

Bab 1. Bahasa Pemrograman Java

OBJEKTIF :

1. Mahasiswa mampu belajar Bahasa Pemrograman Java seperti mengetahui sistem komputer, Bahasa Pemrograman Java, langkah Instalasi dan pengaturan JDK, mengetahui *Integrated Development Environment*, menulis Program Java, menganalisa Program Java, *comment*, *method print* dan *method println* serta *error* pada Bahasa Pemrograman Java.
 2. Mahasiswa mampu menginstal JDK 12 dan Apache Netbeans 12.0.
 3. Mahasiswa mampu menggunakan *Software IDE* Netbeans untuk menulis Program Java.
-

1.1 Sistem Komputer: *Hardware* dan *Software*

Proses pemrograman harus didukung dengan pemahaman mengenai komponen-komponen yang membentuk sebuah komputer. Komponen-komponen ini dikategorikan menjadi dua: *Hardware* dan *Software*.

Hardware

Hardware adalah istilah untuk komponen-komponen fisik yang membentuk komputer. Sebuah komputer adalah sebuah sistem dari komponen-komponen yang masing-masing mempunyai fungsi-fungsi tersendiri. Sistem komputer terdiri dari komponen-komponen utama berikut:

- CPU (*Central Processing Unit*)
- RAM (*Random Access Memory*)
- Hard Disk
- *Device Input*
- *Device Output*

CPU

CPU (*Central Processing Unit*) adalah komponen komputer yang sering disebut sebagai jantung dari komputer. CPU mengatur semua kerja komputer. Tugas dari CPU adalah mengambil instruksi, memroses instruksi, dan memberikan hasil berupa data. Semua proses komputasi komputer dilakukan oleh CPU.

RAM

RAM adalah singkatan dari *Random Access Memory* atau sering disebut hanya dengan memori adalah komponen komputer yang bertugas sebagai tempat penyimpanan. RAM menyimpan instruksi-instruksi komputer sebelum diproses oleh CPU dan juga data yang diproses oleh instruksi-instruksi komputer. RAM bersifat *volatile* yang berarti sementara. Ketika komputer dimatikan, isi dari RAM akan langsung terhapus.

Kita dapat membayangkan RAM sebagai almari yang terdiri dari laci-laci bernomor yang setiap lacinya dapat menyimpan data sebesar 1 byte. Nomor-nomor laci ini disebut sebagai alamat memori. Gambar berikut mengilustrasikan bagaimana RAM menyimpan data.

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29

Pada gambar di atas diilustrasikan angka 149 disimpan pada alamat memori 16 dan angka 72 disimpan dalam alamat memori 23.

Catatan. Pada komputer data disimpan dalam bentuk bilangan biner. Bilangan biner adalah angka yang hanya terdiri dari 0 dan 1. Sedangkan, byte adalah 8 digit angka biner.

Hard Disk

Hard disk adalah tempat penyimpanan data untuk jangka panjang. Meskipun komputer dimatikan data dalam hard disk tidak hilang.

Device Input

Input adalah data yang komputer terima dari pengguna. *Device* yang menerima data dan mengirimkannya ke komputer disebut dengan *device input*. Device input yang umum berupa keyboard, mouse, scanner, dan camera.

Device Output

Output adalah data yang komputer kirimkan untuk pengguna. *Device output* adalah *device* yang menerima *output* dan menyajikannya ke pengguna. *Device output* yang umum adalah monitor dan printer.

Software

Software adalah program-program yang berjalan pada komputer. Terdapat dua kategori *software* yaitu sistem operasi dan *software* aplikasi. Sistem operasi adalah kumpulan program-program yang mengatur *device-device hardware* dan mengatur proses-prosesnya. Windows, Linux, dan Mac OS adalah contoh-contoh sistem operasi.

Software aplikasi adalah program-program yang membuat komputer berguna bagi pengguna. Program-program ini melakukan tugas-tugas umum dan menyelesaikan persoalan-persoalan spesifik yang memenuhi kebutuhan pengguna. Word processing, spreadsheet, dan *database* adalah contoh-contoh *software* aplikasi.

1.2 Bahasa Pemrograman Java

Bahasa Pemrograman

Program adalah rangkaian instruksi-instruksi yang dieksekusi oleh komputer untuk mengerjakan tugas tertentu. Tentunya, kita perlu menuliskan instruksi-instruksi ini dalam sebuah bahasa yang dimengerti oleh komputer. Bahasa yang dimengerti komputer adalah bahasa mesin. Bahasa mesin adalah bahasa yang kosakatanya dalam bilangan biner (bilangan yang hanya terdiri dari angka 0 dan 1). Gambar berikut adalah ilustrasi dari bahasa mesin.

```
1101010101010010100010
100101111000001000100101
0101010101001011100011
010001010010101110001001
010101010000111110101010
0101010101010010001111
```

Proses penulisan program dalam bahasa mesin sangatlah sulit. Dahulu di generasi awal komputer, *programmer* melakukannya. Oleh karena sulitnya menulis sebuah program dalam bahasa mesin, ilmuwan komputer menciptakan bahasa-bahasa pemrograman yang mudah dipahami dan mudah ditulis oleh manusia. Bahasa ini disebut dengan bahasa pemrograman *high-level*. Serta membuat program (dalam bahasa mesin) yang dapat menerjemahkannya ke bahasa mesin. Java adalah salah satu contoh dari bahasa pemrograman *high-level*. Bahasa pemrograman *high-level* lainnya antara lain adalah C, C++, Python, JavaSciprt, Ruby, Perl, Scheme, atau BASIC.

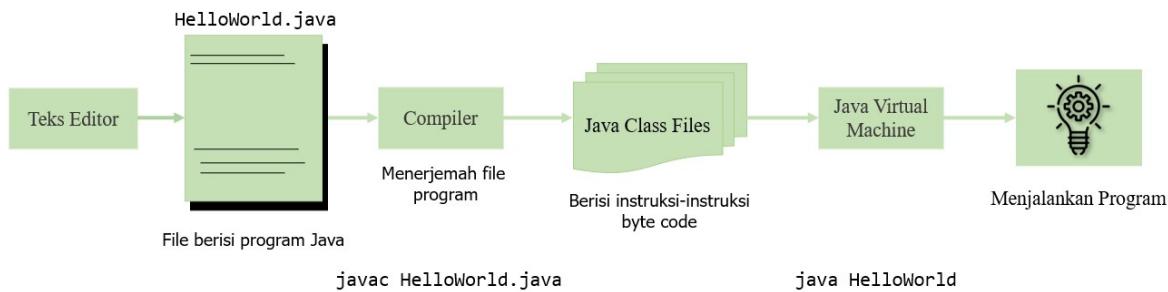
Setiap bahasa pemrograman *high-level* mempunyai kosakata dan aturan penulisan tersendiri untuk menuliskan instruksi-instruksi. Dalam bahasa pemrograman *high-level*, instruksi disebut dengan **statement** dan aturan penulisan *statement* disebut sebagai **syntax**. Aturan *syntax* ini membuat seolah bahasa pemrograman seperti sebuah kode untuk menuliskan instruksi-instruksi ke komputer. Oleh karena ini, *programmer* sering mengatakan program yang mereka buat sebagai kode komputer dan proses menulis program sebagai koding (dari bahasa Inggris *coding*).

Compiler dan Java Virtual Machine

Kita membuat program Java dengan mengetikkannya pada teks editor dan menyimpannya dalam sebuah file. File yang berisi program Java yang kita tulis disebut sebagai **source code**. File *source code* Java haruslah berekstensi .java.

Setelah kita menyimpan *source code* ke sebuah file, kita harus menjalankan compiler Java. *Compiler* adalah program yang menerjemahkan *source code* menjadi instruksi-instruksi yang dapat dijalankan komputer. Saat proses penerjemahan, compiler dapat memberitahukan *error-error* (kesalahan-kesalahan) *syntax* yang mungkin terdapat dalam program. *Error-error syntax* ini harus diperbaiki sebelum *compiler* dapat menerjemahkan *source code*. Setelah program bebas dari *error syntax*, *compiler* membuat sebuah file lain berisi instruksi-instruksi terjemahan.

Compiler Java tidak menerjemahkan *source code* ke instruksi-instruksi bahasa mesin, namun ke dalam instruksi-instruksi *byte code*. Instruksi-instruksi *byte code* ini tidak dapat langsung dijalankan oleh komputer. Kita harus menjalankannya menggunakan *Java Virtual Machine (JVM)*. *JVM* adalah program yang mensimulasikan sebuah komputer yang bahasa mesinnya adalah *byte code*. Gambar berikut mengilustrasikan proses membuat dan menjalankan sebuah program Java.



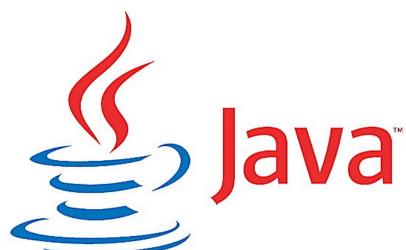
Alasan Java menggunakan *JVM* adalah portabilitas. Portabilitas berarti kita hanya perlu melakukan kompilasi program satu kali ke *bytecode*, lalu kita dapat menjalankan *bytecode* ini pada berbagai jenis prosesor. Dalam bahasa pemrograman yang diterjemahkan ke bahasa mesin langsung, seperti bahasa C, jika kita ingin menjalankan program yang kita tulis pada dua komputer dengan prosesor yang berbeda pabrikan, kita harus mengkompilasi program kita ke masing-masing komputer. Misalkan kita ingin program yang kita tulis berjalan di prosesor Intel dan ARM, kita harus mengkompilasi program dengan *compiler* yang bisa menerjemahkan ke bahasa mesin Intel dan juga mengompilasi program dengan *compiler* yang bisa menerjemahkan ke bahasa mesin ARM. Ini karena setiap prosesor dari pabrikan yang berbeda mempunyai bahasa mesin yang berbeda.

Sejarah Bahasa Java

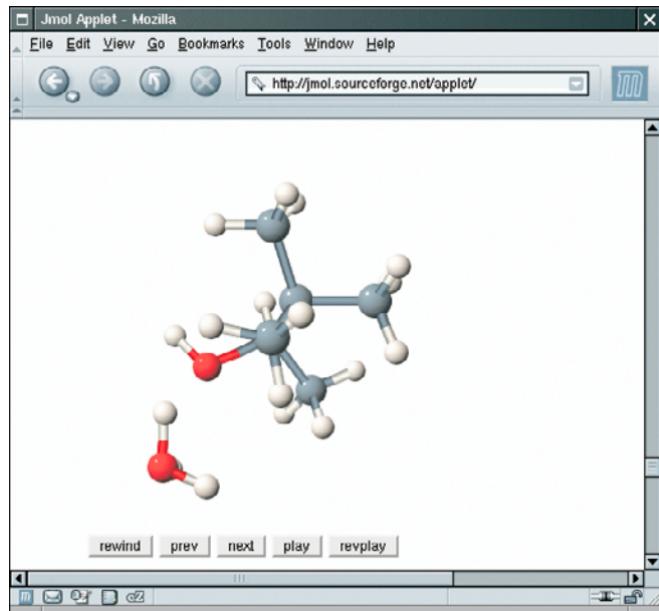


Pada 1991, sebuah tim kerja yang diketuai oleh James Gosling dan Patrick Naughton di perusahaan Sun Microsystem (sekarang dimiliki oleh perusahaan Oracle) mendesain bahasa pemrograman yang diberi nama "Green". Bahasa "Green" diperuntukkan untuk perangkat-perangkat *consumer*, seperti *handphone* atau *Smart TV*. Bahasa ini didesain untuk menjadi sederhana, aman, dan dapat digunakan pada tipe prosesor apapun. Namun, saat itu tidak ada perusahaan pembuat perangkat *consumer* yang tertarik menggunakaninya.

Karena tidak ada yang tertarik menggunakan bahasa "Green" untuk perangkat *consumer*, pada tahun 1994, James Gosling mengubah peruntukan bahasa tersebut untuk digunakan dalam internet. Bahasa "Green" dinamakan ulang menjadi bahasa Java.



Pada pameran SunWorld tahun 1995, tim James Gosling mengenalkan web browser yang dibuat dari bahasa Java yang dapat men-download dan menjalankan program Java kecil yang dikenal dengan applet. Applet memberikan browser kemampuan untuk menampilkan animasi dan berinteraksi dengan pengguna web browser. Gambar berikut memperlihatkan contoh applet yang dapat dijalankan dalam web browser.



Demonstrasi applet ini pada pameran SunWorld mendapatkan puji dan keagungan dari pengunjung pameran. Semenjak itu, Java berkembang cepat menjadi bahasa pemrograman yang populer. Sekarang Java telah menjadi bahasa pemrograman yang paling populer, tidak hanya untuk mengembangkan applet namun juga digunakan untuk membuat aplikasi yang berdiri sendiri. Salah satu penggunaan bahasa Java yang populer saat ini adalah untuk mengembangkan aplikasi *handphone* bersistem operasi Android.

Dari tahun 1995 sampai dengan sekarang, Java telah mengalami beberapa kali pengembangan. Saat ini, Java telah mencapai versi 16. Tabel berikut mendaftar versi-versi Java dan tahun dikeluarkannya versi tersebut.

Versi	Tanggal rilis
JDK Beta	1995
JDK 1.0	23 Januari 1996
JDK 1.1	19 Februari 1997
J2SE 1.2	08 Desember 1998
J2SE 1.3	08 May 2000
J2SE 1.4	06 Februari 2002

Versi	Tanggal rilis
J2SE 5.0	30 September 2004
Java SE 6	11 Desember 2006
Java SE 7	28 Juli 2011
Java SE 8	18 Maret 2014
Java SE 9	21 September 2017
Java SE 10	20 Maret 2018
Java SE 11	25 September 2018
Java SE 12	19 Maret 2019
Java SE 13	17 September 2019
Java SE 14	17 Maret 2020
Java SE 15	15 September 2020
Java SE 16	16 Maret 2021

Aplikasi Java dan Applet

Terdapat dua jenis program yang dapat kita buat dengan Java yaitu aplikasi dan applet. Aplikasi adalah program yang berdiri sendiri. Word processor, spreadsheet, dan *database* adalah contoh-contoh dari aplikasi. Applet adalah program kecil yang berjalan di *software browser web*. Applet berarti program kecil (App adalah singkatan dari *application* yang diterjemahkan menjadi app dan let adalah akhiran kata yang berarti kecil).

Beberapa tahun yang lalu, applet sangat populer karena halaman-halaman web yang ditulis dengan HTML (Hypertext Markup Language) sangatlah terbatas. HTML hanya dapat digunakan untuk *me-layout* dan menuliskan isi berupa teks dan grafik dari halaman web. HTML tidak mempunyai kemampuan untuk melakukan kalkulasi matematika dan berinteraksi dengan pengguna. Applet digunakan untuk mengekstensi keterbatasan HTML ini. Namun, saat ini applet telah berkurang kepopulerannya karena sekarang pengembang web lebih memilih menggunakan JavaScript dibandingkan menggunakan applet untuk menambahkan interaksi pengguna ke halaman web.

1.3 Instalasi dan Pengaturan JDK

Sebelum memulai membuat program dalam Bahasa Java, hal pertama yang harus dilakukan adalah men-*download compiler* Java dan JVM. Keduanya tersedia dalam satu paket yang disebut dengan JDK (Java Development Kit). Terdapat beberapa edisi JDK:

- Java SE - SE singkatan dari *Standard Edition*. Edisi ini menyediakan semua *tool-tool* esensial untuk menulis aplikasi Java dan applet.
- Java EE - EE singkatan dari *Enterprise Edition*. Edisi ini diperuntukkan untuk mengembangkan aplikasi bisnis yang besar.

- Java ME - ME singkatan dari *Micro Edition*. Edisi ini diperuntukkan untuk mengembangkan aplikasi kecil untuk perangkat *consumer* seperti *handphone*, pager, atau *Smart TV*.

Pada praktikum ini, kita akan menggunakan Java SE.

Proses Instalasi JDK:

1. Lakukan pengunduhan *installer* JDK pada *website* resmi Oracle atau melalui *link website* berikut.

<https://www.oracle.com/java/technologies/>

Lalu pada *Technical Details* pilih Java SE.

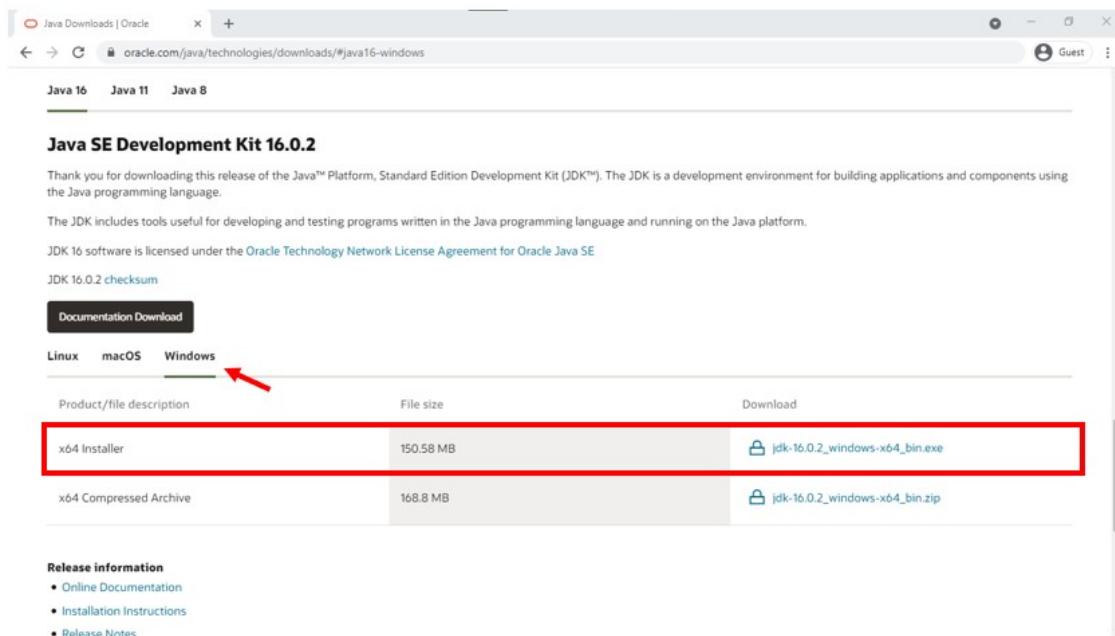
The screenshot shows the Oracle Java Technologies page. At the top, there's a navigation bar with links for Products, Industries, Resources, Support, Events, and Developer. Below that, a banner features a blue cloud icon and the text "Java Is the Language of Possibilities". Underneath the banner, there are two tabs: "Overview" and "Technical Details", with "Technical Details" being the active one. On the left, there's a sidebar with sections for Top Downloads (Java SE, Java EE and GlassFish, Java Card, JDeveloper and ADF, Enterprise Pack for Eclipse) and Newest Downloads (Java SE 17, Java SE 16.0.2, Java SE 11.0.12 (LTS), Java SE 8u301, Java SE Embedded). A red arrow points to the "Java SE" link in the "Top Downloads" section.

Kemudian akan muncul halaman Java Downloads, geser halaman ke bawah sampai Anda menemukan Java SE versi 16.

The screenshot shows the Java Downloads page. At the top, it says "Java SE subscribers have more choices" and "Also available for development, personal use, and to run other licensed Oracle products." There are three tabs: Java 16, Java 11, and Java 8, with Java 16 selected. Below that, it says "Java SE Development Kit 16.0.2" and provides a thank you message about the Java Platform, Standard Edition Development Kit (JDK). It states that the JDK is a development environment for building applications and components using the Java programming language. It also mentions that the JDK includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform. It notes that JDK 16 software is licensed under the Oracle Technology Network License Agreement for Oracle Java SE. It includes a "Documentation Download" button and links for "Linux", "macOS", and "Windows". A table below lists download options for each operating system:

Product/file description	File size	Download
ARM 64 RPM Package	144.87 MB	jdk-16.0.2_linux-aarch64_bin.rpm
ARM 64 Compressed Archive	160.73 MB	jdk-16.0.2_linux-aarch64_bin.tar.gz
x64 Debian Package	146.17 MB	jdk-16.0.2_linux-x64_bin.deb

Sebelum memulai proses pengunduhan, pastikan Anda memilih Sistem operasi sesuai dengan perangkat Anda. Sebagai contoh, dilakukan pengunduhan *installer* Java SE dalam Sistem operasi Windows dengan memilih *link* pada kolom *Download*.

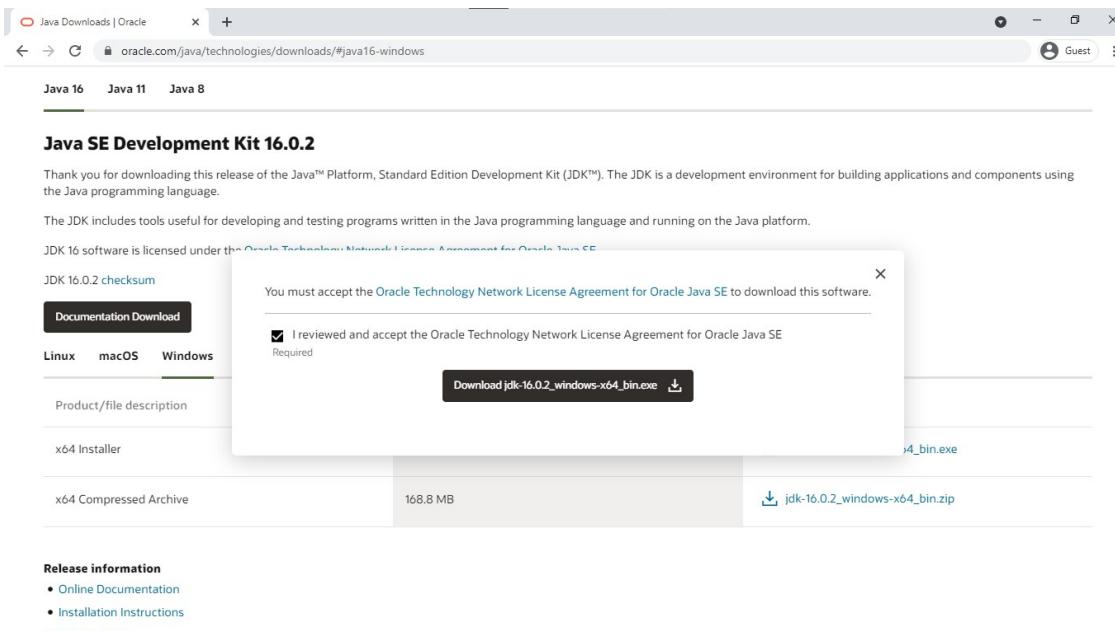


The screenshot shows the Java Downloads page on oracle.com. The URL in the address bar is oracle.com/java/technologies/downloads/#java16-windows. The page displays the Java SE Development Kit 16.0.2 download options. At the top, there are tabs for Java 16, Java 11, and Java 8. Below them, a section for Java SE Development Kit 16.0.2 is shown. A red arrow points to the 'Windows' tab under the download options. The table below lists two download links:

Product/file description	File size	Download
x64 Installer	150.58 MB	jdk-16.0.2_windows-x64_bin.exe
x64 Compressed Archive	168.8 MB	jdk-16.0.2_windows-x64_bin.zip

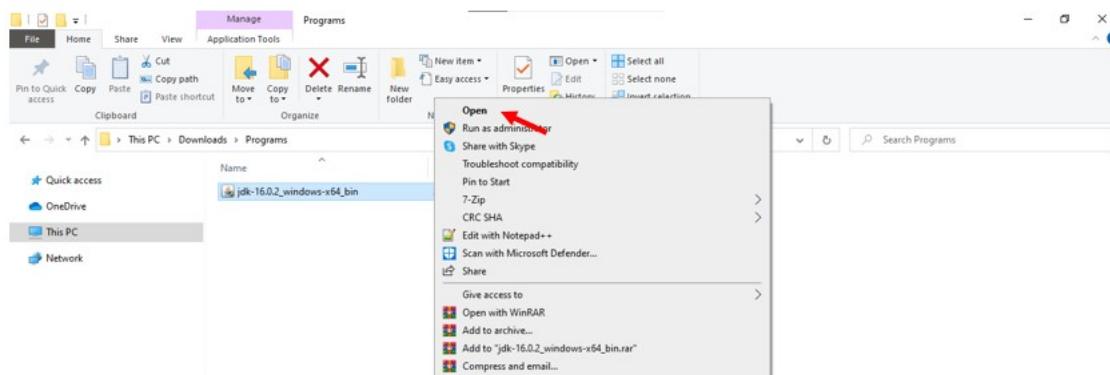
Below the table, there is a 'Release information' section with links to Online Documentation, Installation Instructions, and Release Notes.

Kemudian akan muncul jendela untuk meminta Anda menyetujui perjanjian *Oracle Technology Network* sebelum website mengizinkan Anda untuk melakukan pengunduh Java SE. Untuk menyetujui perjanjiannya, cukup dengan mencentang kotak kecil di atas tombol *link download*. Jika sudah, Anda dapat memulai proses pengunduhan *installer*.



The screenshot shows the Java Downloads page on oracle.com. The URL in the address bar is oracle.com/java/technologies/downloads/#java16-windows. The page displays the Java SE Development Kit 16.0.2 download options. A modal dialog box is overlaid on the page, prompting the user to accept the Oracle Technology Network License Agreement for Oracle Java SE. The dialog contains the following text:
You must accept the Oracle Technology Network License Agreement for Oracle Java SE to download this software.
 I reviewed and accept the Oracle Technology Network License Agreement for Oracle Java SE
Required
A 'Download jdk-16.0.2_windows-x64_bin.exe' button is visible at the bottom of the dialog. The background of the page shows the same download options as the previous screenshot, including the 'Windows' tab and the two download links for the x64 Installer and x64 Compressed Archive.

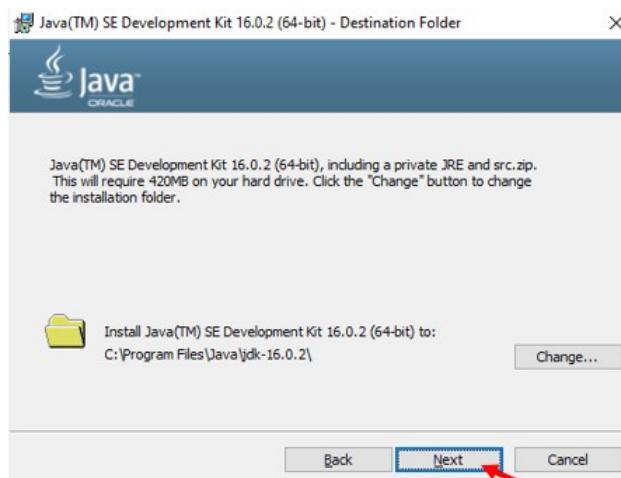
2. Jika pengunduhan telah berhasil, file *installer* akan tersimpan pada direktori perangkat Anda. Kemudian buka file JDK dengan cara mengklik file dua kali atau klik kanan pada file dan pilih Open.



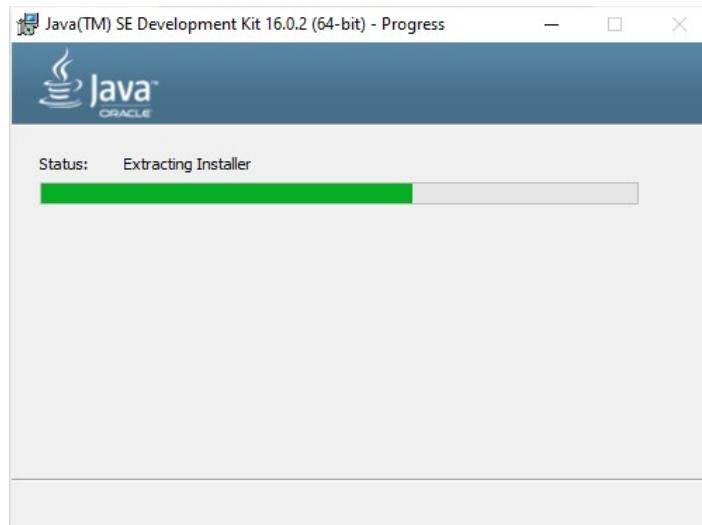
Seketika *installer* di klik akan muncul jendela Setup. Untuk memulai proses instalasi, klik Next pada jendela Setup.



Setelah itu Anda diminta untuk menentukan tempat penyimpanan folder instalasi di direktori perangkat Anda. Secara *default* folder instalasi akan di simpan pada direktori C:\Program Files\Java\jdk-16.0.2\. Jika ingin mengubahnya Anda dapat memilih tombol Change , jika tidak Anda dapat mengklik Next untuk memulai proses instalasi.



Lalu tunggu proses instalasi sampai selesai.



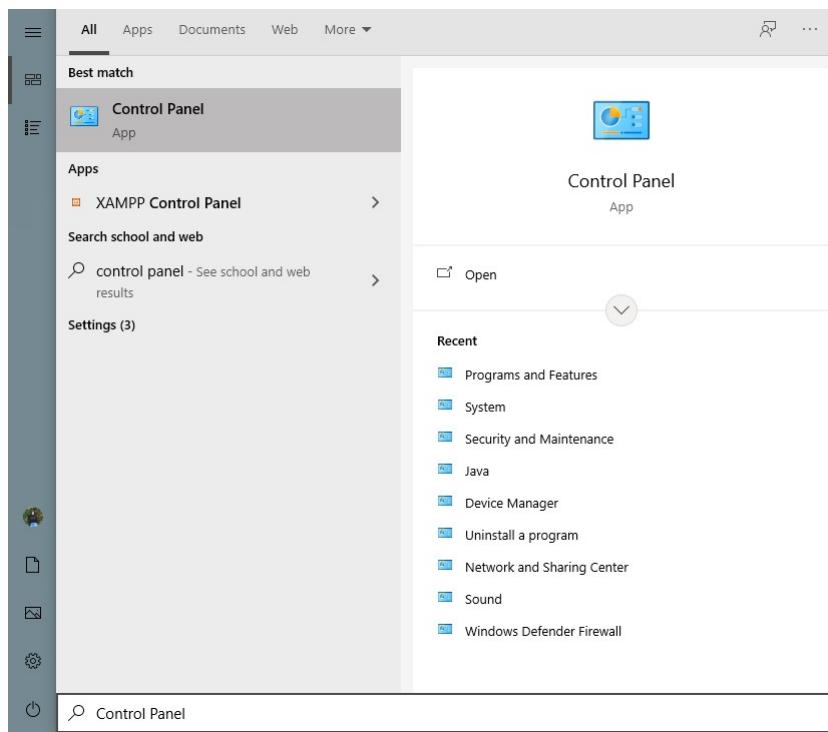
Tahapan proses instalasi berhasil, ketika jendela menampilkan tampilan di bawah ini. Dan kemudian klik *Close* untuk menutup jendela *installer*.



Tahapan Mengatur Path Variabel Environment

Setelah selesai melakukan instalasi JDK, langkah selanjutnya mengatur path JDK pada variable environment. Berikut langkah-langkahnya:

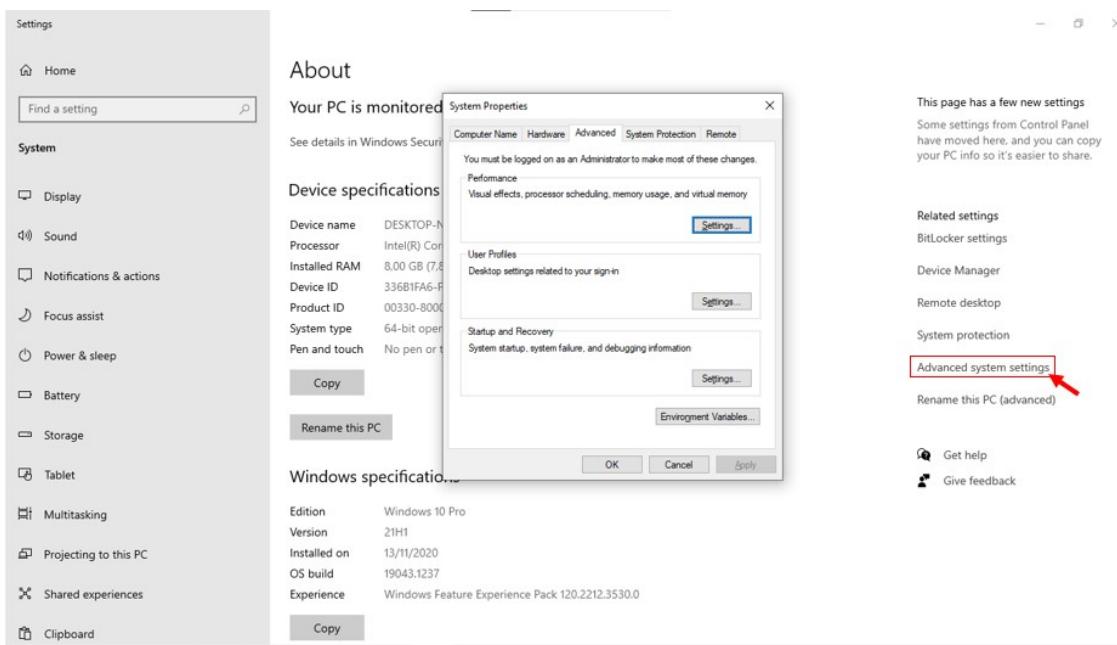
1. Langkah pertama buka Control Panel pada perangkat Anda melalui kolom search di Windows.



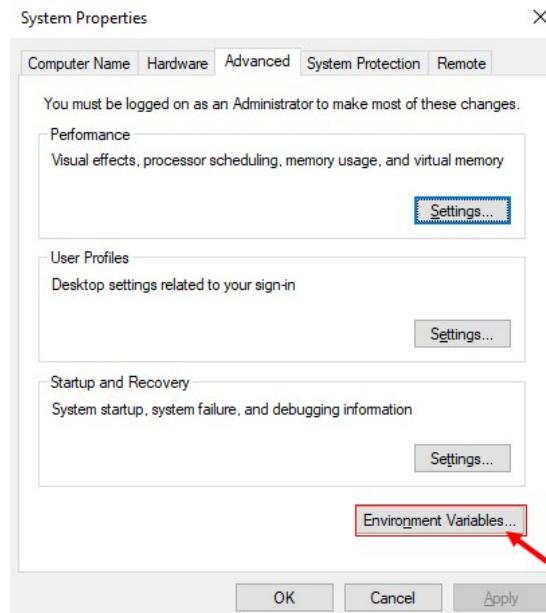
2. Kemudian pada Control Panel pilih System and Security > System.



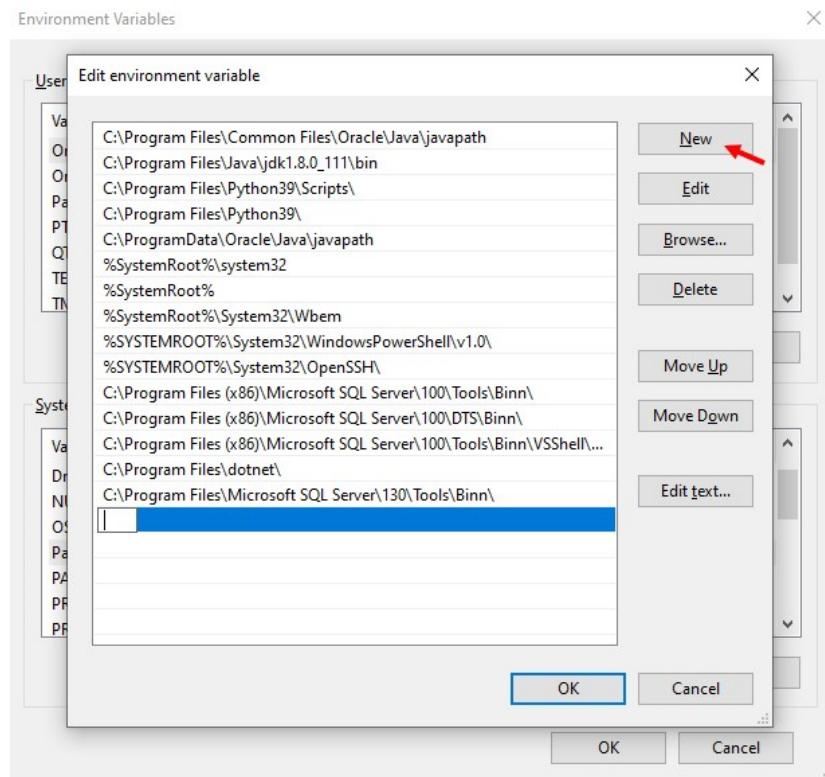
3. Setelah itu, Anda akan masuk kedalam jendela Settings. Pada bagian Related settings klik Advanced system settings sehingga memunculkan jendela System properties seperti tampilan berikut.



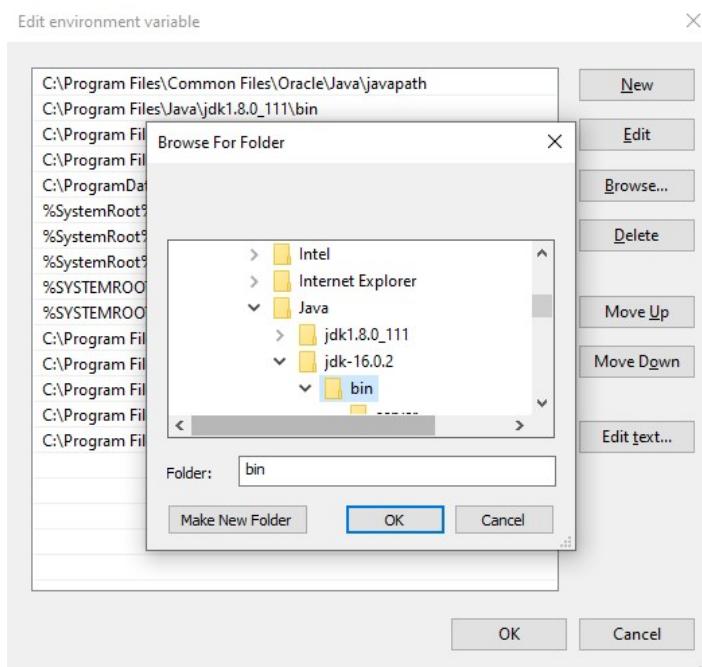
4. Kemudian pada jendela System Properties pastikan Anda memilih menu Advanced, dan klik Environment Variables untuk memulai pengaturan path JDK.



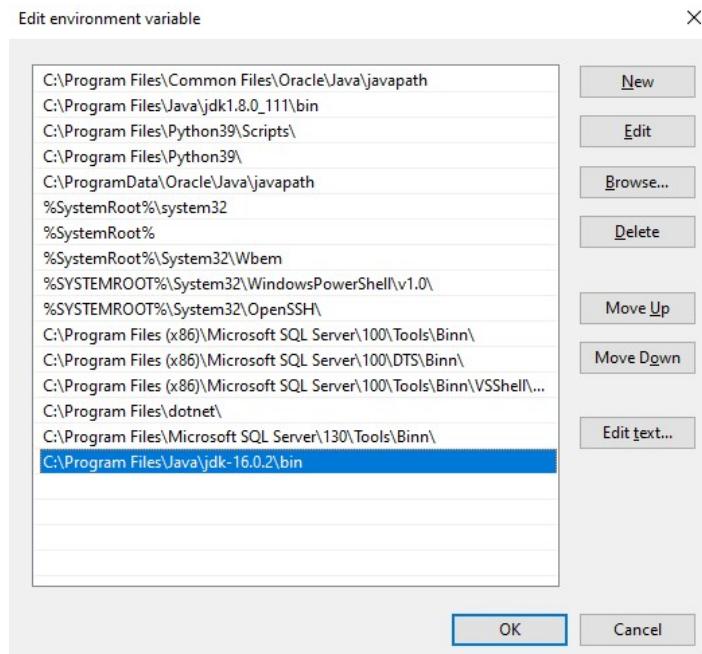
5. Langkah selanjutnya, pada System Variables klik dua kali Variables Path sehingga akan muncul jendela Edit environment variables dan pilih button New untuk membuat path baru.



6. Berikutnya klik button Browse untuk memilih tempat penyimpanan folder instalasi JDK di direktori yaitu pada direktori C:\Program Files\Java\jdk-16.0.2\. Kemudian pilih folder bin pada folder jdk-16.0.2 dan klik OK.



7. Pada jendela Edit environment variables akan muncul path direktori yang sudah kita tambahkan sebelumnya.



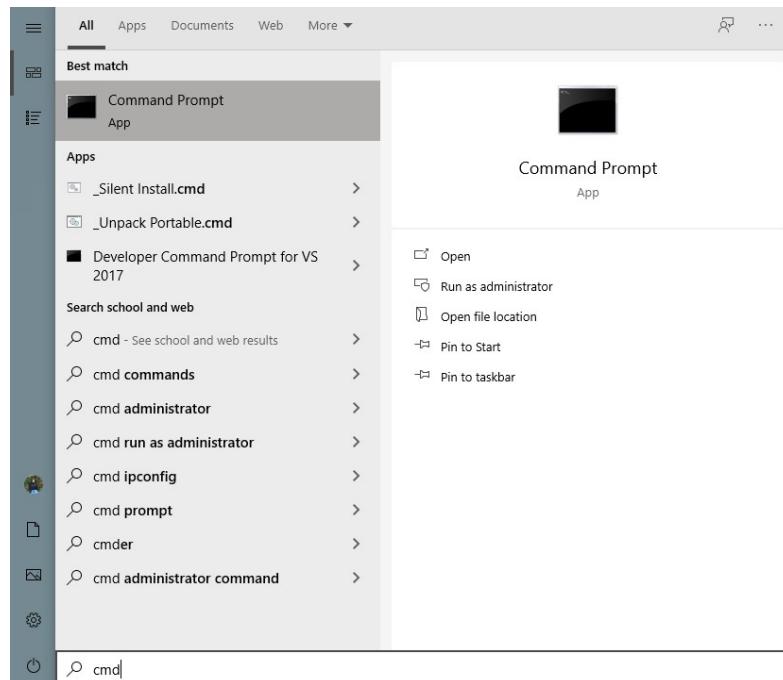
Lalu klik OK.

Dengan ini Anda telah berhasil mengonfigurasi path JDK pada Environment Variable.

Cara mengetahui JDK telah terinstalasi.

Setelah Anda berhasil mengkonfigurasi *path* JDK, langkah selanjutnya Anda harus memastikan bahwa JDK benar-benar sudah terinstal di perangkat Anda. Berikut langkah-langkahnya:

1. Langkah pertama, buka Command Prompt dengan mencarinya di kolom *Search* pada perangkat Anda.

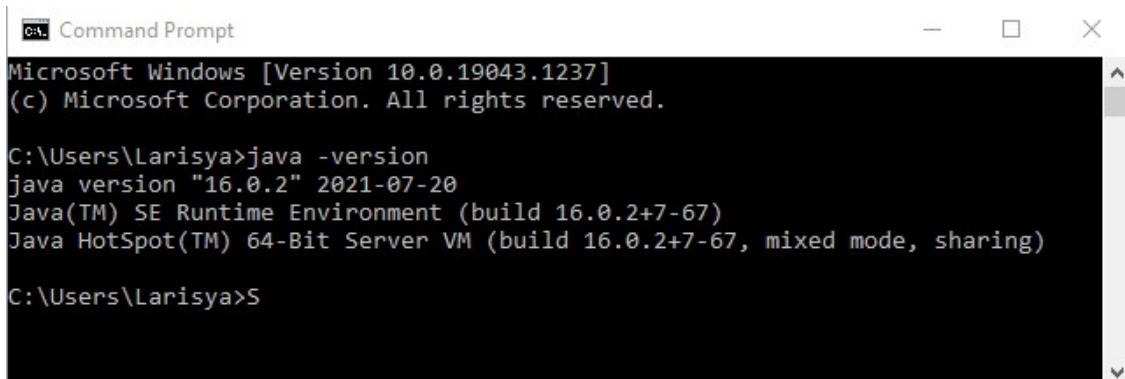


2. Langkah berikutnya ketikkan perintah berikut pada Command Prompt

```
java -version
```

Baris perintah tersebut digunakan untuk meminta sistem melakukan pengecekan versi JDK yang terinstall pada sistem.

Sehingga akan memunculkan tampilan seperti berikut ini



```
Microsoft Windows [Version 10.0.19043.1237]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Larisya>java -version
java version "16.0.2" 2021-07-20
Java(TM) SE Runtime Environment (build 16.0.2+7-67)
Java HotSpot(TM) 64-Bit Server VM (build 16.0.2+7-67, mixed mode, sharing)

C:\Users\Larisya>
```

Dapat dilihat pada Command Prompt bahwa JDK 16.0.2 sudah berhasil diinstal.

1.4 Integrated Development Environment

Source code dari bahasa pemrograman adalah file teks biasa. Kita dapat menggunakan aplikasi teks editor apapun untuk menulis kode Java. Namun semakin kompleks program yang kita tulis, menggunakan teks editor akan menyulitkan dan memperlama proses pembuatan program.

Integrated Development Environment (IDE) adalah aplikasi yang diperuntukkan untuk menulis program. IDE terdiri dari teks editor, *compiler*, *debugger*, dan utilitas-utilitas lainnya yang membantu penulisan program. Dengan IDE, kita tidak perlu menjalankan *compiler* dan JVM dari command prompt. Kedua proses tersebut dapat dilakukan dengan satu klik pada sebuah tombol.

IDE dibuat oleh beberapa organisasi atau perusahaan besar, seperti Visual Studio dari Microsoft, Xcode dari Apple, dan Android Studio dari Google. Terdapat banyak IDE yang mendukung bahasa Java, antara lain IntelliJ IDEA, Eclipse, Netbeans, dan Visual Studio Code.



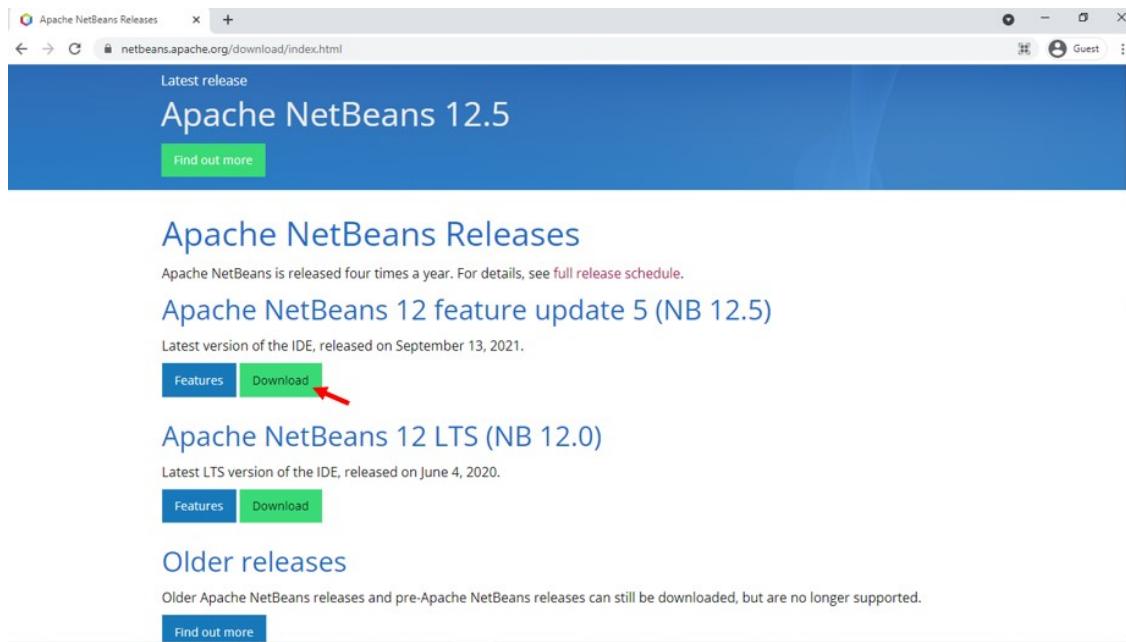
Pada praktikum ini, kita akan menggunakan IDE Apache Netbeans 12.5.

Instalasi Netbeans

1. Lakukan pengunduhan *installer* Apache Netbeans pada website resmi Apache NetBeans atau masuk ke halaman websitenya melalui *link* berikut.

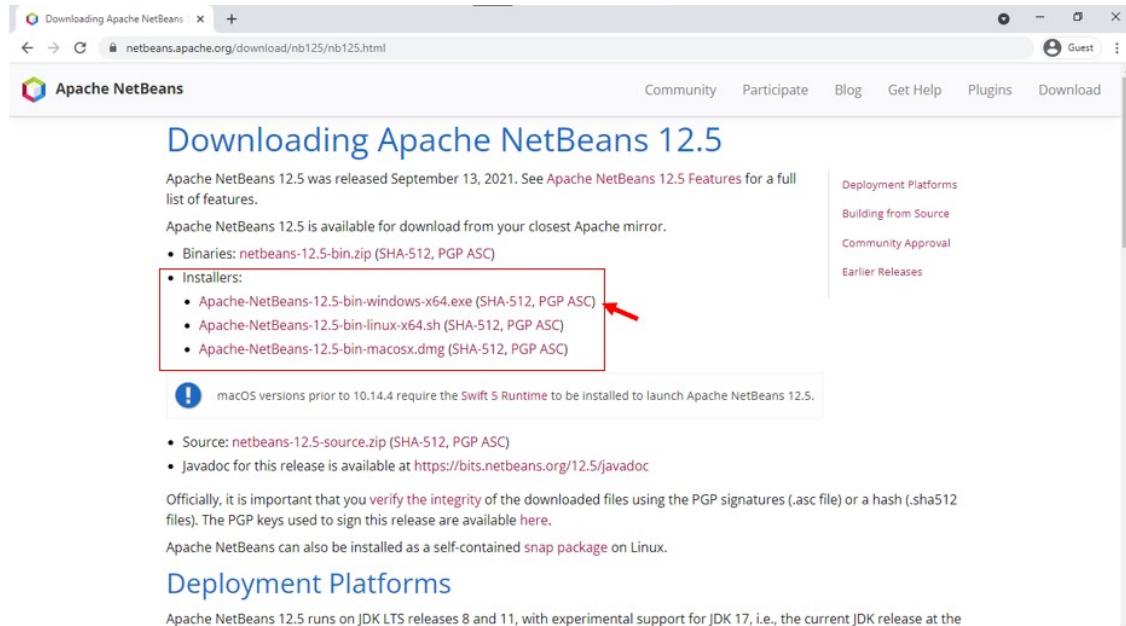
<https://netbeans.apache.org/download/index.html>

Kemudian pada halaman websitenya, klik tombol *Download* pada Apache Netbeans 12 feature update 5 (NB 12.5)



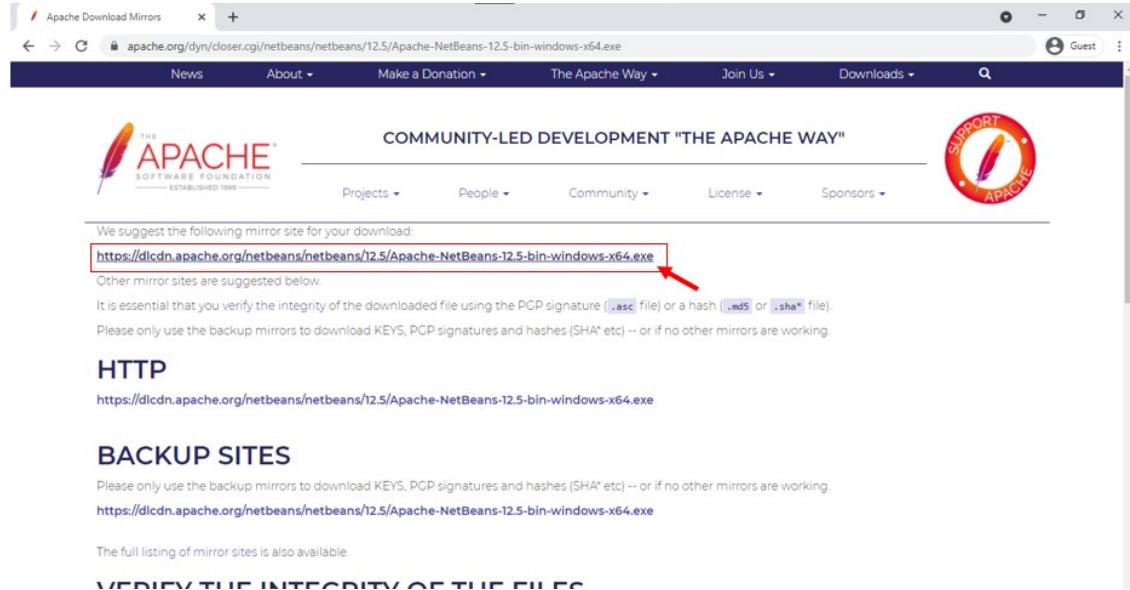
The screenshot shows the Apache NetBeans Releases page. At the top, it says "Latest release" and "Apache NetBeans 12.5". Below that is a green "Find out more" button. The main section is titled "Apache NetBeans Releases" with the sub-section "Apache NetBeans 12 feature update 5 (NB 12.5)". It says "Latest version of the IDE, released on September 13, 2021." There are two buttons: "Features" (blue) and "Download" (green). A red arrow points to the "Download" button. Below this, there's another section for "Apache NetBeans 12 LTS (NB 12.0)" with its own "Features" and "Download" buttons. At the bottom, there's a section for "Older releases" with a "Find out more" button.

Selanjutnya akan muncul halaman yang menampilkan *link download installer* Apache Netbeans. sebelum melakukan pengunduhan, pastikan Anda memilih *link download* sesuai dengan Sistem operasi perangkat Anda. Sebagai contoh akan dilakukan instalasi IDE pada Sistem operasi Windows, maka klik *link download installers* Windows seperti tampilan berikut.



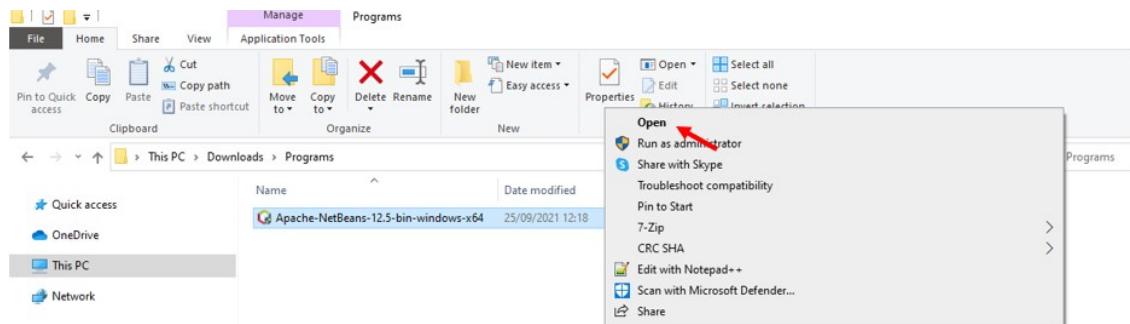
The screenshot shows the "Downloading Apache NetBeans 12.5" page. It says "Apache NetBeans 12.5 was released September 13, 2021. See Apache NetBeans 12.5 Features for a full list of features." It also says "Apache NetBeans 12.5 is available for download from your closest Apache mirror." There are two main sections: "Binaries" and "Installers". The "Installers" section is highlighted with a red box and a red arrow. It lists three options: "Apache-NetBeans-12.5-bin-windows-x64.exe (SHA-512, PGP ASC)", "Apache-NetBeans-12.5-bin-linux-x64.sh (SHA-512, PGP ASC)", and "Apache-NetBeans-12.5-bin-macosx.dmg (SHA-512, PGP ASC)". Below this, there's a note about macOS requiring Swift 5 Runtime. Further down, it lists "Source" and "Javadoc" links. It also mentions that files can be installed as a snap package on Linux. At the bottom, there's a "Deployment Platforms" section and a note about Java versions.

Lalu akan muncul tampilan halaman baru untuk melakukan pengunduhan *installer* Apache Netbeans. proses pengunduhan dimulai dengan cara klik *link download* yang ditunjukkan oleh tanda panah pada gambar berikut ini.

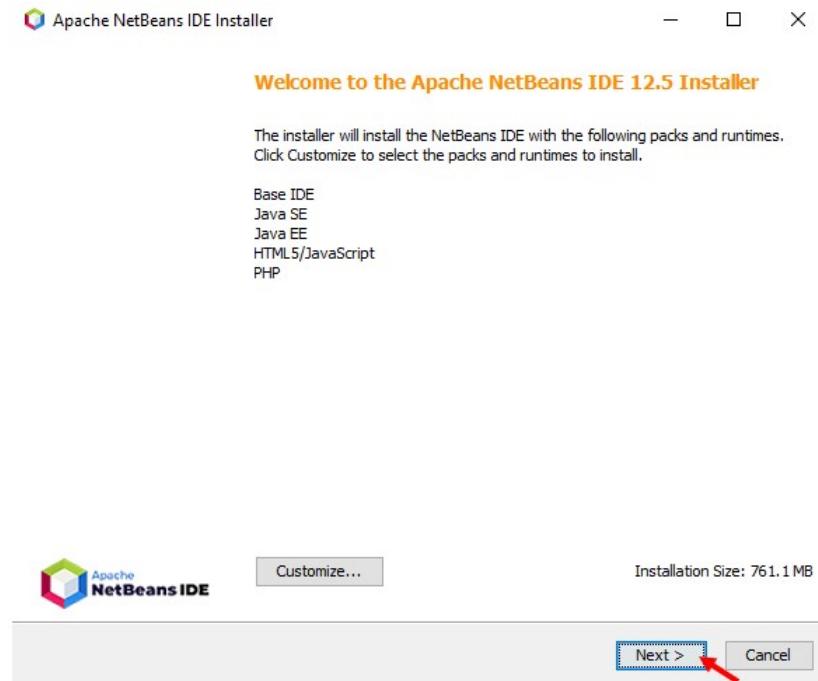


Dan tunggu proses pengunduhan *installer* selesai.

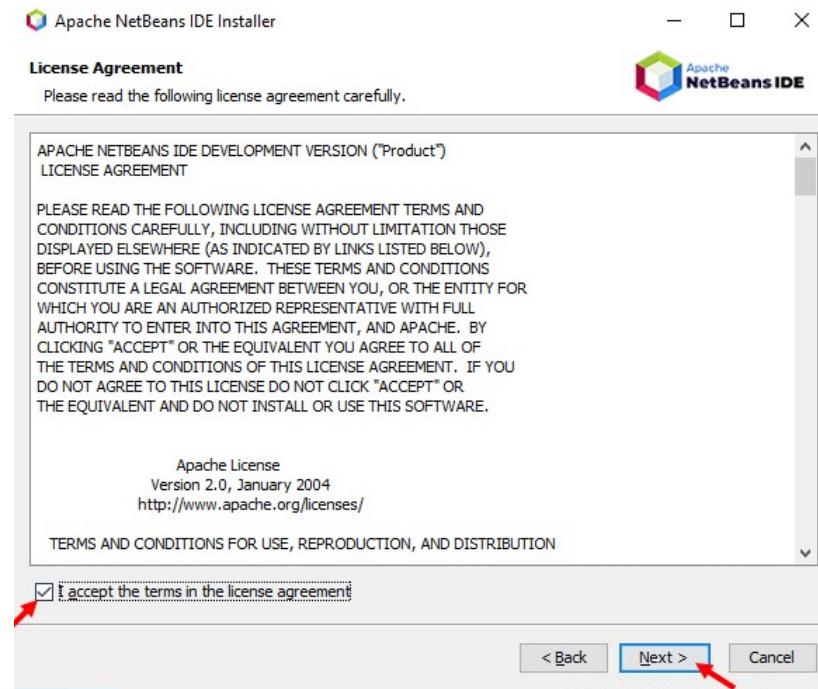
2. Jika pengunduhan telah berhasil, file *installer* akan tersimpan pada direktori perangkat Anda. kemudian buka file tersebut dengan cara mengklik file dua kali atau klik kanan pada file dan pilih Open.



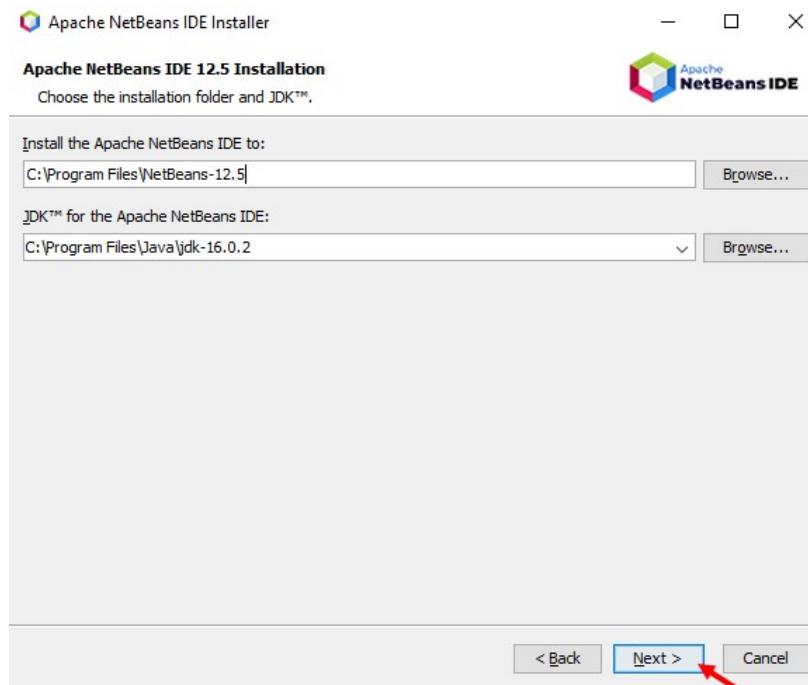
Kemudian tunggu proses konfigurasi *installer* sampai selesai. Setelah selesai, akan muncul jendela Apache Netbeans IDE Installer. Dan klik *Next*.



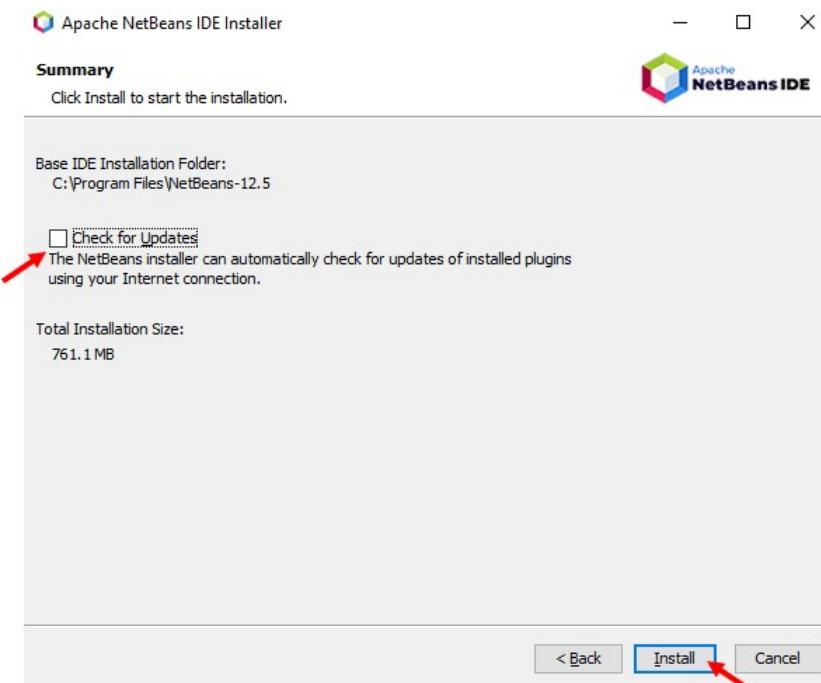
Setelah itu akan muncul jendela untuk meminta Anda menyetujui situasi dan kondisi melalui *License Agreement*. Jika Anda menyetujuinya, klik kotak kecil yang menyatakan bahwa Anda menyetujui situasi dan kondisi sesuai *License Agreement*. Kemudian klik *Next*.



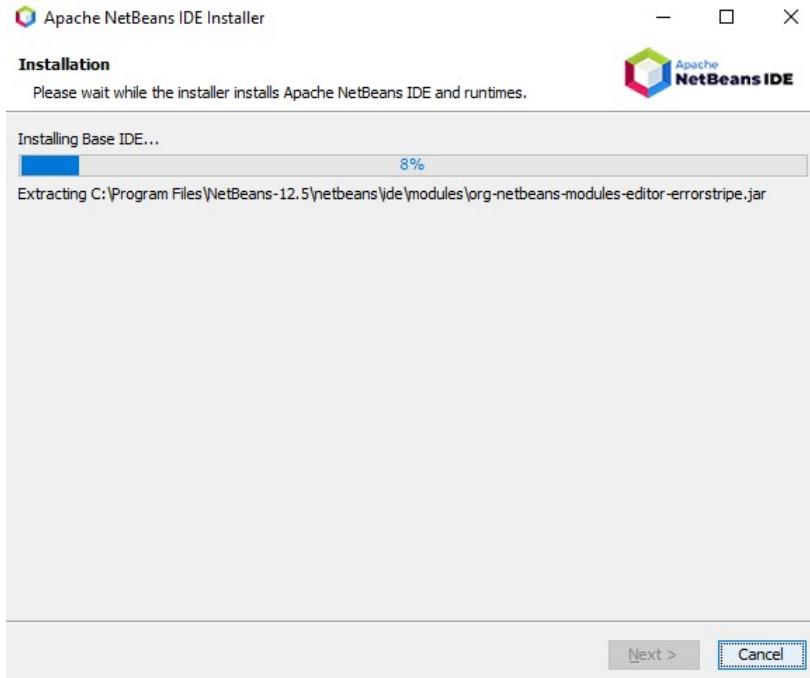
Langkah berikutnya Anda diminta untuk memilih folder untuk instalasi di direktori serta memilih JDK yang sudah terinstall di perangkat anda. Jika sudah klik *Next* untuk masuk ke tahap berikutnya.



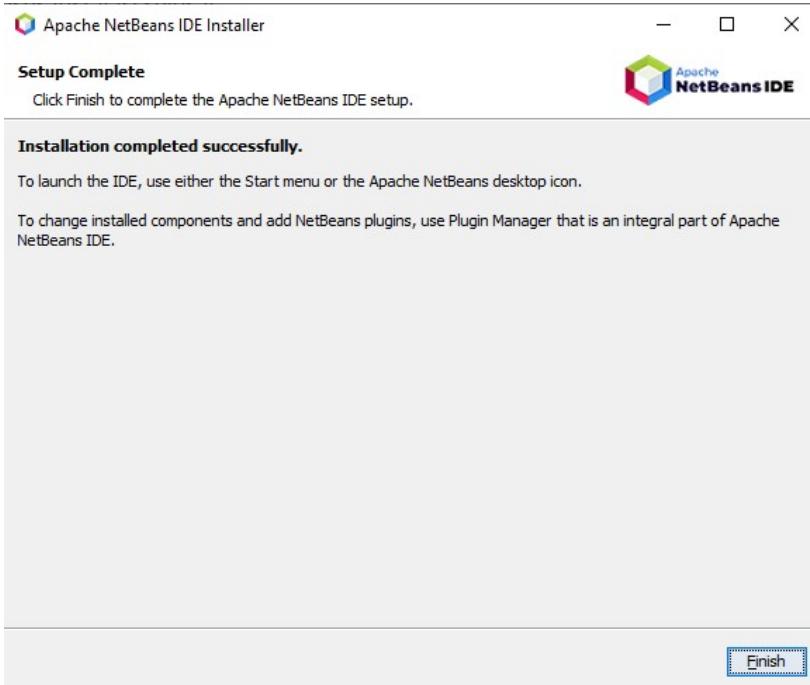
Sebelum melakukan instalasi Apache Netbeans, hilangkan centang untuk sistem melakukan proses pengecekan *Updates* dari Apache Netbeans IDE. Pilihan ini bersifat Opsional. Dan klik tombol *Install* untuk memulai proses instalasi.



Lalu tunggu proses instalasi selesai.



Tahapan proses instalasi berhasil, ketika jendela menampilkan tampilan seperti gambar dibawah ini.

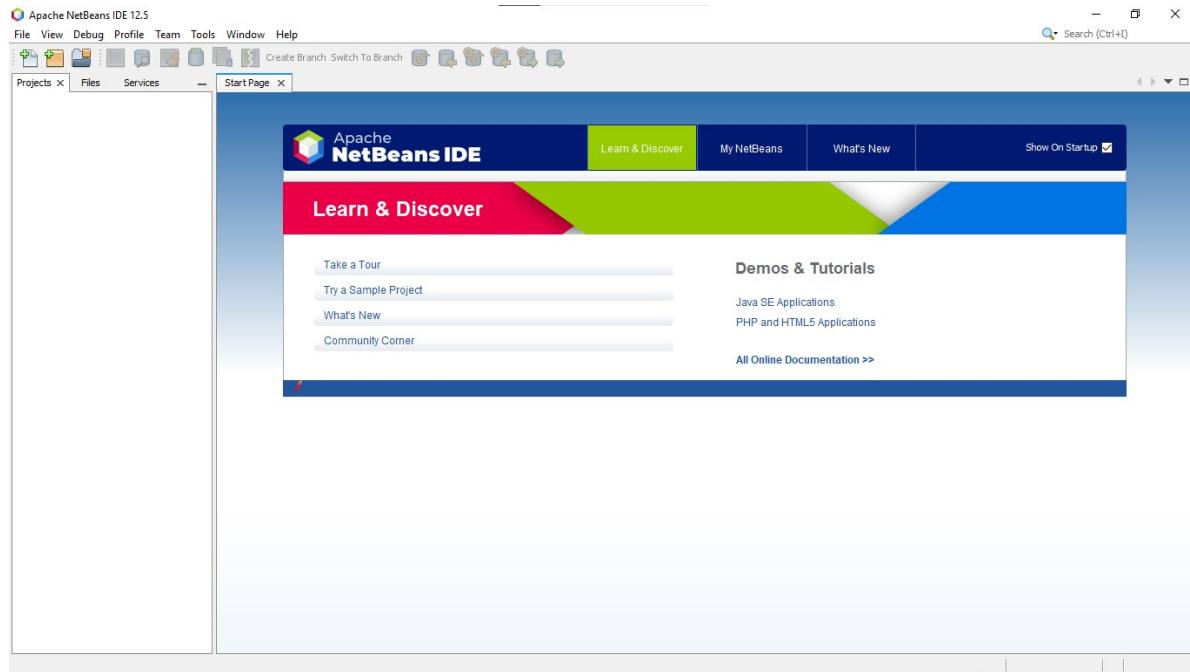


Dan kemudian klik *Finish* untuk menutup jendela *installer*.

Dengan ini proses instalasi Apache Netbeans IDE 12.5 sudah berhasil dilakukan.

Antar Muka IDE

Berikut tampilan antar muka Apache Netbeans IDE 12.5.



1.5 Menulis Program Java Pertama

Pada bagian ini Anda akan menuliskan program Java pertama Anda. Untuk program pertama ini, Anda akan menuliskannya menggunakan teks editor. Ini bertujuan agar Anda dapat mengetahui langkah kompilasi dan langkah menjalankan program Java. Program-program selanjutnya akan kita tulis menggunakan IDE Netbeans.

Ikuti langkah-langkah berikut untuk menuliskan Program Java Pertama Anda.

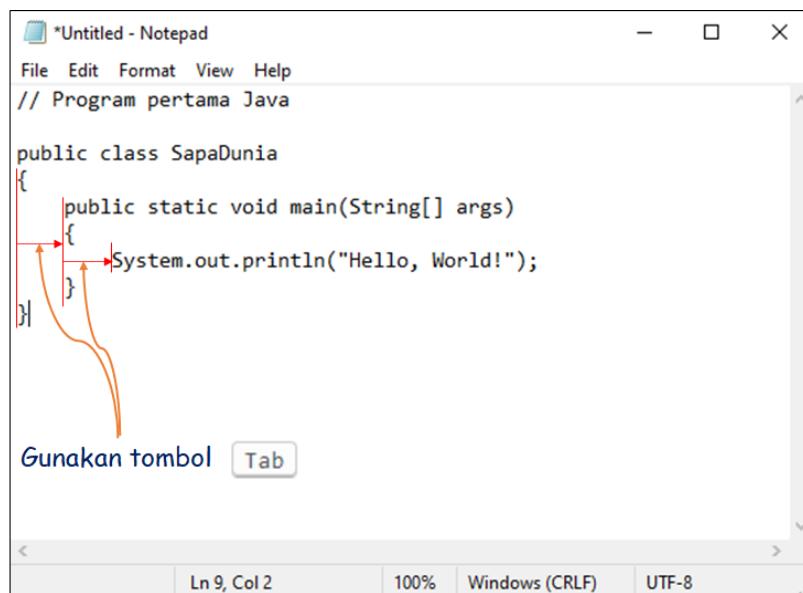
Langkah 1. Ketikkan kode Java berikut pada editor teks seperti Notepad atau Notepad++.

Program (SapaDunia.java)

```
// Program pertama Java

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

Gunakan tombol **Tab** untuk menuliskan teks yang menjorok ke dalam.

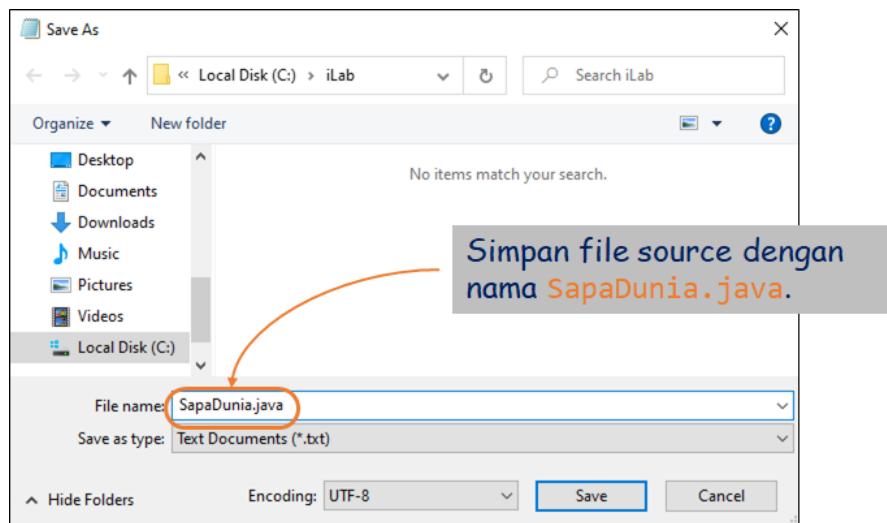


```
// Program pertama Java

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

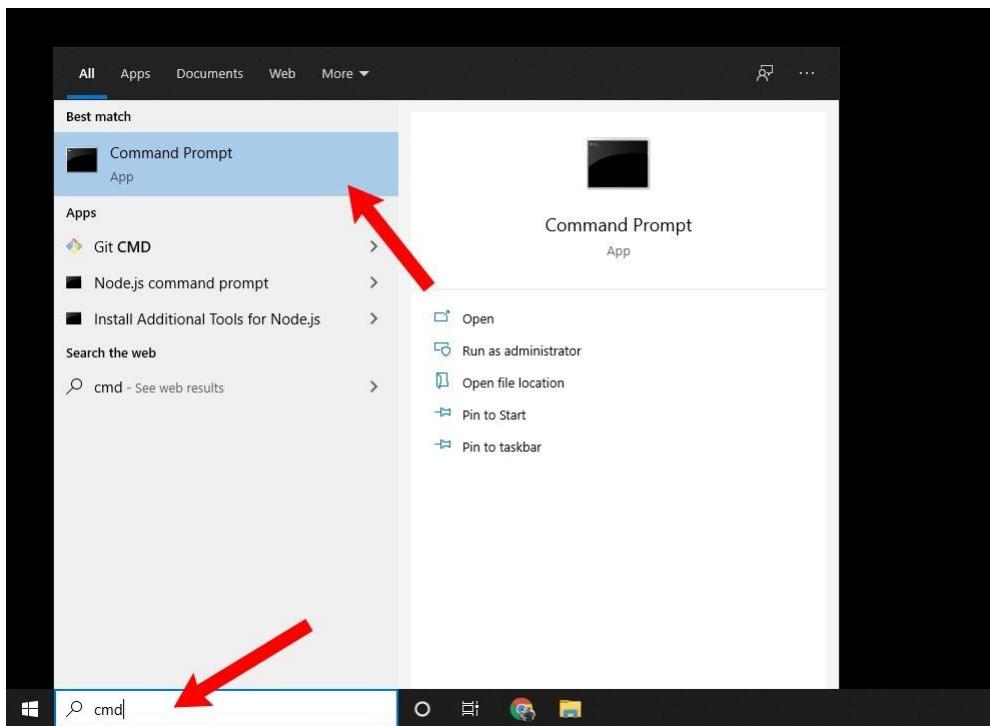
Gunakan tombol Tab

Langkah 2. Simpan program Java dalam file berekstensi .java dengan nama **SapaDunia.java**.



Setiap *source code* Java harus disimpan dalam file dengan ekstensi *.java*. Simpan kode pada file bernama **SapaDunia.java**.

Langkah 3. Buka aplikasi command prompt dan navigasi ke *directory* tempat Anda menyimpan file *source code* **SapaDunia.java**



Gunakan perintah `cd <directory>` untuk pindah ke *directory* tempat *source code* disimpan. Misalkan, jika Anda menyimpan *source code* `SapaDunia.java` di directory `C:\iLab` maka untuk navigasi ke *directory* tersebut, ketikkan perintah berikut.

```
c:\> cd C:\iLab
```

```
c:\> Command Prompt
C:\>cd C:\iLab
C:\iLab>
```

Langkah 4. Kompilasi *source code* program Java dengan menjalankan perintah `javac SapaDunia.java` pada comand prompt.

Sebelum kita dapat menjalankan program Java, kita harus mengkompilasi *source code* Java ke *bytecode*, kita melakukannya dengan mengetikkan perintah berikut pada command prompt.

```
javac NamaSourceCode.java
```

Mengkompilasi *source code* Program 1-1, dengan menjalankan perintah berikut.

```
javac SapaDunia.java
```

Compiler akan memberikan sebuah file bernama `SapaDunia.class` yang berisi hasil terjemahan *source code* dalam Java *byte code*.

Kompilasi file source dengan perintah:
javac SapaDunia.java.

```
C:\ Command Prompt
C:\iLab>javac SapaDunia.java
C:\iLab>dir
 Volume in drive C has no label.
 Volume Serial Number is 7A83-E35E

 Directory of C:\iLab

25/08/2021 17:04    <DIR>      .
25/08/2021 17:04    <DIR>      ..
25/08/2021 17:04           425 SapaDunia.class
25/08/2021 14:21           159 SapaDunia.java
                           2 File(s)   584 bytes
                           2 Dir(s)  717.781.254.144 bytes free

C:\iLab>
```

Compiler akan memberikan file bytecode berekstensi class: **SapaDunia.class**.

Langkah 5. Jalankan bytecode dari program pada JVM.

Menjalankan *bytecode* dari program pada JVM dengan mengetikkan perintah berikut pada command prompt.

```
java NamaSourceCode
```

File Java *bytecode* program pertama, `ProgramPertama.class`, kita jalankan dengan perintah berikut.

```
java ProgramPertama
```

Perhatikan, kita tidak mengetikkan ekstensi `.class` ketika menjalankan perintah `java`.

Saat program berjalan, program akan menampilkan teks berikut pada command prompt.

```
Hello, World!
```

Jalankan file bytecode dengan perintah: **java SapaDunia**.

```
C:\ Command Prompt
C:\iLab>java SapaDunia
Hello, World!
C:\iLab>
```

JVM akan mengeksekusi program **SapaDunia** yang menampilkan output pesan "Hello, World!"

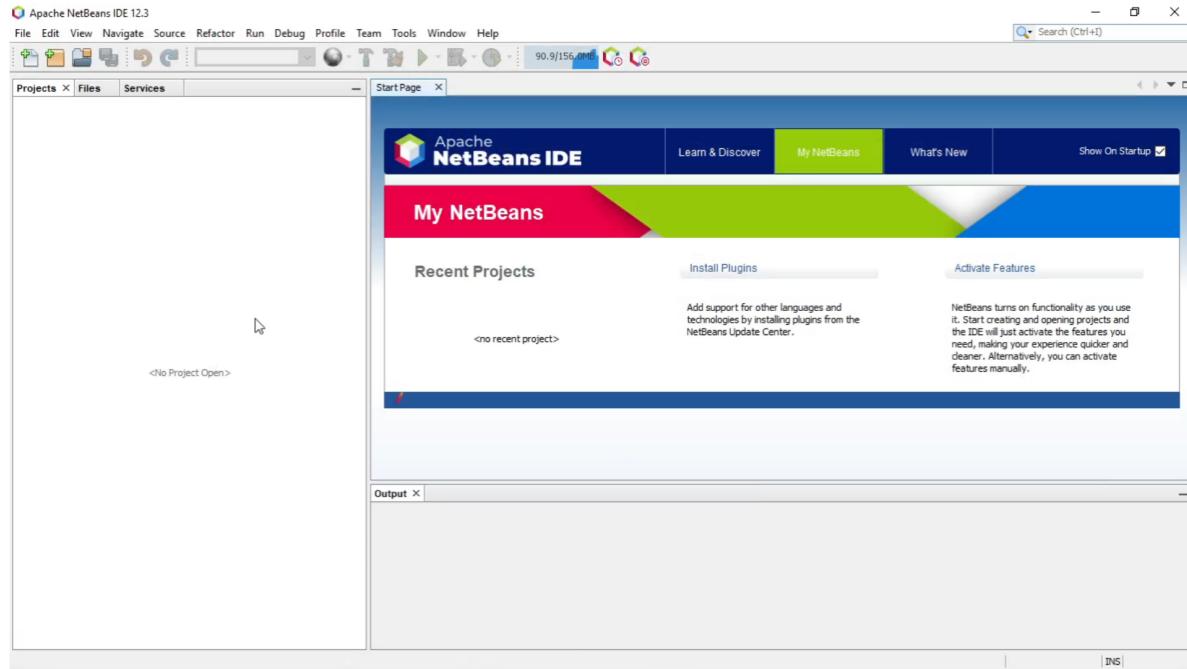
Teks yang ditampilkan oleh program pada command prompt disebut sebagai *output* program.

Selamat! Dengan ini Anda telah berhasil menuliskan program Java pertama Anda.

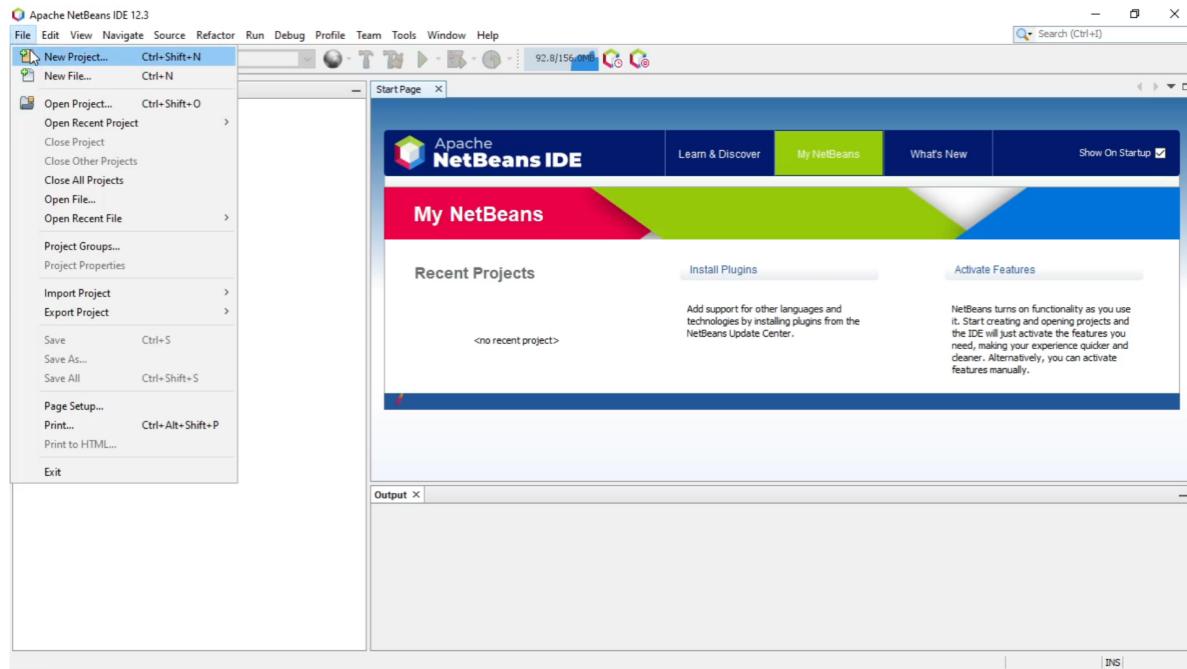
Menulis dan Menjalankan Program Menggunakan IDE

Ikuti langkah-langkah berikut untuk menuliskan Program Java Pertama Anda menggunakan Apache Netbeans IDE.

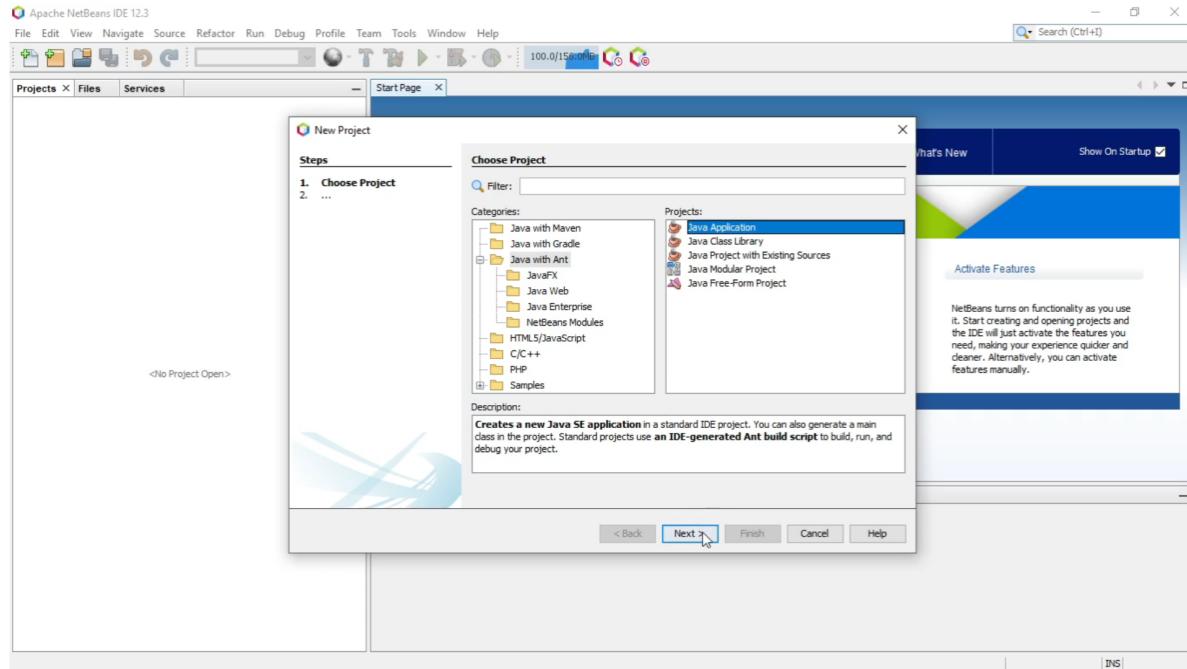
Langkah 1. Buka Apache Netbeans IDE Anda



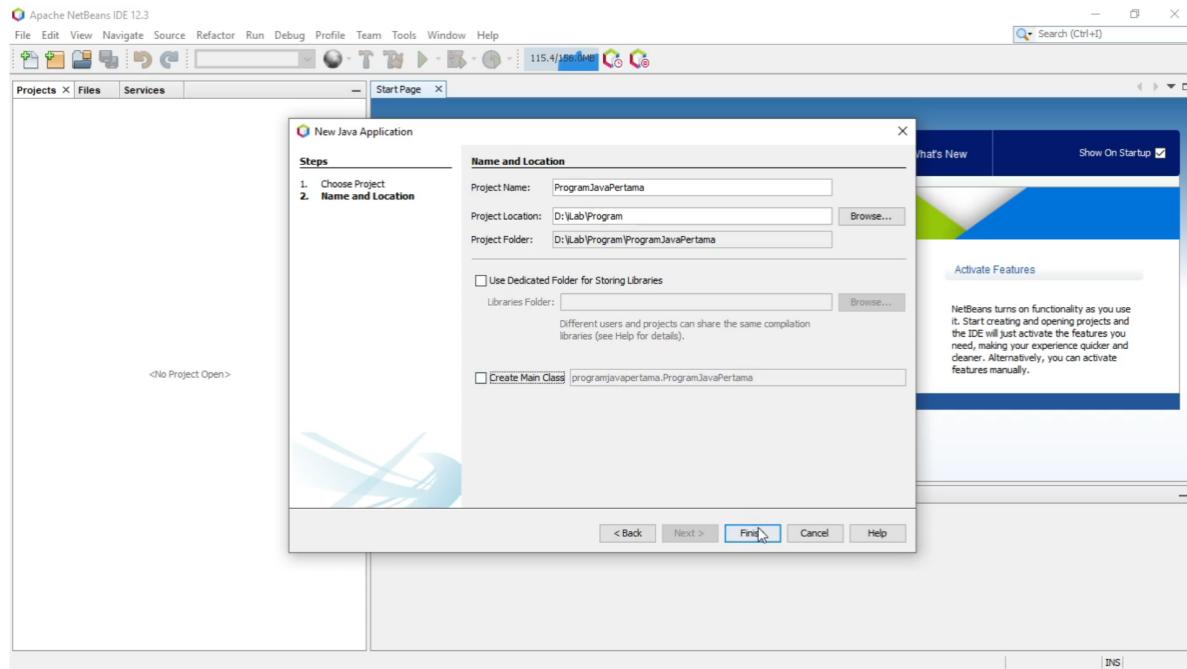
Langkah 2. Buat Project baru dengan memilih File pada Menu. Kemudian pilih New Project.



Langkah 3. Pilih Java with Ant pada kolom *Categories* dan Java Application pada kolom *projects*. Lalu klik Next.

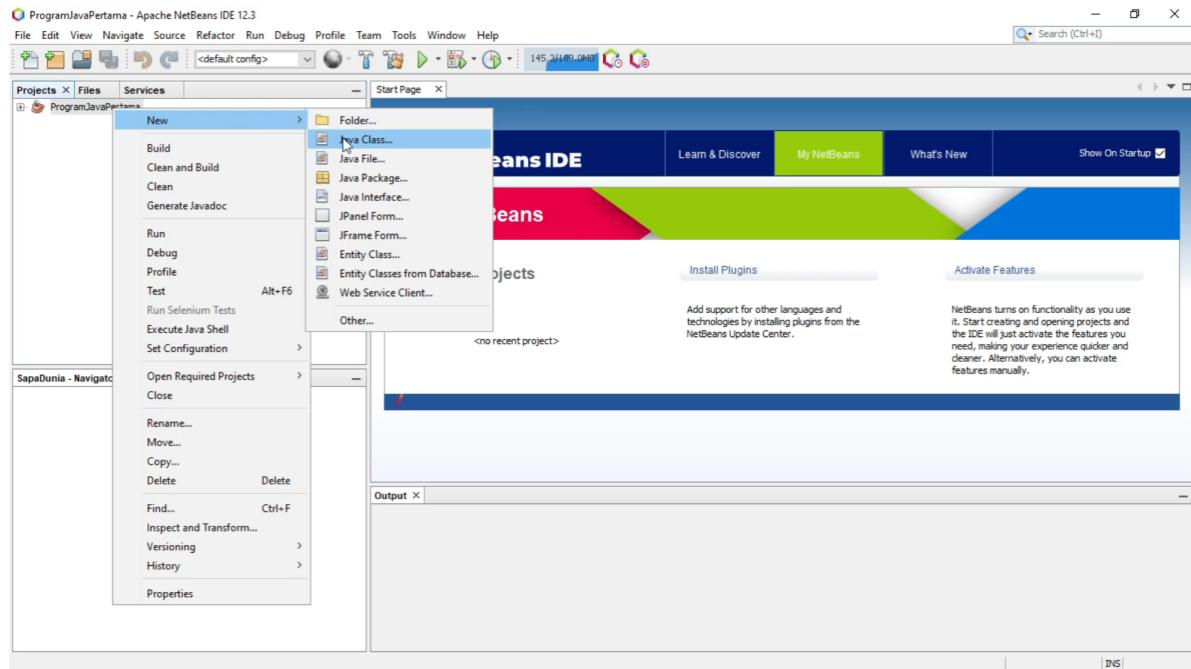


Langkah 4. Selanjutnya ubah nama *project* program menjadi ProgramJavaPertama. Dan hapus centang pada *Create Main Class*. Kemudian klik *Finish*.

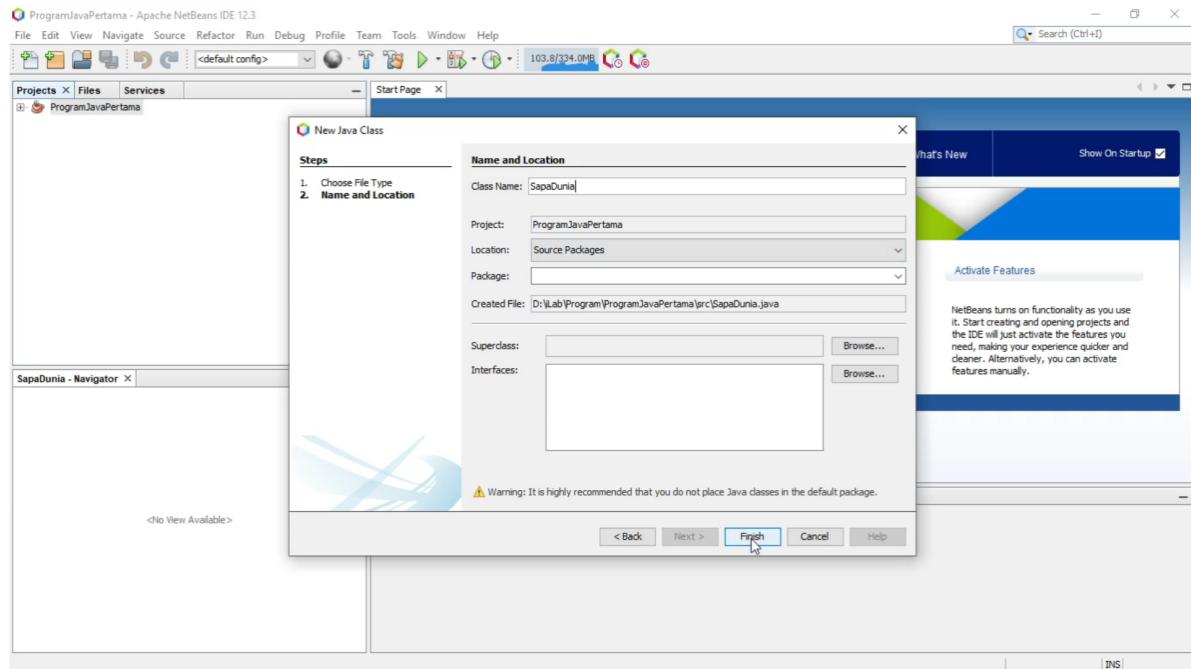


Tunggu IDE selesai membuat *project* baru.

Langkah 5. Klik kanan project ProgramJavaPertama lalu pilih New dan klik Java Class.



Langkah 6. Ketikkan Nama *class* dengan nama SapaDunia. Lalu klik *Finish*.



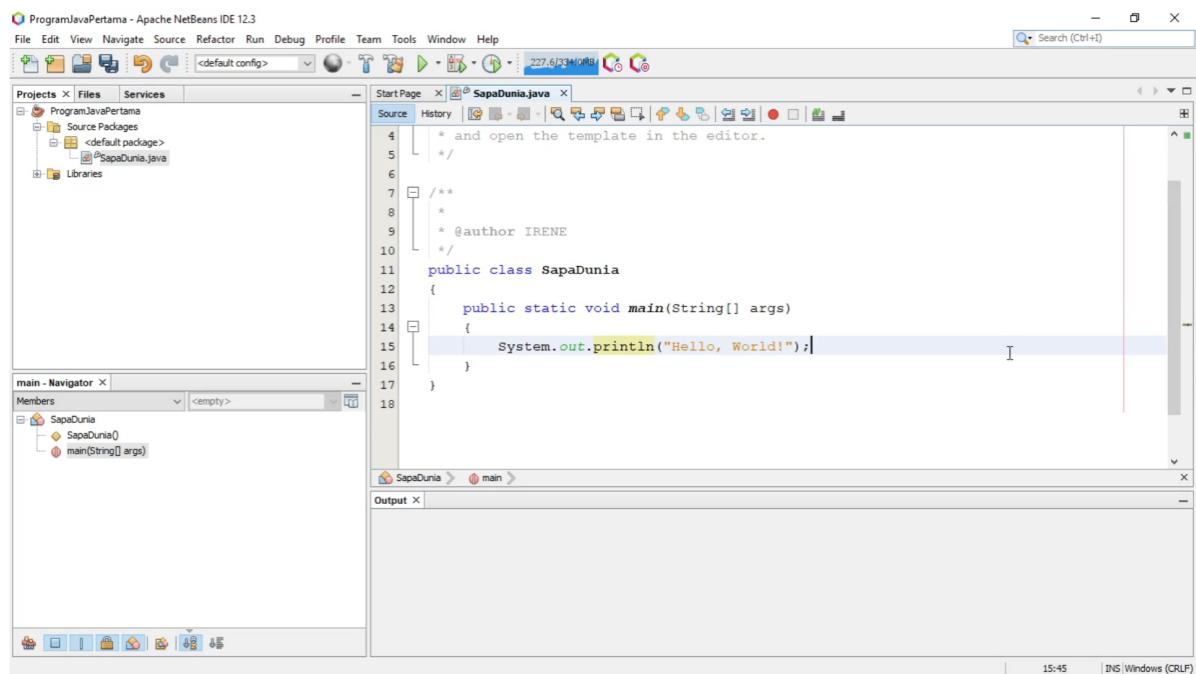
Langkah 7. Ketikkan kode program pada *Source Editor*

Program (*SapaDunia.java*)

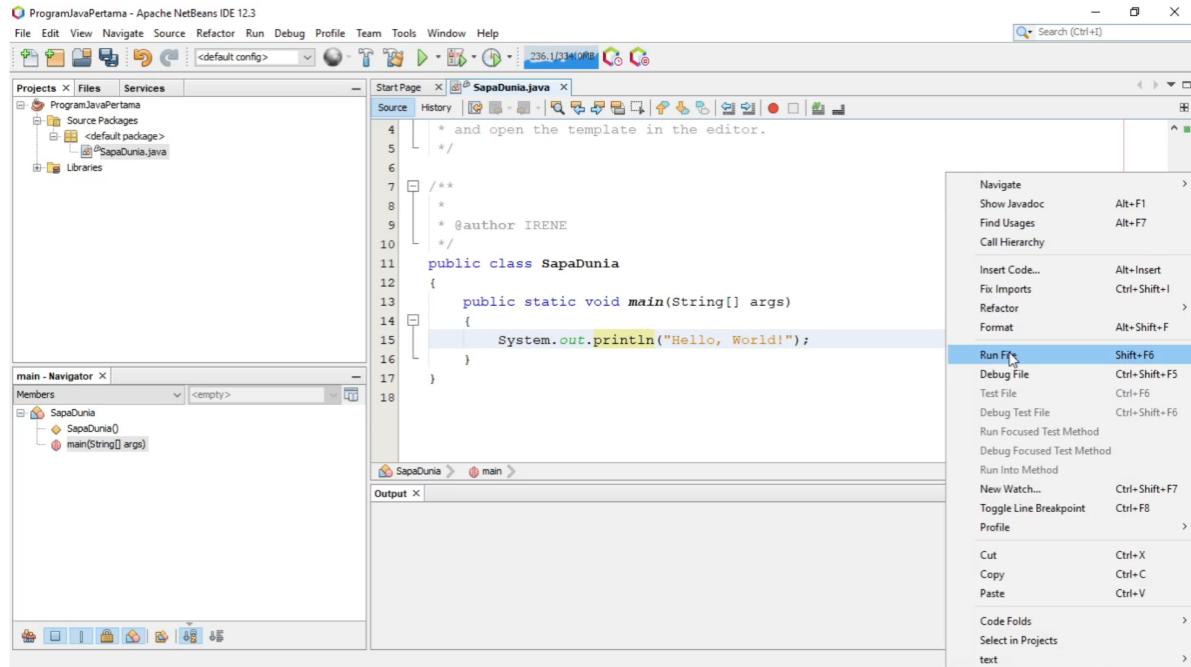
```
// Program pertama Java

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

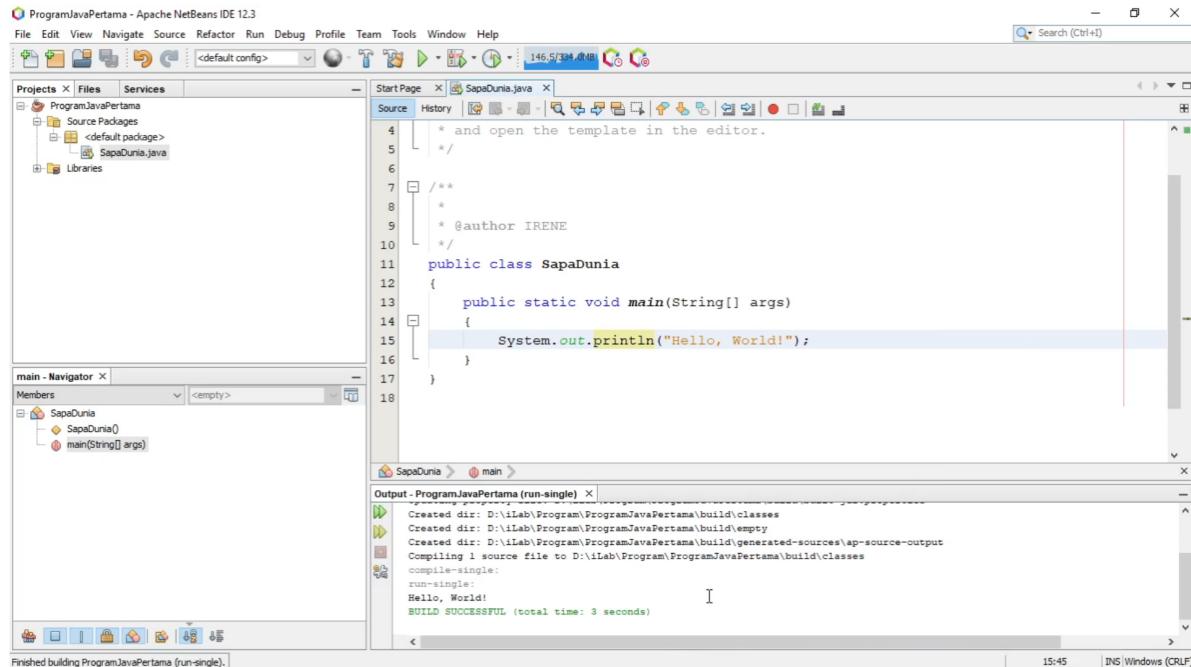
Seperti gambar berikut.



Langkah 8. Jalankan program dengan klik kanan sembarang pada Source Editor lalu pilih Run File.



Maka akan muncul output program yaitu Hello,World! seperti tampilan berikut.



Selamat! Dengan ini Anda telah berhasil menuliskan program Java pertama Anda menggunakan Apache Netbeans IDE.

1.6 Menganalisa Program Pertama

Kita akan menganalisa `sapaDunia.java`. Kode berikut adalah program `SapaDunia.java` yang kita tulis pada bagian sebelumnya.

```
//Program pertama Java

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

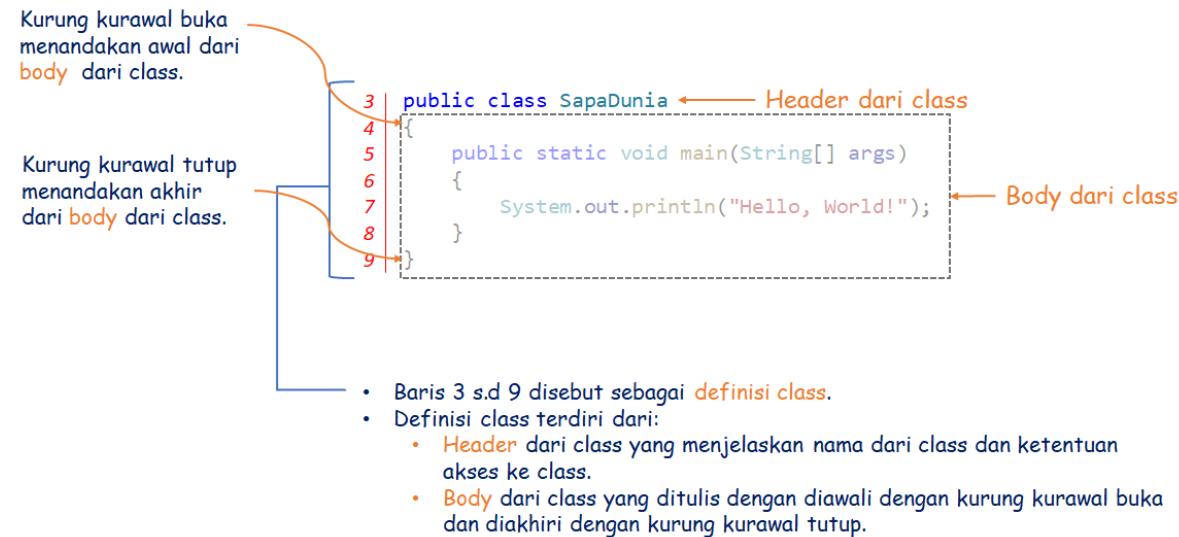
Pada baris 1:

```
// Program pertama Java
```

Teks setelah `//` adalah *comment*. *Compiler* mengabaikan semua teks dari tanda `//` sampai dengan akhir baris. Kita dapat menulis apapun sebagai *comment*. *Comment* membantu pembaca program untuk memahami apa yang dikerjakan oleh program.

Baris 2 adalah baris kosong. Kita menambahkan baris kosong untuk memudahkan kita membaca kode program.

Baris 3 sampai dengan 9 disebut sebagai definisi *class*. Definisi *class* terdiri dari *header* dan *body*. Kita dapat mengartikan *header* sebagai judul dari definisi *class* dan *body* sebagai isi dari definisi *class*. Baris 3 adalah *header* dari *class* ini. Baris 4 sampai dengan baris 9 adalah *body* dari *class*. Baris 4 adalah tanda kurung kurawal buka yang menandakan awal dari *body class* ini. Baris 9 adalah tanda kurung kurawal tutup yang menandakan akhir dari *body class* ini. Gambar berikut mengilustrasikan konstruksi penulisan dari definisi *class*.



Kita akan membahas mengenai *class* secara detail nanti pada beberapa bab selanjutnya. Untuk saat ini, hal yang perlu kita ketahui mengenai *class* adalah setiap program Java harus setidaknya memiliki satu definisi *class* dan salah satu kegunaan dari *class* adalah sebagai wadah dari sebuah aplikasi. Ingat aplikasi adalah program yang berdiri sendiri.

Header dari *class* pada baris ke 3:

```
public class SapaDunia
```

Terdiri dari tiga kata: `public`, `class`, dan `SapaDunia`. Masing-masing kata ini mempunyai arti sebagai berikut:

- `public` menandakan bahwa akses ke *class* ini tidak terbatas. Dengan kata lain, kata `public` menandakan bahwa *class* ini terbuka untuk *public* (terbuka untuk umum). Kata ini harus ditulis dalam huruf kecil semua. Kita akan membahas ini nanti pada pembahasan *class*.
- `class` menandakan bahwa kita menuliskan definisi *class*. Kata ini harus ditulis dalam huruf kecil semua.
- `SapaDunia` menandakan nama dari *class*. Nama dari *class* dapat kita tentukan sendiri.

Gambar berikut mengilustrasikan bagian-bagian *header* dari *class* pada program `SapaDunia`.



Pada contoh ini, kita menuliskan aplikasi kita dalam wadah `public class` bernama `ProgramPertama`. Terdapat dua hal penting yang perlu kita ketahui mengenai `public class` dalam Java:

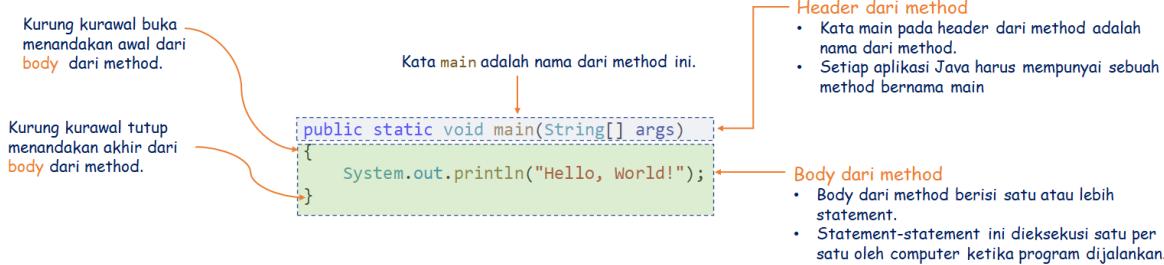
- Kita dapat membuat lebih dari satu *class* dalam sebuah file, tapi kita hanya dapat mempunyai paling banyak satu `public class` dalam satu file *source code* Java.
- Nama dari `public class` harus sama dengan nama file *source code*. Pada contoh ini, kita mempunyai program dengan `public class` bernama `ProgramPertama` dan menyimpannya dalam sebuah file *source code* bernama `ProgramPertama.java`

Di dalam *body* dari *class* pada baris 5 s.d 8 kita menuliskan sebuah definisi *method*. Definisi *method* seperti definisi *class*, memiliki bagian *header* dan *body*. Baris 5 adalah *header* dari *method* ini. Baris 6 adalah tanda kurung kurawal yang menandakan awal dari *body* dari *method*. Baris 7 adalah *body* dari *method*. dan baris 8 adalah tanda kurung kurawal tutup yang menandakan akhir dari *body* dari *method*.

Header dari *method* pada baris 5 dituliskan dengan *syntax* berikut.

```
public static void main(String[] args)
```

Mirip dengan *header* pada definisi *class*, *header* dari definisi *method* terdiri dari nama dan informasi-informasi lainnya. Kata `main` adalah nama dari *method* ini. Kita akan membahas arti kata-kata lainnya yang terdapat pada *header method* ini pada Bab 5. Untuk saat ini, yang perlu kita perhatikan adalah nama dari *method* ini adalah `main` dan kata-kata lainnya diperlukan untuk penulisan *header* dari definisi *method*.



Pada `body` dari `method main`, baris 7, kita mempunyai:

```
System.out.println("Hello, World!");
```

Baris ini adalah *statement*. *Statement* adalah instruksi yang kita perintahkan untuk komputer kerjakan. penulisan *statement* harus diakhiri dengan tanda titik koma (;). *Statement* pada baris 7 ini memerintahkan komputer untuk mencetak teks "Hello, World" tanpa tanda kutip ganda ke layar.



Hal-hal Penting dalam Program Java

Berikut adalah ringkasan dari hal-hal yang perlu kita ketahui dalam program pertama kita dan juga dalam memulai membuat program lainnya:

- Semua program Java harus disimpan dalam sebuah file dengan ekstensi `.java`.
- File `.java` dapat berisi lebih dari satu `class`, namun hanya dapat mempunyai satu `public class`.
- Jika file `.java` berisi sebuah `public class`, maka nama file tersebut harus sama dengan nama dari `public class`. Misalkan, jika dalam file berisi `public class` bernama `Pizza` maka file tersebut harus dinamakan dengan `Pizza.java`.
- Setiap program aplikasi Java harus mempunyai *method* bernama `main`.
- *Statement-statement* harus diakhiri dengan tanda titik koma (;).

Kerangka Program

Kita akan membahas mengenai *class* dan *method* nanti pada bab-bab selanjutnya. Untuk saat ini, kita akan menggunakan kerangka program berikut untuk menulis program Java.

```
public class NamaClass
{
    public static void main(String[] args)
    {
        ...
    }
}
```

Proses pembuatan program baru, kita dapat dengan mengubah `NamaClass` dengan nama yang kita mau lalu kita menuliskan *statement-statement* dalam *body* dari *method* `main`. Kemudian file *source code* kita simpan dalam file berekstensi .java dengan nama file sama dengan nama *class* yang kita berikan.

```
public class NamaClass
{
    public static void main(String[] args)
    {
        ...
    }
}
```

Ubah ini dengan nama yang diinginkan. Pastikan file disimpan dengan nama yang sama.

Tuliskan statement-statement disini.

1.7 Comment

Comment (komentar) adalah catatan singkat berisi penjelasan dari baris program ataupun bagian dari program. *Comment* adalah bagian dari program namun *compiler* mengabaikannya. Kita menuliskan *comment* untuk *programmer* lain yang membaca kode program kita atau sebagai pengingat untuk diri kita sendiri.

Terdapat dua cara menuliskan *comment* yaitu, *comment* satu baris dan *comment multi* baris.

Kita telah melihat cara menulis *comment* satu baris pada program pertama kita. *Comment* satu baris ditulis dengan diawali `//` seperti berikut:

```
// Ini adalah comment.
```

Semua teks setelah `//` akan diabaikan oleh *compiler*.

Comment multi baris ditulis dengan diawali dengan `/*` dan diakhiri dengan `*/`. Semua teks yang dituliskan setelah `/*` sampai dengan `*/` akan diabaikan oleh *compiler*. Berikut adalah contoh penulisan *comment multi* baris.

```
/*
    Comment multi baris dapat dituliskan
    Lebih dari satu baris.
*/
```

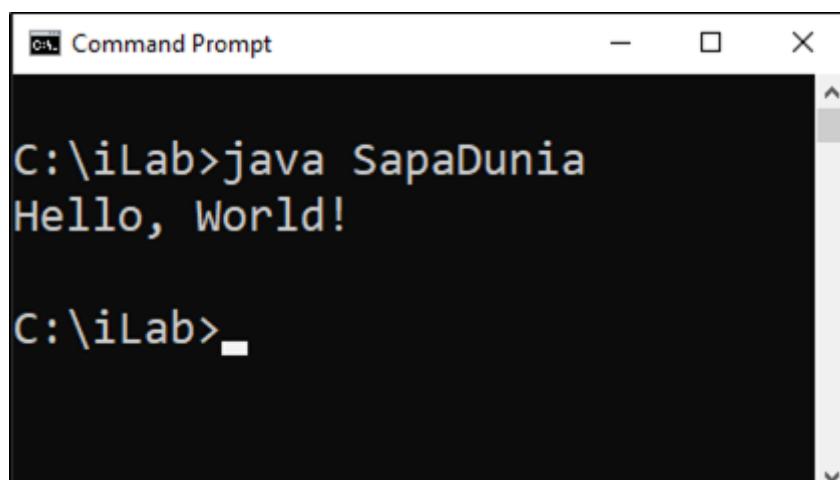
Program berikut mencontohkan penulisan *comment* pada program.

```
/*
    Program: Comment.java
    Program ini menampilkan pesan Selamat Datang
*/
public class Comment
{
    // Method main adalah titik awal mulai program.
    public static void main(String[] args)
    {
        System.out.println("Selamat Datang"); // Cetak selamat datang
    }
}
```

Comment satu baris dapat ditulis dimanapun dalam program, termasuk setelah penulisan *statement*. Pada baris 11 program di atas sebuah *comment* dituliskan setelah *statement*.

1.8 Method `print` dan Method `println`

Command prompt yang kita gunakan untuk mengkompilasi dan menjalankan program pertama kita disebut sebagai *console*. *Console* adalah antar muka untuk menjalankan program dan menampilkan *output* dari program berupa teks.



Program haruslah memberikan *output* sehingga pengguna program dapat mengetahui hasil dari program tersebut. Program contoh kita menggunakan *statement* berikut untuk mencetak *output* berupa pesan teks "Hello, world!" ke *console*.

```
System.out.println("Hello, world!");
```

Menampilkan pesan teks lain, kita dapat mengubah teks "Hello, world!" dengan teks yang kita inginkan. Misalkan, *statement* berikut:

```
System.out.println("Pemrograman sangat menyenangkan!");
```

Akan menampilkan pesan teks berikut ke *console*.

```
Pemrograman sangat menyenangkan
```

`System.out.println` adalah sebuah *method* yang telah ditulis sebelumnya oleh pengembang bahasa Java dan disertakan dalam JDK. *Method* adalah rangkaian *statement-statement* yang diberikan nama dan melakukan tugas spesifik tertentu. Untuk menggunakan *method* kita harus memanggilnya. Untuk memanggil *method* kita perlu menuliskan:

1. Nama *method* yang ingin dipanggil (dalam contoh ini, `System.out.println`).
2. Nilai-nilai yang diperlukan oleh *method* untuk melakukan tugasnya (dalam contoh ini kita memberikan nilai seperti "Hello, world" atau "Pemrograman sangat menyenangkan!"). Istilah teknis untuk nilai-nilai yang kita berikan ke *method* disebut sebagai **argument**. Kita menuliskan argumen di dalam tanda kurung.

Catatan. Tanda titik yang memisahkan nama-nama object dibaca sebagai "dot."

`System.out.println` dibaca "system dot out dot println".

Teks yang kita tuliskan dengan diapit tanda kutip ganda:

```
"Hello, world!"
```

atau

```
"Pemrograman sangat menyenangkan!"
```

Disebut sebagai **string**. *String* adalah istilah teknis dalam pemrograman untuk teks. Disebut *string* (untai) karena teks dapat dipandang sebagai untaian karakter-karakter.

terdapat dua *method* dasar untuk menampilkan pesan ke layar: `System.out.println` dan `System.out.print`.

Method `System.out.println`

Method `System.out.println` mencetak baris baru setelah mencetak teks. Sehingga teks berikutnya yang dicetak akan berada dalam baris baru. Sebagai contoh, lihat program berikut.

Program (`DuaBaris.java`)

```
/*
Ini contoh program sederhana Java yang mencetak
dua baris pesan ke console.

*/
public class DuaBaris
{
    public static void main(String[] args)
    {
        System.out.println("Pemrograman sangat menyenangkan!");
        System.out.println("Saya senang sekali mempelajarinya!");
    }
}
```

Output Program (`DuaBaris.java`)

```
Pemrograman sangat menyenangkan!
Saya senang sekali mempelajarinya!
```

Method `System.out.println` tanpa argument (dituliskan dengan mengosongkan teks dalam tanda kurung) digunakan untuk mencetak baris kosong. Berikut program yang mendemonstrasikannya.

Program (`TigaBaris.java`)

```
/*
Ini contoh program sederhana Java yang mencetak
tiga baris pesan yagn salah satunya baris kosong
ke console.

*/
public class TigaBaris
{
    public static void main(String[] args)
    {
        System.out.println("Pemrograman sangat menyenangkan!");
        System.out.println(); // Method println tanpa argument mencetak baris
kosong.
        System.out.println("Saya senang sekali mempelajarinya!");
    }
}
```

Output Program (`TigaBaris.java`)

```
Pemrograman sangat menyenangkan!
Saya senang sekali mempelajarinya!
```

Method `System.out.print`

Method `System.out.print` adalah method lain yang dapat juga digunakan untuk mencetak pesan ke layar. Perbedaan *method `System.out.print`* dengan *method `System.out.println`* adalah *method `System.out.print`* tidak mencetak baris baru. Program berikut mencontohkan penggunaan *method `System.out.print`*:

Program (*Menyenangkan.java*)

```
/*
 * Program ini mencontohkan penggunaan method print
 */
public class Menyenangkan
{
    public static void main(String[] args)
    {
        System.out.print("Pemrograman ");
        System.out.print("Java ");
        System.out.println("sangat menyenangkan!");
        System.out.println("Saya senang sekali mempelajarinya!");
    }
}
```

Output Program (*Menyenangkan.java*)

```
Pemrograman Java sangat menyenangkan!
Saya senang sekali mempelajarinya!
```

1.9 Error

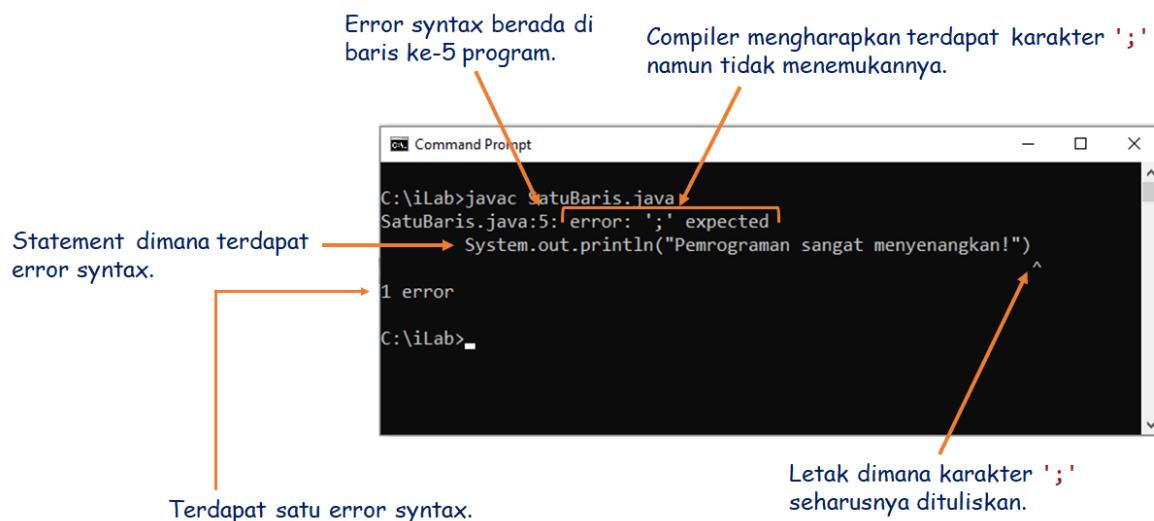
Seperti yang telah disebutkan sebelumnya, program harus ditulis mengikuti aturan-aturan penulisan yang disebut *syntax*. Kesalahan penulisan kode yang tidak mengikuti *syntax*, seperti kesalahan eja atau lupa menuliskan satu karakter seperti titik koma pada akhir *statement*, disebut dengan *error syntax*. *Error syntax* akan menyebabkan program tidak dapat dikompilasi. Namun, ketika *compiler* menemukan *error syntax* pada program kita, *compiler* akan memberitahukan kita dengan menampilkan pesan *error* yang berisi *error-error syntax* apa saja yang terdapat dalam program sehingga kita bisa memperbaiki *error-error syntax* tersebut. Sebagai contoh, misalkan pada program berikut, kita lupa menuliskan titik koma setelah akhir dari *statement `println`*.

```
public class SatuBaris
{
    public static void main(String[] args)
    {
        System.out.println("Pemrograman sangat menyenangkan!")
    }
}
```

Ketika kita menjalankan *compiler* untuk source code program di atas, *compiler* akan memberikan pesan *error* seperti berikut.

```
SatuBaris.java:5: error: ';' expected
    System.out.println("Pemrograman sangat menyenangkan!")
                                         ^
1 error
```

Pada bagian bawah pesan *error*, terdapat informasi mengenai banyaknya *error* yang ditemukan yaitu 1 *error*. Lalu, di baris pertama pesan, terdapat informasi mengenai, nama *source code* yang diikuti dengan informasi pada baris keberapa *error syntax* terjadi dan *error syntax* apa yang terdapat dalam baris tersebut. Angka 5 menunjukkan pada baris ke-5 program terdapat *error*, dan *error* tersebut adalah *compiler* mengharapkan karakter ; namun tidak dapat menemukannya. Pada baris selanjutnya, terdapat informasi *statement* mana yang menghasilkan *error* dan tanda ^ menunjukkan di mana karakter ; seharusnya dituliskan. Gambar berikut menjelaskan pesan *error compiler* ini.



Catatan. Bagaimana pesan *error syntax* ditampilkan bergantung pada versi JDK dan IDE yang digunakan. Namun, semua pesan *error syntax* akan berisi informasi pada baris keberapa *error syntax* terjadi dan *error syntax* apa yang terdapat dalam baris tersebut.

Ketika *compiler* menemukan sebuah *error syntax*, *compiler* tidak langsung berhenti namun tetap berusaha memeriksa *error-error syntax* lainnya. Semua *error-error syntax* yang berhasil ditemukan oleh *compiler* akan ditampilkan pada pesan *error compiler*.

Terkadang informasi pada pesan *error syntax* tidak begitu jelas memberitahukan *error syntax* apa yang terdapat dalam program kita. Misalkan, kita lupa menuliskan tanda kutip ganda pada awal dan akhir dari *string* seperti *statement* berikut:

```
System.out.println(Hello, World);
```

Compiler tidak akan memberikan pesan *error syntax* bahwa kita lupa menuliskan tanda kutip ganda, namun, *compiler* akan melaporkan *error syntax* ini sebagai "Cannot find symbol" yang berarti "Tidak dapat menemukan simbol". *Compiler* tidak begitu pintar untuk menyadari bahwa kita ingin menggunakan *string* dan kita lupa menuliskan tanda kurung ganda. Pada akhirnya, perbaikan *error syntax* ini kembali ke kita, apakah kita menyadari bahwa kita lupa menuliskan tanda kutip ganda.

Error-error syntax adalah hal yang lazim kita temui ketika menulis program. Bahkan *programmer* berpengalaman sekalipun, setiap kali *mengkompilasi* program yang baru ditulisnya pertama kali akan selalu mendapatkan error-error syntax ini.

Selain *error syntax*, terdapat jenis *error* lainnya yang disebut dengan ***error run-time***. *Error run-time* adalah *error* yang terjadi ketika program berjalan. Program dengan *error run-time* dapat dikompilasi namun ketika dijalankan program tersebut melakukan sesuatu tetapi tidak melakukan hal yang diharapkan. *Error run-time* disebabkan oleh kesalahan logika dalam program. Program dengan *error run-time* umumnya tidak memberikan pesan *error* namun menghasilkan *output* yang salah.

Beberapa jenis *error run-time* dapat menyebabkan program berhenti di tengah jalan (tidak menyelesaikan eksekusi semua *statement*) dan menghasilkan sebuah **eksepsi**. Eksepsi adalah pesan *error* dari Java Virtual Machine. Sebagai contoh, program berikut mempunyai *error run-time* yang menghasilkan eksepsi:

```
public class PembagianNol
{
    public static void main(String[] args)
    {
        System.out.println(1 / 0);
        System.out.println("Selesai");
    }
}
```

Pada baris ke-5, kita menuliskan *statement* `println` dengan *argument* `1 / 0`. Ketika statement ini dieksekusi, komputer melakukan operasi pembagian 1 dengan 0 yang tidak mungkin dilakukan sehingga komputer berhenti menjalankan program dan keluar dari program. Sebuah eksepsi yang memberitahukan jenis *error run-time* akan ditampilkan seperti berikut:

```
Exception in thread "main" java.lang.ArithmetricException: / by zero
at PembagianNol.main(PembagianNol.java:5)
```

Memperbaiki *error run-time* lebih sulit dibandingkan memperbaiki *error syntax*. *Compiler* akan mengkompilasi program dengan *error run-time* tanpa masalah namun program akan berjalan tidak sesuai dengan yang diharapkan. Oleh karena itu, umumnya *programmer* melakukan serangkaian pengujian program untuk memastikan program berjalan sesuai dengan yang diharapkan dan juga untuk mencari *error-error run-time* dalam programnya.

REFERENSI

- [1] Horstmann, Cay S. 2012. *Big Java: Late Objects, 1st Edition*. United States of America: John Wiley & Sons, Inc.
- [2] Gaddis, Tony. 2016. *Starting Out with Java: From Control Structures through Objects (6th Edition)*. Boston: Pearson.