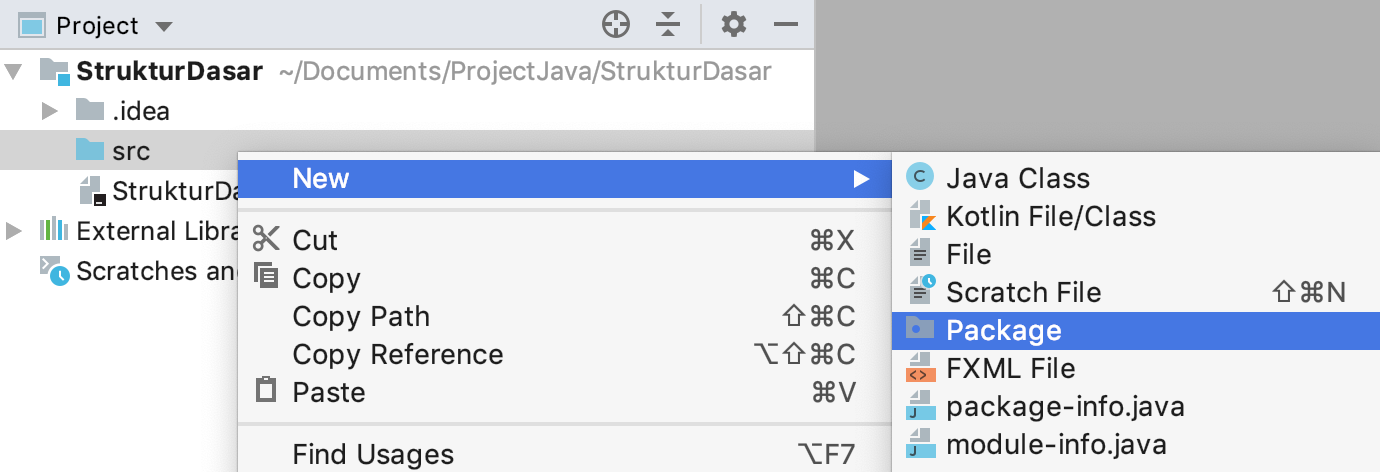
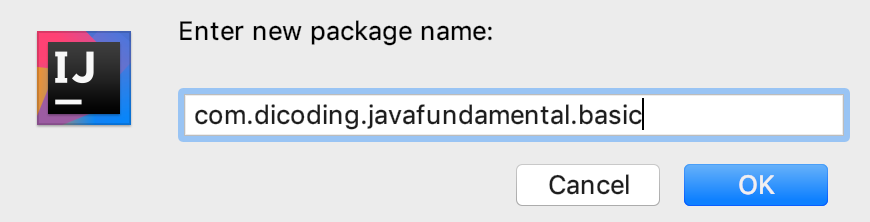
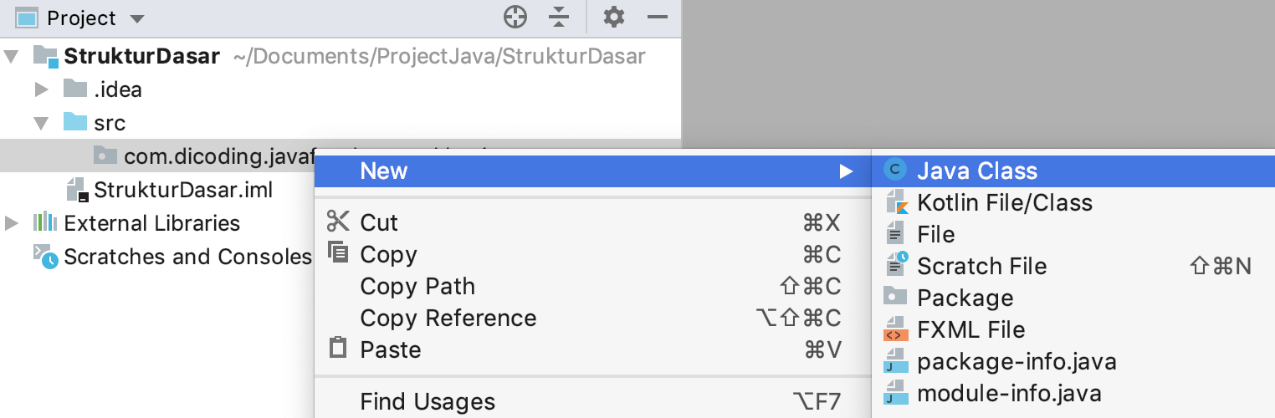
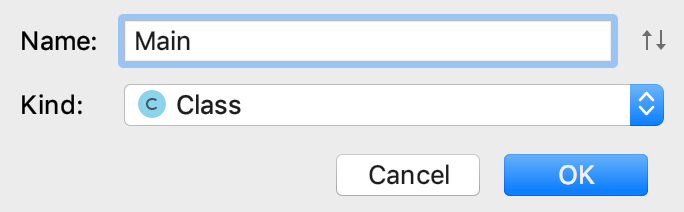
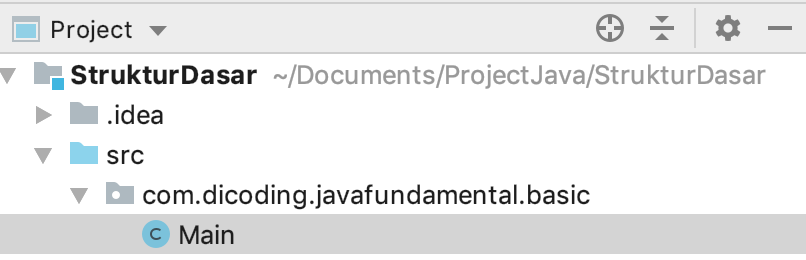
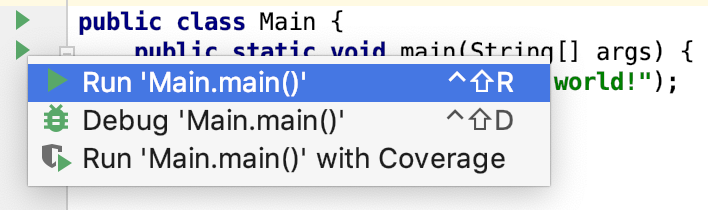
Anda tidak perlu bingung, berikut contohnya:

|  |  |
| --- | --- |
| Nama domain institusi | **dicoding.com** |
| Nama aplikasi | **Java Fundamental** |
| Nama modul | **Basic** |
| Maka nama package dapat kita tulis seperti ini | com.dicoding.javafundamental.basic |

Perhatikan tabel di atas, namapackage harus menggunakan **huruf kecil tanpa spasi.** Package akan sinkron dengan directory dari berkas (file) .java (source-code) dan juga hasil compile-nya yaitu file .class.

### **Codelab Struktur Dasar**

Di bagian pendahuluan Anda sudah membuat program HelloJava dengan sebuah class Main.Java, namun jika dilihat kembali Anda tidak akan melihat informasi package di sana. Dalam kasus ini program HelloJava mengunakan default package, di mana hal tersebut adalah cara yang sangat tidak disarankan. Oleh karena itu mari kita mulai koding dengan membuat kelas Main dalam package.

1. Buka aplikasi IntelliJ.  
   [](https://www.dicoding.com/academies/60/tutorials/1797?from=1794)
2. Setelah mengetahui beberapa penjelasan tersebut, sekarang kita pilih **Create New Project.**
3. Pastikan pilih **Java** di panel kiri lalu klik tombol **Next**.  
   [](https://www.dicoding.com/academies/60/tutorials/1797?from=1794)
4. Kemudian, masukkanlah **Project name**dengan StrukturDasar. Untuk lokasi proyeknya, Anda bisa memilih lokasi yang diinginkan. Kemudian **Klik** tombol **Finish**.  
   [](https://www.dicoding.com/academies/60/tutorials/1797?from=1794)**Catatan**: Jika ada popup-window ***Tip of the Day***, silakan lihat untuk membantu Anda beradaptasi dengan aplikasi IntelliJ.  
   [](https://www.dicoding.com/academies/60/tutorials/1797?from=1794)Tutuplah jendela ***Tip of the Day***untuk melanjutkan ke langkah berikutnya.
5. Buat package baru dengan cara sebagai berikut. Di panel proyek sebelah kiri, **klik kanan** directory **src**. Kemudian pilih **New** - **Package**.  
   [](https://www.dicoding.com/academies/60/tutorials/1797?from=1794)Kemudian masukkan **package name**-nya dengan com.dicoding.javafundamental.basic.  
   [](https://www.dicoding.com/academies/60/tutorials/1797?from=1794)
6. Selanjutnta buat class baru di dalam package yang baru saja Anda buat dengan cara, **klik kanan**di packagetersebut dan pilihlah **New** - **Java Class**.  
   [](https://www.dicoding.com/academies/60/tutorials/1797?from=1794)Kemudian masukan nama class tersebut dengan Main dan klik **OK**.  
   [](https://www.dicoding.com/academies/60/tutorials/1797?from=1794)
7. Perhatikan posisi dari class  Main yang Anda buat sebelumnya. Class tersebut akan berada di dalam directory yang sama dengan nama package-nya.  
   [](https://www.dicoding.com/academies/60/tutorials/1797?from=1794)
8. Tambahkan kode sebagai berikut di class Main.
   1. package com.dicoding.javafundamental.basic;
   3. public class Main {
   4. **public static void main(String[] args) {**
   5. **System.out.println("Hello world!");**
   6. **}**
   7. }
9. Jika Anda perhatikan terdapat kode package yang merupakan lokasi dari kelas Main. Kelas Main berada di dalam package com.dicoding.javafundamental.basic.
10. Lalu jalankanlah class Main.java dengan **klik** salah satu tombol  
    di sebelah kiri panel Editor. Lalu pilih **Run** ‘**Main.main()**’.  
    [](https://www.dicoding.com/academies/60/tutorials/1797?from=1794)
11. Hasilnya akan muncul di panel [201804171944099cc1e52abb263e031f78a155b8f4430d.](https://www.dicoding.com/academies/60/tutorials/1797?from=1794)di bawah:

|  |
| --- |
| Hello world! |

1. [](https://www.dicoding.com/academies/60/tutorials/1797?from=1794)

Apa yang sudah kita lakukan sebenarnya sama saja dengan bagian Pendahuluan. Bedanya, Anda meletakkan class Main di dalam package com.dicoding.javafundamental.basic. Ayo kita lanjut!

Ketika setiap class diletakkan di dalam sebuah package, bagaimana cara memanggil class tertentu di antara class lain yang berbeda package-nya? Caranya adalah dengan memanggil nama package secara lengkap, kemudian diikuti nama class-nya. Hal ini biasa disebut dengan istilah Fully Qualified Name. Misalnya, fully qualified name dari class  Main yang baru saja kita buat adalah:

1. com.dicoding.javafundamental.basic.Main

## Import

Import digunakan untuk menyederhanakan pemanggilan class yang berbeda package. Alhasil, Anda tak perlu menyebutkan fully qualified name dari class yang ingin digunakan. Dari pengertian ini juga tersirat kita tak perlu juga menyebutkan fully qualified name jika dalam package yang sama.

Khusus untuk class dari package java.lang, ia tidak perlu dipanggil menggunakan fully qualified name. Artinya tidak perlu menggunakan import saat memanggilnya, misalnya class java.lang.System yang kita gunakan untuk print “Hello world!”. Perhatikan bahwa kode di bawah ini:

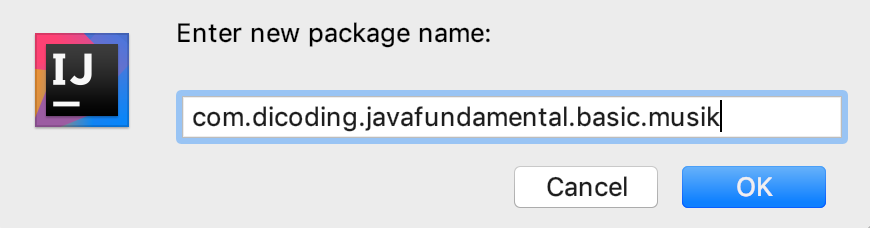
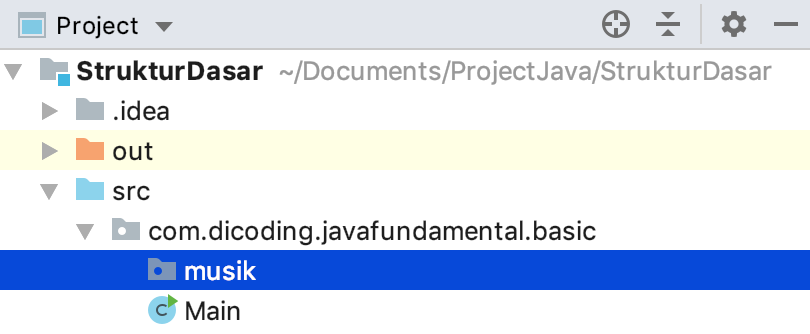
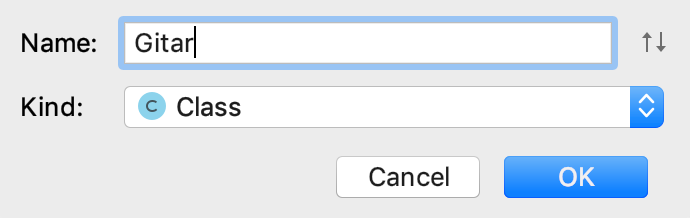
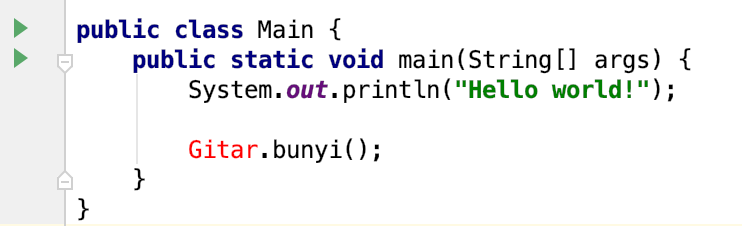
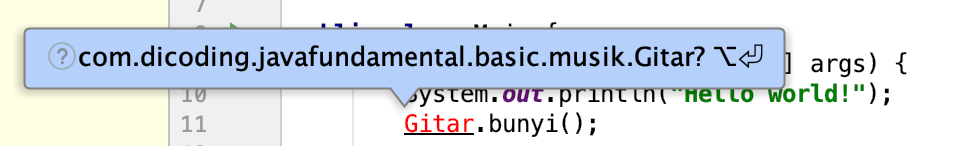
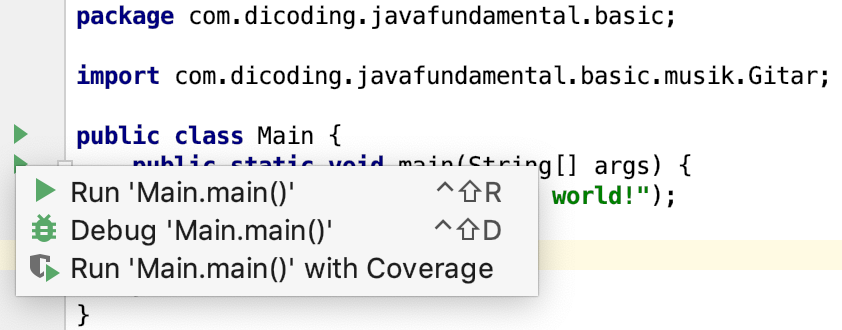
1. System.out.println("Hello world!");

Pada kode di atas, kita hanya memanggil class dengan nama class System tanpa nama package-nya.

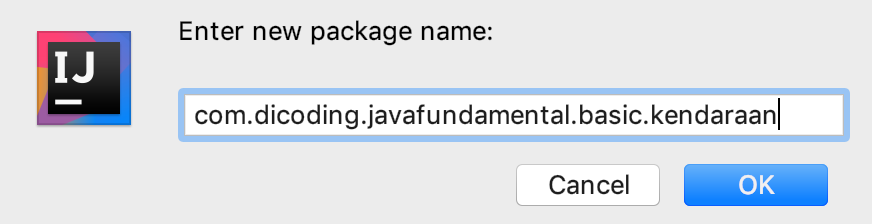
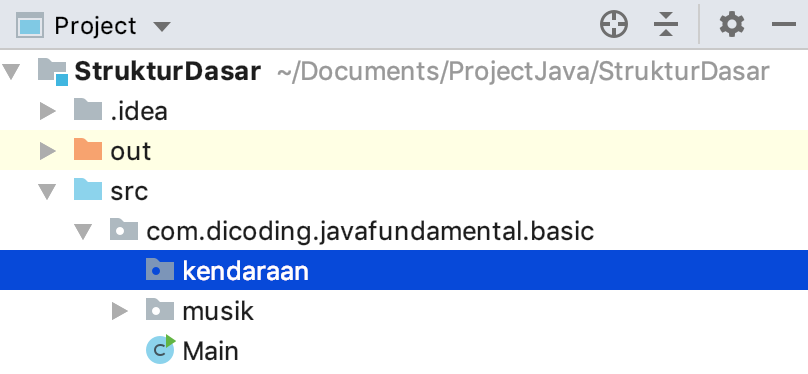
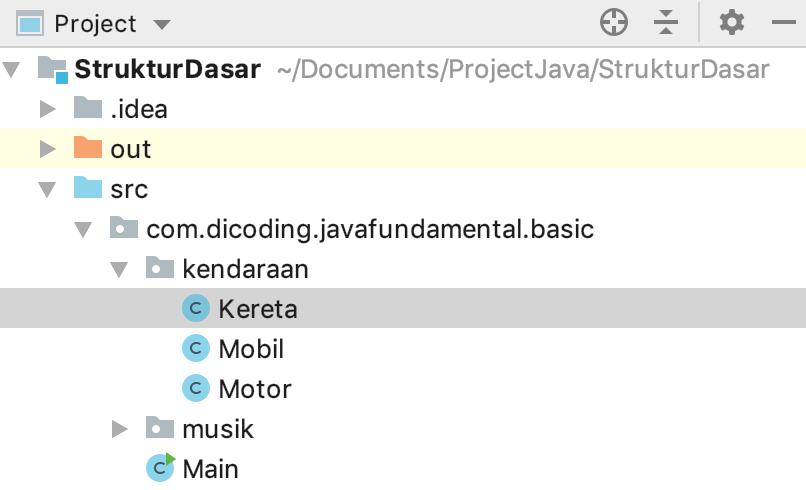
Bagaimana kalau kita import beberapa class dari package yang sama? Kita bisa menggunakan wildcard (simbol \*) untuk menggantikan nama class-nya. Artinya, Java akan otomatis mengenali seluruh class dari package tersebut.

### Codelab Import

OK untuk lebih jelasnya, mari kita langsung koding saja.

1. Buka kembali project **StrukturDasar.**
2. Buatlah kembali package dengan cara **klik kanan** directory **src,**kemudian pilih **New** - **Package.**Masukkanlah nama package-nya dengan com.dicoding.javafundamental.basic.musik.  
   [](https://www.dicoding.com/academies/60/tutorials/1797?from=1794)Maka tampilan package saat ini menjadi seperti ini:  
   [](https://www.dicoding.com/academies/60/tutorials/1797?from=1794)
3. Selanjutnya buatlah classGitar di dalam packagemusic.  
   [](https://www.dicoding.com/academies/60/tutorials/1797?from=1794)Tambahkan kode berikut di class Gitar:
   1. public class Gitar {
   2. **public static void bunyi(){**
   3. **System.out.println("jrenggg..");**
   4. **}**
   5. }
4. Buka kembali classMain, tambahkan kode seperti di bawah ini:[](https://www.dicoding.com/academies/60/tutorials/1797?from=1794)
5. Perhatikan **IntelliJ** akan memberi tanda eror berwarna merah karena classGitar tidak dikenali. Untuk itu kita perlu importclassGitar agar dikenali. Pindahkan kursor ke baris kode yang ditandai dengan error tersebut, lalu tekan **alt + Enter(Windows)**atau**option + return(MacOS).**  
   [](https://www.dicoding.com/academies/60/tutorials/1797?from=1794)Otomatis **IntelliJ**akan menambahkan baris kode seperti berikut:
   1. **import com.dicoding.javafundamental.basic.musik.Gitar;**
6. Lalu jalankanlah class Main dengan klik tombol run p  
   astikan tidak ada eror.  
   [](https://www.dicoding.com/academies/60/tutorials/1797?from=1794)Hasilnya akan jadi seperti ini:

|  |
| --- |
| Hello world! jrenggg.. |

1. [](https://www.dicoding.com/academies/60/tutorials/1797?from=1794)
2. Selanjutnya buatlah kembali package baru dengan cara **klik kanan** directory **src,**kemudian pilih **New** - **Package.**Masukkanlah nama package-nya dengan: com.dicoding.javafundamental.basic.kendaraan.  
   [](https://www.dicoding.com/academies/60/tutorials/1797?from=1794)Maka struktur package kali ini akan menjadi seperti ini:  
   [](https://www.dicoding.com/academies/60/tutorials/1797?from=1794)
3. Selanjutnya buat class baru dengan nama  Mobil, Motor, dan Kereta di dalam package tersebut.  
   [](https://www.dicoding.com/academies/60/tutorials/1797?from=1794)Tambahkan baris kode berikut di masing-masing class tersebut.  
   Pertama buka classMobildan ubahlah kode di dalamnya menjadi seperti iniberikut:
   1. public class Mobil {
   2. **public static void jumlahBan(){**
   3. **System.out.println("Ban mobil 4");**
   4. **}**
   5. }

Selanjutnya bukalah classMotordan ubahlah kode di dalamnya:

* 1. public class Motor {
  2. **public static void jumlahBan() {**
  3. **System.out.println("Ban motor 2");**
  4. **}**
  5. }

Terakhir bukalah classKeretadan ubah juga kode di dalamnya menjadi seperti ini:

* 1. public class Kereta {
  2. **public static void jumlahBan() {**
  3. **System.out.println("Ban kereta banyak!");**
  4. **}**
  5. }

1. Bukalah classMain lalu ketik (jangan copy-paste) kode berikut.
   1. **Mobil.jumlahBan();**
2. Ketika kode di atas mulai diketik sebagian, maka IntelliJ akan memberi saran kode (dikenal dengan istilah auto-complete) seperti gambar di bawah. Tekan Enter agar IntelliJ melengkapi baris kode.  
   [](https://www.dicoding.com/academies/60/tutorials/1797?from=1794)Dengan bantuan auto complete, Anda tidak perlu melakukan import secara manual.
3. Lengkapi classMaindengan kode sebagai berikut :
   1. Motor.jumlahBan();
   2. Kereta.jumlahBan();
4. Baris import dapat Anda sederhanakan yang awalnya seperti ini:
   1. import com.dicoding.javafundamental.basic.kendaraan.Kereta;
   2. import com.dicoding.javafundamental.basic.kendaraan.Mobil;
   3. import com.dicoding.javafundamental.basic.kendaraan.Motor;
   4. import com.dicoding.javafundamental.basic.musik.Gitar;

Menjadi:

* 1. **import com.dicoding.javafundamental.basic.kendaraan.\*;**
  2. import com.dicoding.javafundamental.basic.musik.Gitar;

Maka kelas Mainmenjadi seperti ini:

* 1. package com.dicoding.javafundamental.basic;
  3. **import com.dicoding.javafundamental.basic.kendaraan.\*;**
  4. **import com.dicoding.javafundamental.basic.musik.Gitar;**
  6. public class Main {
  7. public static void main(String[] args) {
  8. System.out.println("Hello world!");
  10. Gitar.bunyi();
  12. **Mobil.jumlahBan();**
  13. **Motor.jumlahBan();**
  14. **Kereta.jumlahBan();**
  15. }
  16. }

**Catatan:**  
Penggunaan wildcard (\*) kadang tidak disarankan jika class di dalam package tersebut terlalu banyak. Sebabnya, ia memperlama waktu untuk pencarian class tersebut. Untuk itu kita serahkan ke IntelliJ mengatur import yang optimal. Pilih menu **Code** | **Optimize Imports**. Baris import akan kembali ke sebelumnya tanpa menggunakan wildcard (\*).

1. Lalu jalankanclass Main dengan **klik** tombol [20180417194425b07c0f2c5fc367290a3884f55f676307.](https://www.dicoding.com/academies/60/tutorials/1797?from=1794)  
   dan pastikan tidak ada error di dalamnya.  
   [](https://www.dicoding.com/academies/60/tutorials/1797?from=1794)
2. Maka hasilnya jadi seperti ini:

|  |
| --- |
| Hello wordl! jrenggg.. Ban mobil 4 Ban motor 2 Ban kereta banyak! |

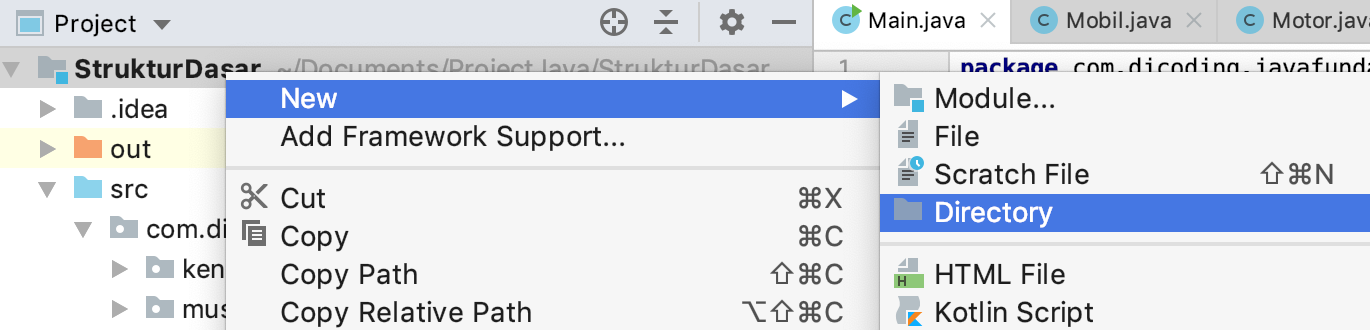
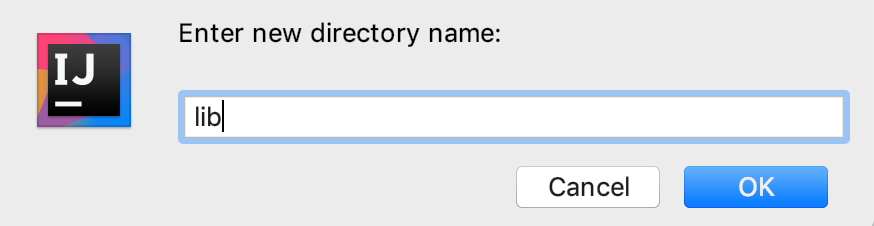
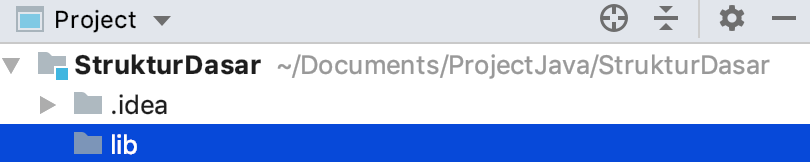
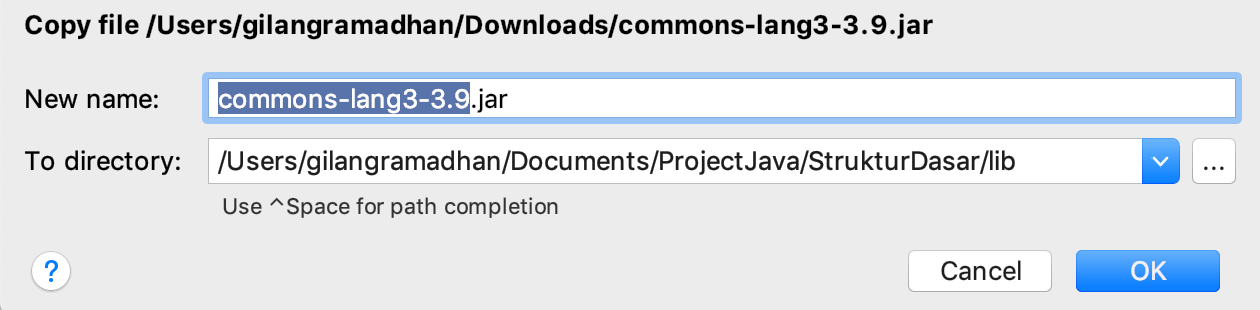
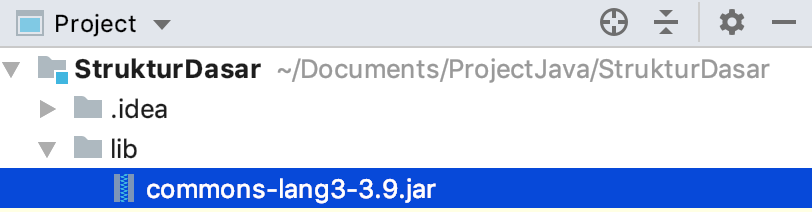
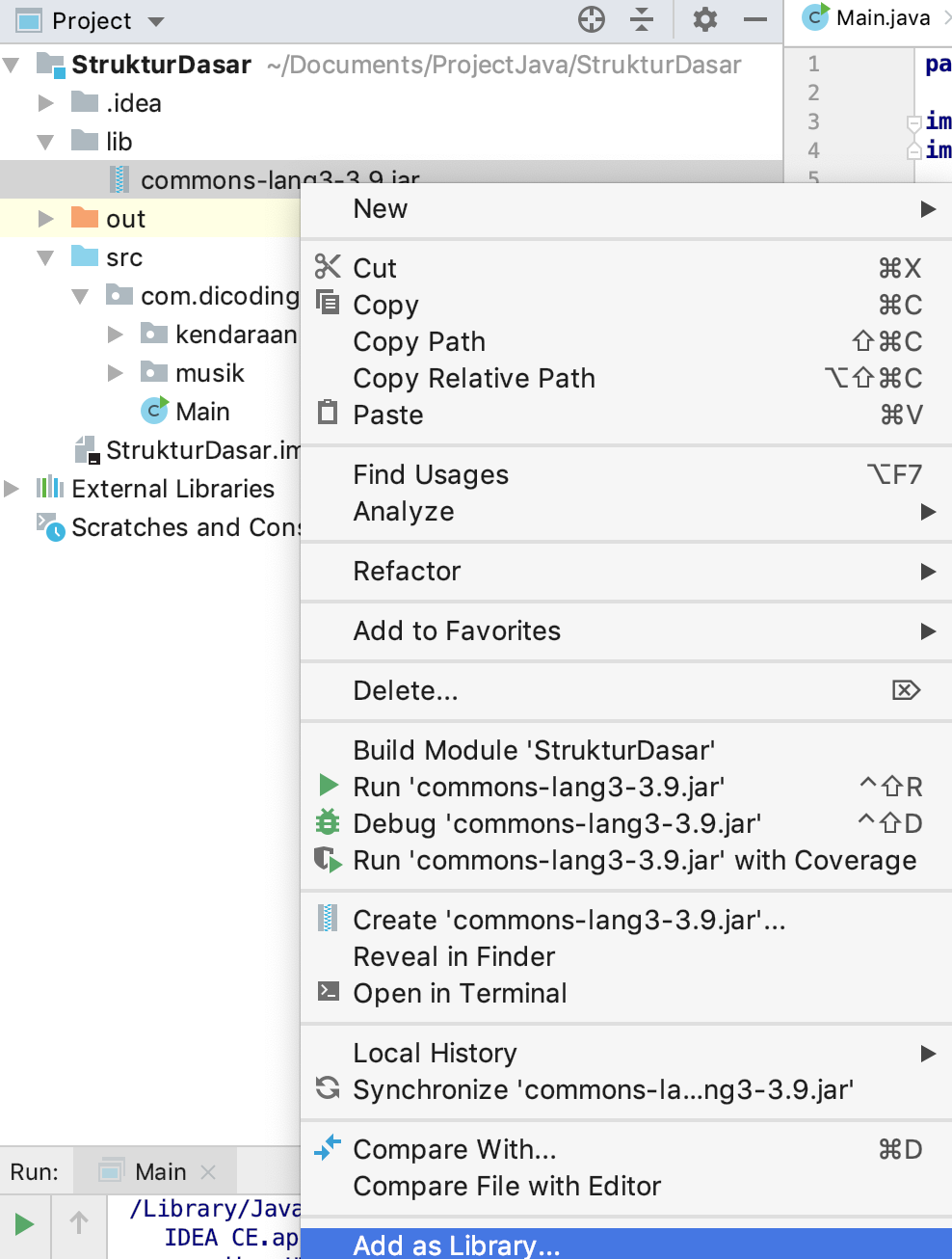
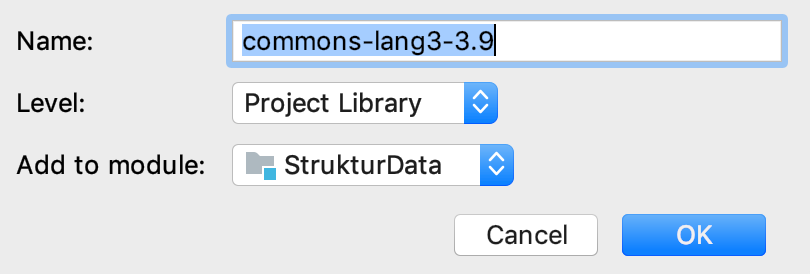
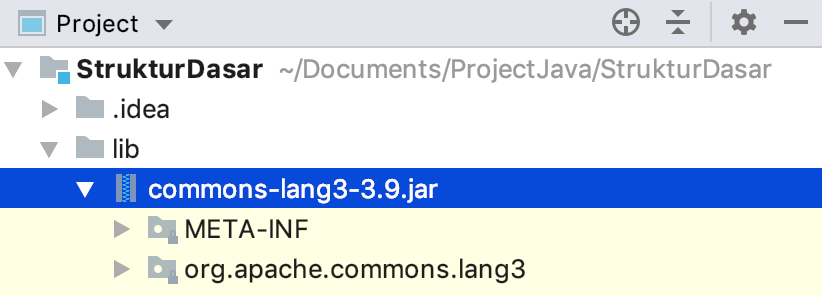
1. [](https://www.dicoding.com/academies/60/tutorials/1797?from=1794)

## Classpath

Classpath adalah mekanisme di Java untuk menemukan class lain. Biasanya class lain tersebut berasal dari library yang berbeda atau bahkan JDK itu sendiri (kita sudah memakai class System). Jika Java tidak bisa menemukan class yang kita panggil melalui kode kita, akan terjadi error ClassNotFoundException atau NoClassDefFoundError. Error ini biasa terjadi bagi pemula Java karena biasanya belum paham konsep classpath atau ada kesalahan pengaturan classpath.

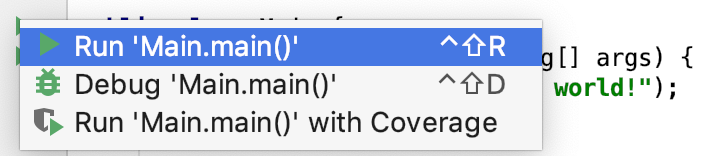
Sekarang tenang ya kalau ketemu error serupa. Kita kan sudah tahu bahwa terduga adalah problem di classpath. Jadi paham juga deh, dari mana kita mesti melakukan troubleshooting.

### Codelab Classpath

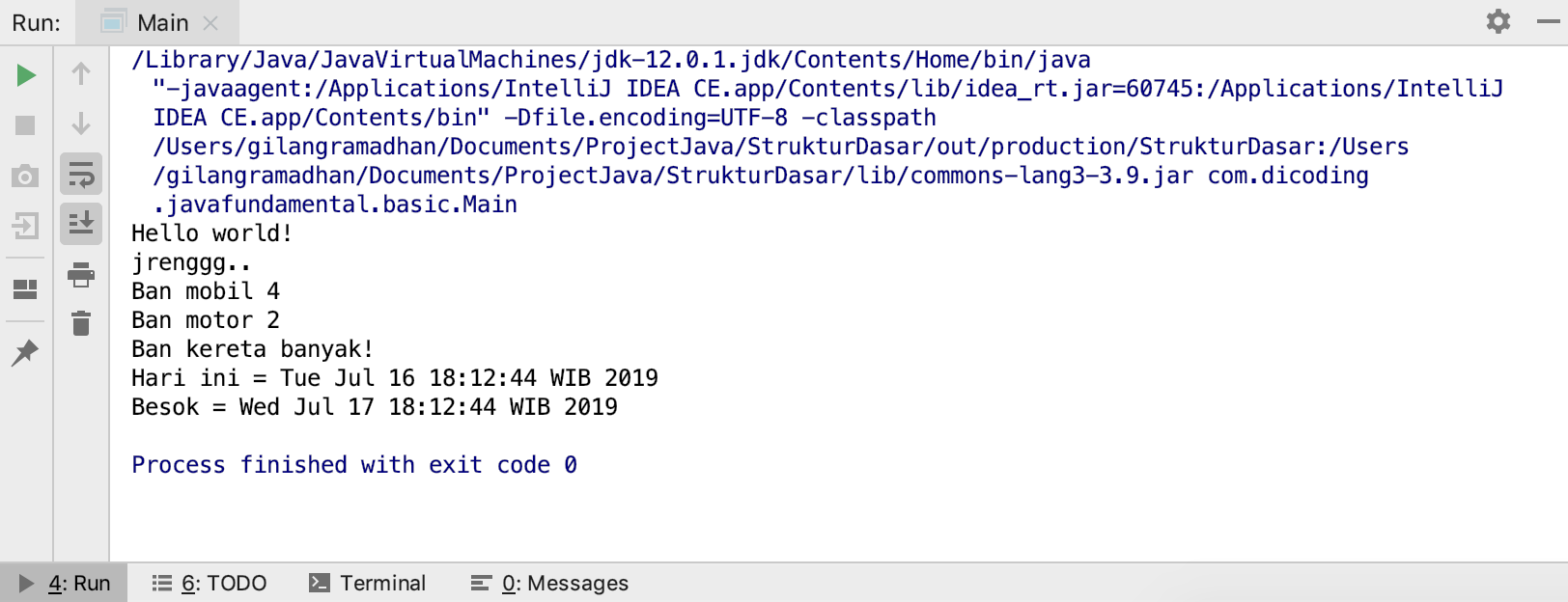
1. Buka kembali proyek StrukturDasar. Buat directory baru dengan cara **klik kanan** di proyek StrukturDasar seperti gambar di bawah:  
   [](https://www.dicoding.com/academies/60/tutorials/1797?from=1794)Selanjutnya masukkan **directory name**dengan lib.  
   [](https://www.dicoding.com/academies/60/tutorials/1797?from=1794)Maka akan jadi seperti ini:  
   [](https://www.dicoding.com/academies/60/tutorials/1797?from=1794)
2. Unduh library apache**common-lang**dari tautan berikut:
   * <https://repo1.maven.org/maven2/org/apache/commons/commons-lang3/3.9/commons-lang3-3.9.jar>
3. Salin berkas .jar tersebut ke directory **lib**.  
   [](https://www.dicoding.com/academies/60/tutorials/1797?from=1794)Maka akan jadi seperti ini:[](https://www.dicoding.com/academies/60/tutorials/1797?from=1794)
4. Klik kanan di commons-lang3-3.9.jar untuk menjadikan file jar menjadi **library,** pilihlah **Add as Library...**.  
   [](https://www.dicoding.com/academies/60/tutorials/1797?from=1794)Setelah itu biarkan namanya menjadi default dan tekan **OK**.  
   [](https://www.dicoding.com/academies/60/tutorials/1797?from=1794)Maka hasilnya akan jadi seperti ini:  
   [](https://www.dicoding.com/academies/60/tutorials/1797?from=1794)Perhatikan sekarang ada tanda panah di kiri sehingga kita bisa browse isi berkas jar tersebut. Jika tanda tersebut tidak ada, maka proses penambahan library yang Anda lakukan gagal.
5. Sekarang berkas jar sudah menjadi library di projectStrukturDasar.
6. Tambahkan baris kode berikut di classMain. Sebaiknya ketik (koding) saja supaya get a grip tentang fitur auto -completeIntelliJ ini.
   * Date today = new Date();
   * System.out.println("Hari ini = " + today);
   * Date tomorrow = DateUtils.addDays(today, 1);
   * System.out.println("Besok = " + tomorrow);

Sehingga kelas Main.java menjadi seperti ini:

* + package com.dicoding.javafundamental.basic;
  + import com.dicoding.javafundamental.basic.kendaraan.\*;
  + import com.dicoding.javafundamental.basic.musik.Gitar;
  + **import org.apache.commons.lang3.time.DateUtils;**
  + import java.util.Date;
  + public class Main {
  + public static void main(String[] args) {
  + System.out.println("Hello world!");
  + Gitar.bunyi();
  + Mobil.jumlahBan();
  + Motor.jumlahBan();
  + Kereta.jumlahBan();
  + **Date today = new Date();**
  + **System.out.println("Hari ini = " + today);**
  + **Date tomorrow = DateUtils.addDays(today, 1);**
  + **System.out.println("Besok = " + tomorrow);**
  + }
  + }

1. Jalankan class Main dengan **klik tombol**[20180417194425b07c0f2c5fc367290a3884f55f676307.](https://www.dicoding.com/academies/60/tutorials/1797?from=1794), pastikan tidak ada error.  
   [](https://www.dicoding.com/academies/60/tutorials/1797?from=1794)Maka hasilnya akan menjadi seperti ini:

|  |
| --- |
| Hello wordl! jrenggg.. Ban mobil 4 Ban motor 2 Ban kereta banyak! Hari ini = Tue Jul 16 18:12:44 WIB 2019 Besok = Wed Jul 17 18:12:44 WIB 2019 |

1. [](https://www.dicoding.com/academies/60/tutorials/1797?from=1794)

Praktik yang kita lakukan barusan adalah menambahkan library (file jar) secara manual ke proyek. Cara ini sebenarnya kurang efektif jika dilakukan untuk proyek besar. Bayangkan jika kita menggunakan library A yang bergantung (dependencies) ke library B, C, D, lalu library B bergantung ke library X dan Y. Cukup bikin repot kan? Solusinya, gunakan tools seperti **maven** atau **gradle**. Pembahasan maven dan gradle di luar cakupan dari class ini. Saran kami, bacalah referensi yang ada di bawah karena kedua tools tersebut sudah menjadi standar dalam development Java.

* <http://www.baeldung.com/java-classnotfoundexception-and-noclassdeffounderror>
* [https://maven.apache.org](https://maven.apache.org/)
* [https://gradle.org](https://gradle.org/)
* <https://spring.io/guides/gs/maven>
* <https://spring.io/guides/gs/gradle>

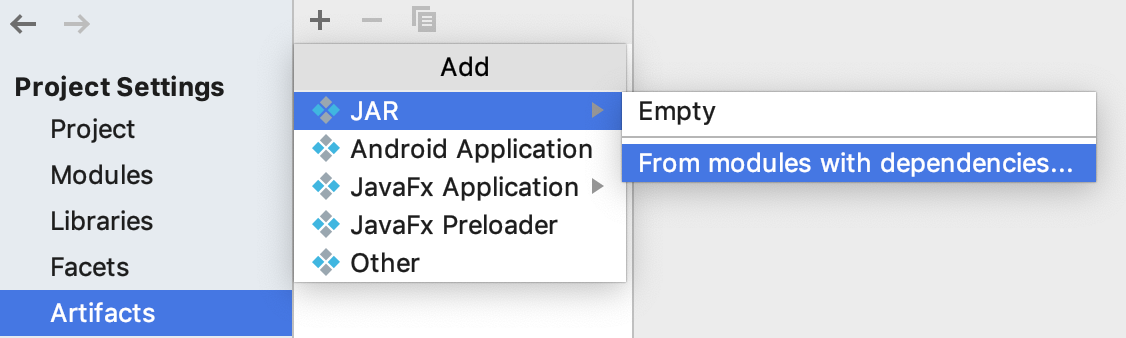
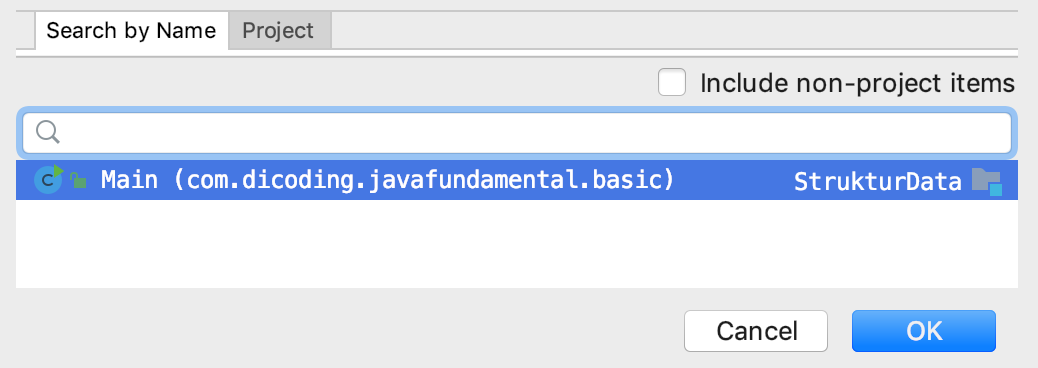
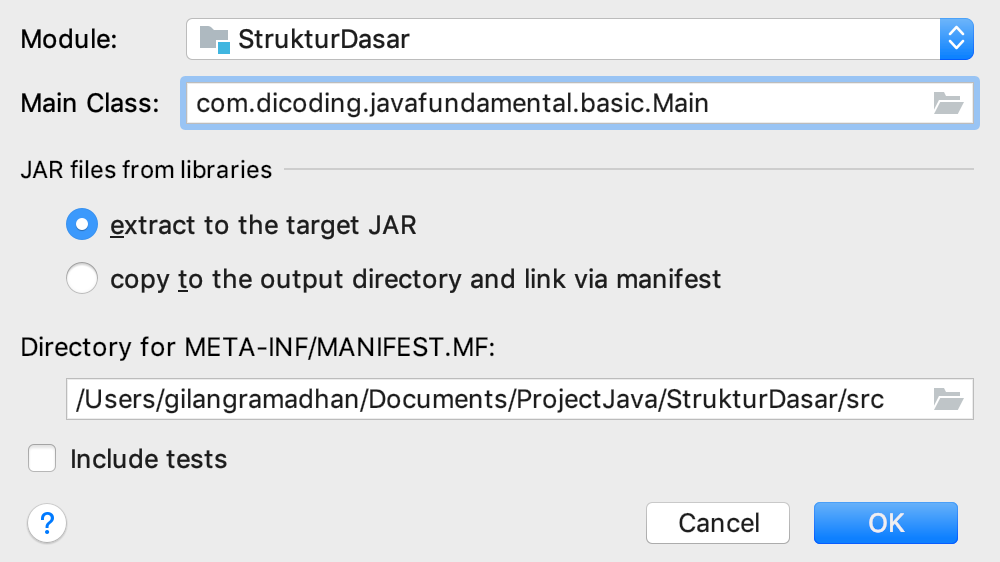
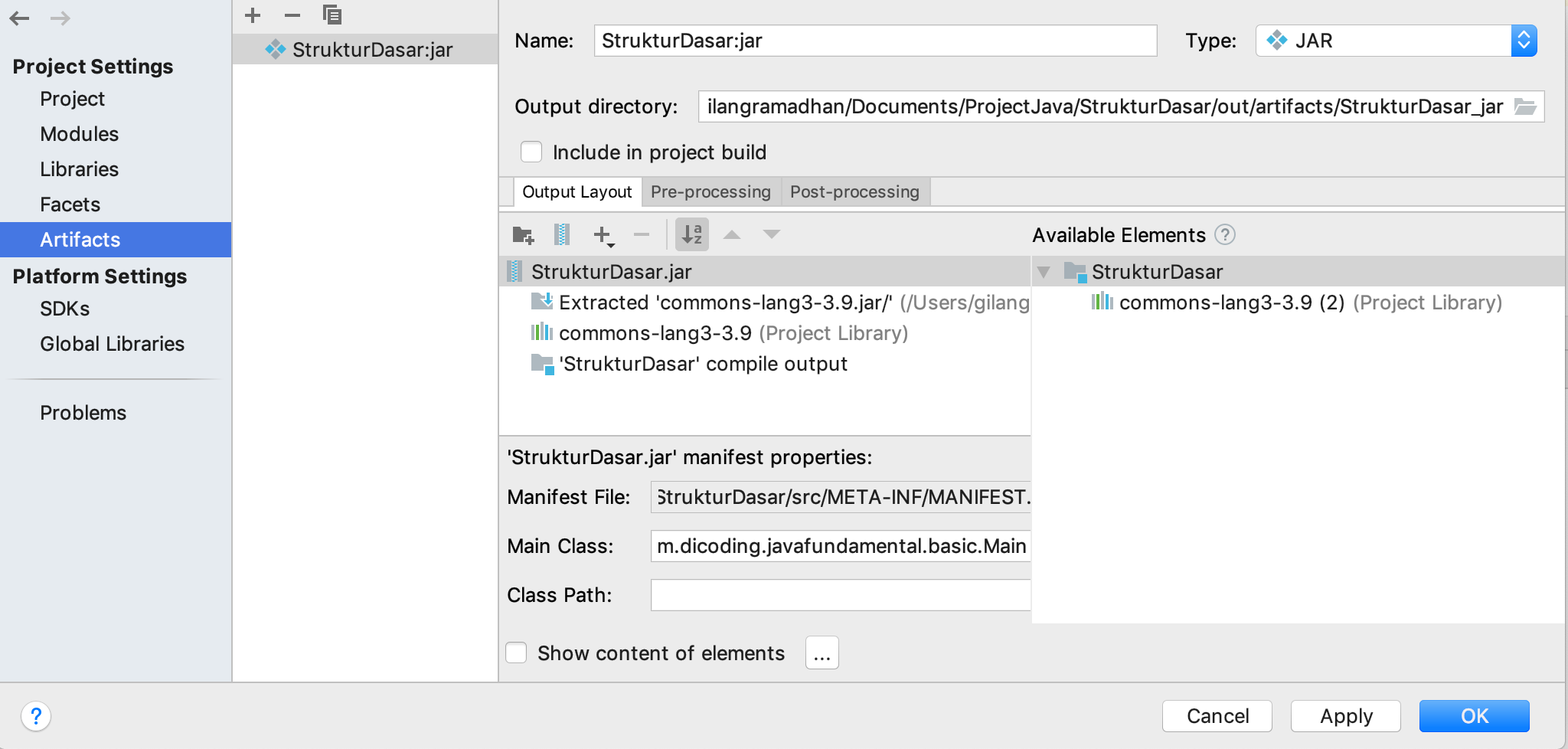
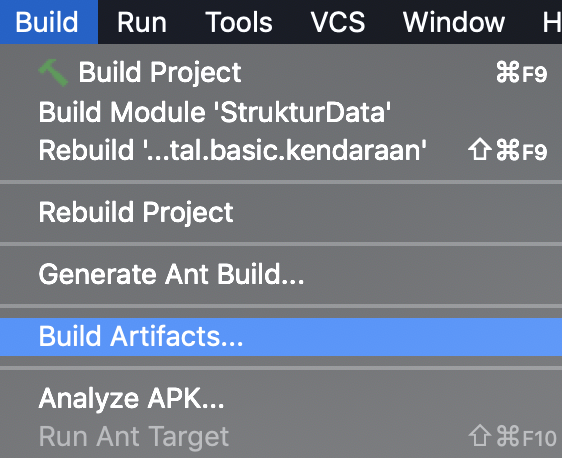
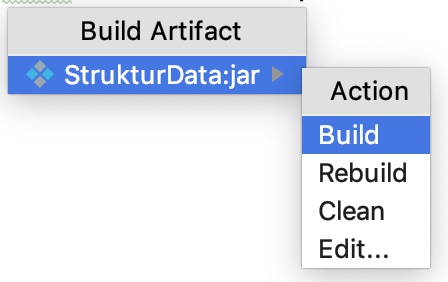
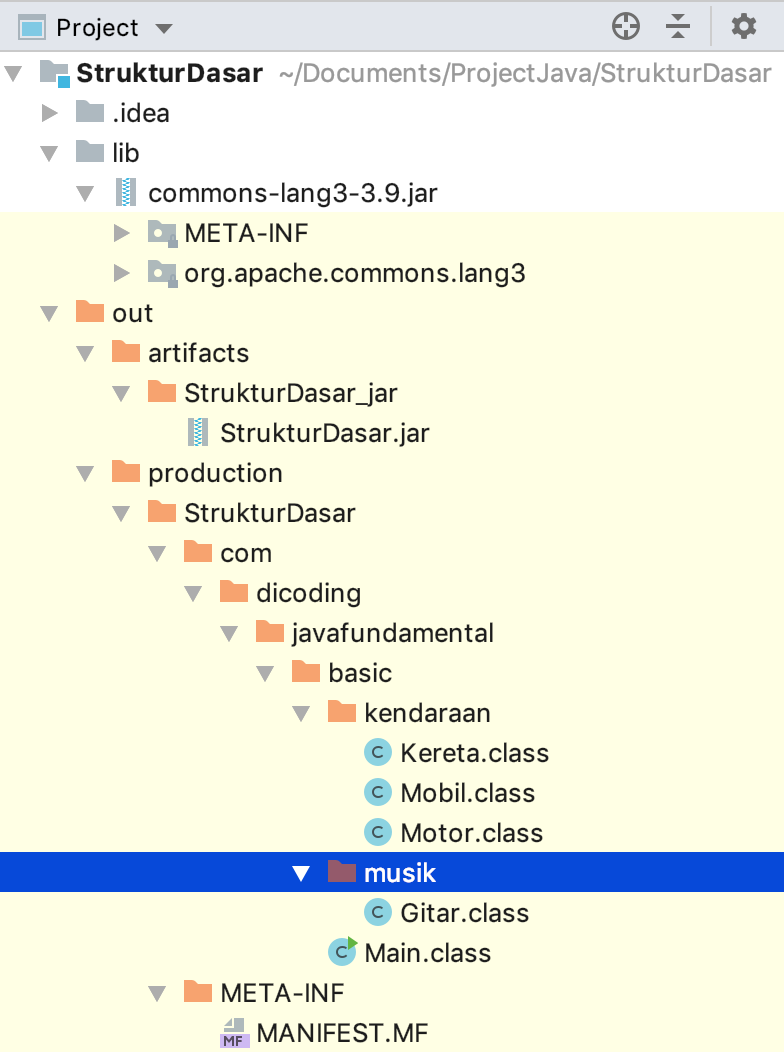
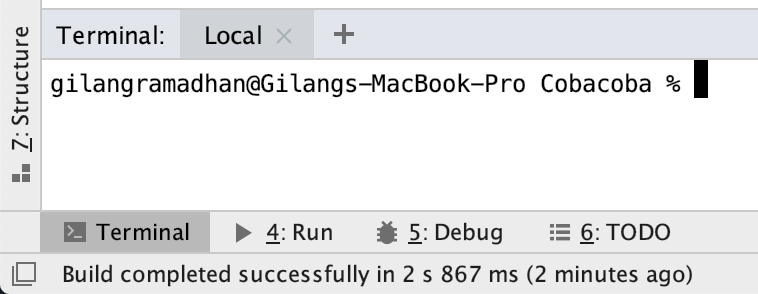
## Pemaketan

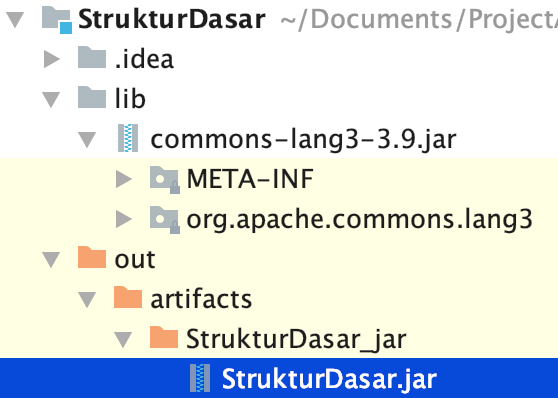
Setiap kode Java yang ditulis akan dikompilasi menjadi bytecode (file dengan extension .class) lalu pada akhirnya akan dipaketkan untuk didistribusikan. Bentuk pemaketan yang sering digunakan adalah bentuk **JAR** (Java ARchive), ada juga bentuk lain misal **WAR** (Web ARchive) dan **EAR** (Enterprise ARchive). Dari nama pemaketan ini bisa ditebak sebenarnya hanya berkas archive atau berkas zip. Isinya bisa diintip menggunakan winzip, winrar atau aplikasi sejenis.

 Di dalam hasil pemaketan tersebut ada berkas metadata yang menjelaskan isi berkas JAR. File tersebut adalah manifest.mf yang diletakkan di directory META-INF.

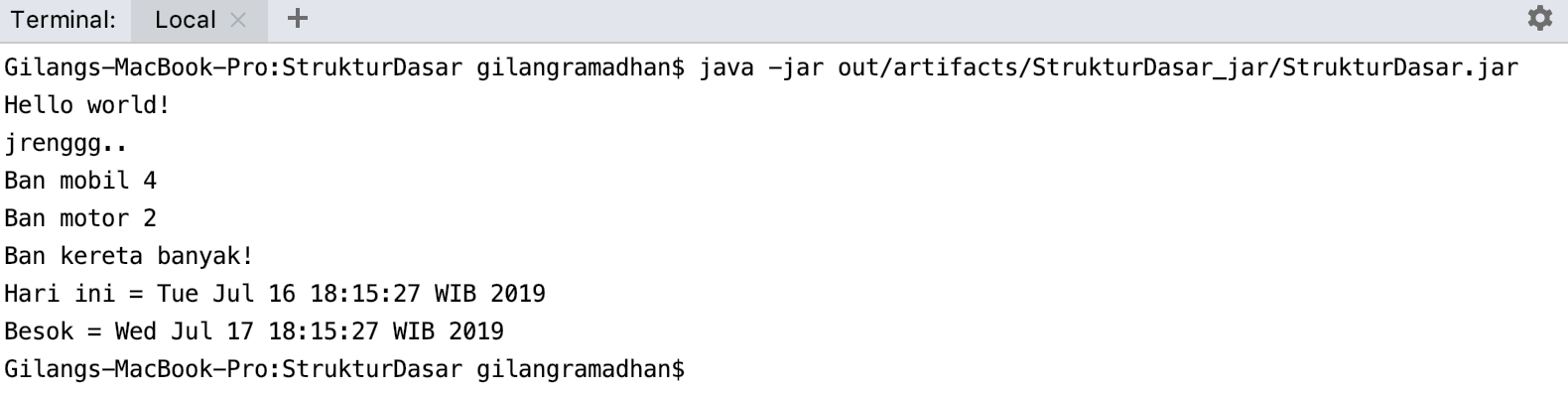
### Codelab Pemaketan

Mari kita coba paketkan project **StrukturDasar** dalam bentuk JAR.

1. Buka kembali IntelliJ. **File** | **Project Structure**|**Artifacts**. Klik tombol **+** lalu pilih **JAR** | **From modules with dependencies**.  
   [](https://www.dicoding.com/academies/60/tutorials/1797?from=1794)Pilih **Main Class**, klik tombol [20180417194311e6ec29961e0e73c32c1ac560705cc1ed.](https://www.dicoding.com/academies/60/tutorials/1797?from=1794)  
   atau[201907161511350bf93fce0195078422335d24db38b479](https://www.dicoding.com/academies/60/tutorials/1797?from=1794) lalu ketik Main di popup. Kemudia klik **OK** untuk melanjutkan.  
   [](https://www.dicoding.com/academies/60/tutorials/1797?from=1794)Maka akan jadi seperti gambar di bawah. Klik **OK** untuk melanjutkan.  
   [](https://www.dicoding.com/academies/60/tutorials/1797?from=1794)Hasil akhirnya akan menjadi seperti gambar di bawah. Klik **OK** untuk melanjutkan.  
   [](https://www.dicoding.com/academies/60/tutorials/1797?from=1794)
2. Selanjutnya pilihlah menu **Build**dan pilih **Build Artifacts**.  
   [](https://www.dicoding.com/academies/60/tutorials/1797?from=1794)Pilihlah **Build** untuk memulai membangun **Artifacts**.  
   [](https://www.dicoding.com/academies/60/tutorials/1797?from=1794)Perhatikan, akan terbentuk StrukturDasar.jar di directory artifacts yang isinya sama persis seperti directory production.  
   [](https://www.dicoding.com/academies/60/tutorials/1797?from=1794)
3. Perhatikan dan buka berkasMANIFEST.MF yang dibuat otomatis oleh proses **Build**.
   1. Manifest-Version: 1.0
   2. Main-Class: com.dicoding.javafundamental.basic.Main
4. Buka panel **Terminal** di bagian bawah IDE (sesuaikan dengan versi IDE Anda):  
   [](https://www.dicoding.com/academies/60/tutorials/1797?from=1794)Kemudian ketikkan**command**atau**perintah** berikut dan lihat hasilnya.
   1. java -jar **[lokasi .jar]**

Untuk lokasi .jar, Anda dapat melihat letak berkas .jar yang sudah Anda generate. Contohnya seperti ini:  
[](https://www.dicoding.com/academies/60/tutorials/1797?from=1794)Maka dapat disimpulkan lokasinya adalah **out/artifacts/StrukturDasar\_jar/StrukturDasar.jar**.  
  
Maka hasilnya akan jadi seperti ini:

|  |
| --- |
| Hello wordl! jrenggg.. Ban mobil 4 Ban motor 2 Ban kereta banyak! Hari ini = Tue Jul 16 18:12:44 WIB 2019 Besok = Wed Jul 17 18:12:44 WIB 2019 |

[](https://www.dicoding.com/academies/60/tutorials/1797?from=1794)

Selesai sudah materi struktur dasar. Sampai ketemu di materi selanjutnya, happy coding!

Jika Anda mengalami kesulitan, silakan unduh source code-nya di [Source Code Struktur Dasar](https://github.com/dicodingacademy/BelajarJava/tree/master/Modul1Basic/StrukturDasar).