# DAFTAR ISI

Contents

[1. DAFTAR ISI 1](#_Toc161481779)

[2. Kumpulan Project 7](#_Toc161481780)

[a. Java Fundamental 7](#_Toc161481781)

[3. CH1 TOP 1 - Digital Product Development 7](#_Toc161481782)

[b. Analogi 7](#_Toc161481783)

[c. FE / FrontEnd 7](#_Toc161481784)

[1) Contoh 8](#_Toc161481785)

[2) Framework 9](#_Toc161481786)

[d. BE / BackEnd 9](#_Toc161481787)

[3) Contoh 10](#_Toc161481788)

[e. Latihan – Tidak Ada 10](#_Toc161481789)

[f. Referensi 10](#_Toc161481790)

[4. Tujuan Belajar 10](#_Toc161481791)

[5. Instalasi Sebelum Belajar Java 10](#_Toc161481792)

[a. Instalasi JDK 11](#_Toc161481793)

[b. Instalasi Intellij IDEA 14](#_Toc161481794)

[1) install 14](#_Toc161481795)

[2) Fitur intellij 14](#_Toc161481796)

[c. Instalasi Eclipse 15](#_Toc161481797)

[1) url 15](#_Toc161481798)

[2) Buat folder di C 15](#_Toc161481799)

[3) Instalasi 16](#_Toc161481800)

[4) Run eclipse 16](#_Toc161481801)

[5) Add Lombok to Eclipse 17](#_Toc161481802)

[d. Instalasi PostgreSQL 19](#_Toc161481803)

[e. Instalasi Postman 23](#_Toc161481804)

[f. Apa itu Git ? 24](#_Toc161481805)

[1) Register Gitlab 25](#_Toc161481806)

[2) Login Gitlab 25](#_Toc161481807)

[3) Membuat project pada gitlab 25](#_Toc161481808)

[4) Menambahkan project pada gitlab anda 26](#_Toc161481809)

[5) Perintah melihat status perubahan file 31](#_Toc161481810)

[g. Instalasi Pencil 32](#_Toc161481811)

[h. Instalasi jasper report 32](#_Toc161481812)

[1) Download 32](#_Toc161481813)

[2) Klik dua kali file jasper report .exe 32](#_Toc161481814)

[3) Pilih lokasi instalasi 32](#_Toc161481815)

[4) Tunggu proses instalasi sampai selesai 33](#_Toc161481816)

[5) Finish 33](#_Toc161481817)

[i. Instalasi DBEaver 33](#_Toc161481818)

[j. Setup Koneksi PostgreSql dengan Dbever 38](#_Toc161481819)

[k. install Maven 39](#_Toc161481820)

[1) download pada bagian ini : 39](#_Toc161481821)

[2) daftarkan pada systemm env: 40](#_Toc161481822)

[3) Jika terjadi eror: silhkan tambahkan JAVA\_HOME 41](#_Toc161481823)

[6. CH1 TOP 2 - Java Introduction 41](#_Toc161481824)

[a. Situasi dunia kerja sebagai BackEnd Enginering 42](#_Toc161481825)

[1) soft skill 42](#_Toc161481826)

[2) hardskill 43](#_Toc161481827)

[b. Latihan Membuat Project java dengan intellij 43](#_Toc161481828)

[c. Hello word 43](#_Toc161481829)

[d. Deploy Gitlab 43](#_Toc161481830)

[e. Branch Hello Word 43](#_Toc161481831)

[7. CH1 TOP 3 - Data Types and Variables 43](#_Toc161481832)

[a. Materi 43](#_Toc161481833)

[b. Java program structure and Konsep Java Syntax dan Convention 44](#_Toc161481834)

[1) Referensi 44](#_Toc161481835)

[2) Pembahasan 44](#_Toc161481836)

[c. Variabel 48](#_Toc161481837)

[d. Type Data 48](#_Toc161481838)

[1) Tipe Data Primitif 49](#_Toc161481839)

[2) Tipe Data Non Primitif (Reference Tipe Data) 51](#_Toc161481840)

[3) Latihan 52](#_Toc161481841)

[e. Methods 52](#_Toc161481842)

[1) Cara Membuat Fungsi di Java 52](#_Toc161481843)

[2) Cara Memanggil/Eksekusi Fungsi 53](#_Toc161481844)

[3) Fungsi dengan Parameter 53](#_Toc161481845)

[4) Fungsi yang Mengembalikan Nilai 54](#_Toc161481846)

[5) Pemanggilan Fungsi di Fungsi Lain 55](#_Toc161481847)

[6) Fungsi Static dan Non-Static 55](#_Toc161481848)

[7) Latihan 56](#_Toc161481849)

[f. Basic Input-Output 56](#_Toc161481850)

[1) Scanner 57](#_Toc161481851)

[2) BufferReader 58](#_Toc161481852)

[3) Perbedaan 59](#_Toc161481853)

[4) Mengambil Input dengan Class Console 59](#_Toc161481854)

[5) Menampilkan Output 61](#_Toc161481855)

[6) Referensi 62](#_Toc161481856)

[7) Latihan 62](#_Toc161481857)

[g. Runtime Error & Compile Time Error 62](#_Toc161481858)

[1) Apa itu Compile Time Error ? 62](#_Toc161481859)

[2) Apa itu Runtime Error? 62](#_Toc161481860)

[3) Perbedaan Antara Compile Time Error dan Runtime Error : 63](#_Toc161481861)

[4) Referensi 63](#_Toc161481862)

[h. Debugging Error with IDE 63](#_Toc161481863)

[i. Latihan Aplikasi Kalkulator 64](#_Toc161481864)

[1) Pertanyaan Dasar:Type Data: 64](#_Toc161481865)

[2) Scanner Input Output 64](#_Toc161481866)

[3) Input dan Output: 64](#_Toc161481867)

[4) Metod untuk Kalkulasi: 65](#_Toc161481868)

[5) Contoh Output 65](#_Toc161481869)

[6) Code 66](#_Toc161481870)

[7) Branch 67](#_Toc161481871)

[8. Session: GIT 67](#_Toc161481872)

[a. Apa itu git dan jenisnya 67](#_Toc161481873)

[b. Gitlab 67](#_Toc161481874)

[1) Register 67](#_Toc161481875)

[2) Create projet in gitlab 67](#_Toc161481876)

[3) Get Personal Access Token and Remote Gitlab 67](#_Toc161481877)

[4) Push Project to gitlab 67](#_Toc161481878)

[5) Perintah Git 67](#_Toc161481879)

[c. Eror and Solusi 69](#_Toc161481880)

[1) master -> master (non-fast-forward) 69](#_Toc161481881)

[2) .gitignore 69](#_Toc161481882)

[d. Documentation Gitlab 69](#_Toc161481883)

[e. Setting git username 69](#_Toc161481884)

[f. Setting User Password GIT di Windows 69](#_Toc161481885)

[9. CH1 TOP 4 - Operator, Conditional \_ Looping 70](#_Toc161481886)

[a. Materi 70](#_Toc161481887)

[b. Tipe Reference Data pada Java 70](#_Toc161481888)

[c. Konsep dasar Array 70](#_Toc161481889)

[1) Array 70](#_Toc161481890)

[2) Deklarasi Array 1 Dimensi 70](#_Toc161481891)

[3) Array Multidimensi 71](#_Toc161481892)

[d. Math 72](#_Toc161481893)

[1) Referensi 72](#_Toc161481894)

[2) Apa itu Class Math? 72](#_Toc161481895)

[e. Equality 73](#_Toc161481896)

[1) Referensi 73](#_Toc161481897)

[2) Pembahasan 73](#_Toc161481898)

[f. Operators Operasi dasar matematis dan perbandingan 75](#_Toc161481899)

[1) Operator Artimatika 75](#_Toc161481900)

[2) Operator Relational 77](#_Toc161481901)

[3) Operator Logical 78](#_Toc161481902)

[4) Operator Assignment 80](#_Toc161481903)

[5) Operator Ternary 80](#_Toc161481904)

[6) Operator Bitwise 81](#_Toc161481905)

[g. Expression, Statement dan Block 85](#_Toc161481906)

[1) Ekspresi 85](#_Toc161481907)

[2) Statement 86](#_Toc161481908)

[3) Blok 86](#_Toc161481909)

[h. Struktur Percabangan 86](#_Toc161481910)

[1) Struktur if-else-if 87](#_Toc161481911)

[2) Struktur nested-if 87](#_Toc161481912)

[3) Struktur switch … Case ... 89](#_Toc161481913)

[i. Looping For, While, DO-While 89](#_Toc161481914)

[1) Looping with FOR 90](#_Toc161481915)

[2) Looping with While 91](#_Toc161481916)

[3) Looping do while 91](#_Toc161481917)

[4) Perintah Jump 92](#_Toc161481918)

[j. Recursive 94](#_Toc161481919)

[k. Latihan – Aplikasi Kalkulator 96](#_Toc161481920)

[1) Pertanyaan Dasar: 96](#_Toc161481921)

[2) Input dan Output: 96](#_Toc161481922)

[3) Tipe Data Referensi: 96](#_Toc161481923)

[4) Operasi Matematis 96](#_Toc161481924)

[5) Penggunaan If-Else. Switch case: 96](#_Toc161481925)

[6) Ternary Operator: 96](#_Toc161481926)

[7) Looping: while, for , do while 97](#_Toc161481927)

[8) Validasi Input: 97](#_Toc161481928)

[9) Cetak history transaksi pada file txt 97](#_Toc161481929)

[10) Melihat Total Hasil Operasi: 97](#_Toc161481930)

[11) Contoh Output 97](#_Toc161481931)

[12) Branch/gitlab 99](#_Toc161481932)

[10. CH1 TOP 5 - Programming Algorithm 99](#_Toc161481933)

[a. Materi 99](#_Toc161481934)

[11. CH1 TOP 6 - Java Standard Class 99](#_Toc161481935)

[a. Materi 99](#_Toc161481936)

[b. Perbedaan StringBuilder dengan StringBuffer 100](#_Toc161481937)

[c. StringBuffer 100](#_Toc161481938)

[1) Method 100](#_Toc161481939)

[d. StringBuilder 101](#_Toc161481940)

[1) Contoh 101](#_Toc161481941)

[e. String Joiner 102](#_Toc161481942)

[1) Contoh penerapan 102](#_Toc161481943)

[2) Implementasi dunia kerja 102](#_Toc161481944)

[f. Penerapan Stringbuilder dan StringBuffer dunia kerja 103](#_Toc161481945)

[g. Konsep Date, Calender, TimeZone, LocalDate 104](#_Toc161481946)

[h. Apa perbedaan Date and Calendar class, Timezone, localDate 104](#_Toc161481947)

[i. Contoh : Date, Calender, TimeZone, LocalDate 105](#_Toc161481948)

[j. Latihan Txt buku telpon 106](#_Toc161481949)

[k. Latihan txt dan csv buku telepon 109](#_Toc161481950)

[l. Branch 113](#_Toc161481951)

[m. Soal latihan 113](#_Toc161481952)

[12. Referensi 113](#_Toc161481953)

**JAVA**

# Kumpulan Project

## Java Fundamental

E:\binar\Binat Batch 6\MODUL\Project FINAL\java-fundamental

# CH1 TOP 1 - Digital Product Development

## Analogi



## FE / FrontEnd

front end developer adalah orang yang bertugas untuk menghubungkan sebuah situs atau aplikasi dengan pengguna.

Tugas dan tanggung jawab seorang Front-end Developer meliputi:

**Pengembangan Antarmuka Pengguna (UI):** Mereka merancang dan mengembangkan antarmuka pengguna yang menarik dan fungsional. Ini termasuk hal-hal seperti tata letak, elemen visual, ikon, warna, dan jenis huruf.

**Pengembangan Responsif:** Memastikan bahwa antarmuka dapat diakses dengan baik di berbagai perangkat dan ukuran layar, dari desktop hingga perangkat seluler.

**Pengembangan Interaksi Pengguna:** Membuat efek animasi, transisi, dan fitur interaktif seperti tombol, formulir, dan elemen-elemen lain yang memungkinkan pengguna berinteraksi dengan aplikasi atau situs.

**Optimasi Kinerja:** Memastikan bahwa halaman atau aplikasi berjalan dengan cepat dan efisien dengan mengoptimalkan gambar, kode, dan sumber daya lainnya.

**Kepatuhan Standar Web:** Mengikuti praktik terbaik dan standar web seperti HTML, CSS, dan JavaScript untuk memastikan konsistensi dan kinerja yang baik di berbagai platform.

**Pengujian dan Debugging:** Memastikan bahwa antarmuka dan fitur berfungsi dengan benar dengan melakukan pengujian dan debugging secara teratur.

**Kerjasama dengan Tim:** Bekerja sama dengan desainer UI/UX, Back-end Developers, dan anggota tim lainnya untuk memastikan koordinasi yang baik dalam pengembangan keseluruhan aplikasi.

**Tantangan Peramban:** Menangani perbedaan dalam cara berbagai peramban (browser) memproses dan menampilkan konten web.

**Pembaruan Teknologi:** Tetap mengikuti perkembangan terbaru dalam teknologi Front-end, bahasa pemrograman, framework, dan alat pengembangan.

**Keamanan:** Memahami praktik keamanan Front-end untuk menghindari ancaman seperti serangan lintas situs (XSS) dan serangan lintas situs (CSRF).

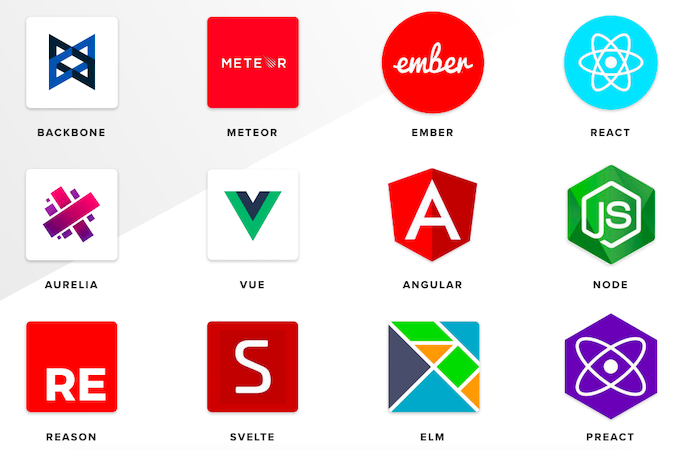
### Contoh

* WEB : tokopedia , shopee
* Mobile : BCA Mobile

A person and person looking at a computer

Description automatically generated

### Framework



## BE / BackEnd

Seorang Back-end Developer adalah seorang profesional dalam bidang pengembangan perangkat lunak yang fokus pada pengembangan dan pemeliharaan bagian belakang (back-end) dari sebuah aplikasi atau situs web. Back-end Developer bertanggung jawab untuk mengatur dan mengelola logika, data, dan fungsionalitas yang tidak langsung dilihat oleh pengguna akhir, tetapi sangat penting untuk operasi yang sukses dari sebuah aplikasi.

Tugas dan tanggung jawab seorang Back-end Developer meliputi:

Pengembangan Logika Aplikasi: Membangun komponen dan logika yang mengontrol perilaku aplikasi, termasuk alur kerja, pengelolaan data, pemrosesan, dan integrasi.

Pengelolaan Database: Mendesain, mengimplementasikan, dan mengelola struktur database dan operasi yang berkaitan dengan penyimpanan dan pengambilan data.

Pengembangan API (Application Programming Interface): Membangun API yang memungkinkan komunikasi antara berbagai bagian aplikasi dan sistem lain.

Keamanan: Memastikan sistem aman dari serangan dan ancaman dengan menerapkan praktik keamanan seperti autentikasi, otorisasi, enkripsi, dan pencegahan serangan.

Optimasi Kinerja: Memastikan bahwa bagian belakang berjalan dengan efisien dan memiliki waktu respons yang cepat.

Penanganan Skalabilitas: Mengelola dan memastikan bahwa aplikasi dapat beroperasi dengan baik saat menghadapi peningkatan lalu lintas atau pengguna.

Integrasi Sistem: Menghubungkan aplikasi dengan layanan eksternal, basis data, layanan cloud, dan komponen lainnya.

Pengujian dan Debugging: Memastikan bahwa logika dan fungsi aplikasi bekerja dengan benar melalui pengujian dan debugging.

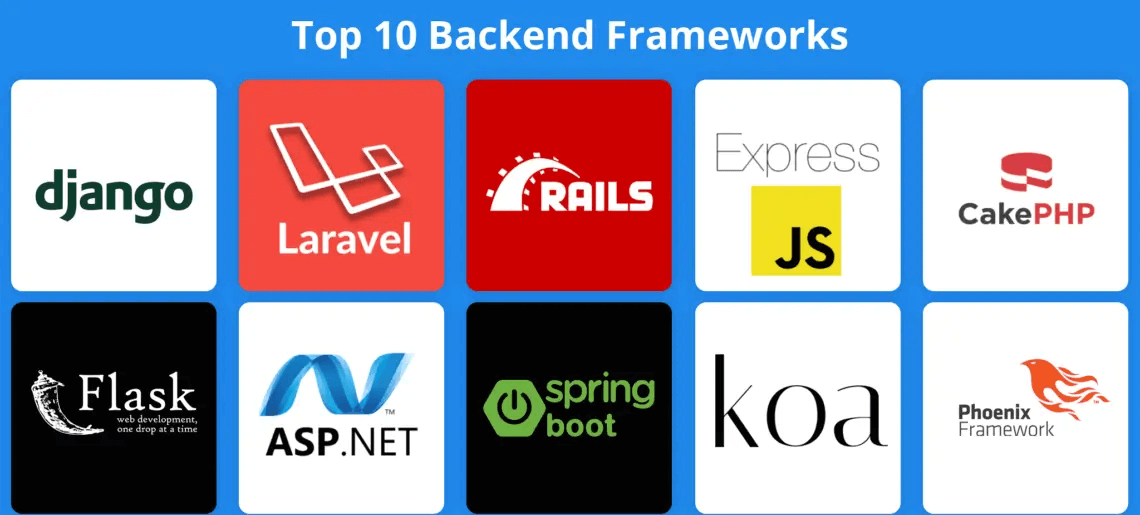
Kerjasama dengan Tim: Bekerja sama dengan Front-end Developers, desainer UI/UX, dan anggota tim lainnya untuk memastikan integrasi yang baik dalam pengembangan keseluruhan aplikasi.

Pemahaman tentang Teknologi Server-side: Menggunakan bahasa pemrograman server-side seperti Java, Python, Ruby, Node.js, dan sebagainya.

Manajemen Versi: Menggunakan alat manajemen versi seperti Git untuk mengelola kode sumber dan kolaborasi tim.

Back-end Developers umumnya memiliki pengetahuan mendalam tentang teknologi server-side, database, bahasa pemrograman server-side, serta konsep dan praktik keamanan. Mereka bekerja untuk memastikan bahwa bagian belakang aplikasi berjalan dengan baik, aman, dan dapat diandalkan.

### Contoh



## Latihan – Tidak Ada

## Referensi

https://www.binaracademy.com/blog/perbedaan-front-end-dan-back-end

# Tujuan Belajar

Membangun REST API dengan teknologi terbaik untuk mengontrol komunikasi data dengan client.

# Instalasi Sebelum Belajar Java

|  |  |
| --- | --- |
| 1. [JDK 1.8](https://www.oracle.com/id/java/technologies/javase/javase8-archive-downloads.html) 2. [Maven](https://fajar-isnandio.com/cara-instal-apache-maven-di-windows/) 3. [IntelliJ](https://www.nesabamedia.com/cara-install-intellij-idea-di-windows-10/) or [Download](https://www.jetbrains.com/idea/download/?section=windows) 4. [Git](https://www.petanikode.com/git-install/) or [Download](https://git-scm.com/downloads) 5. [Postgres](https://www.nesabamedia.com/cara-install-postgresql-di-windows-10/) or [Download Version 14.9](https://www.enterprisedb.com/downloads/postgres-postgresql-downloads) 6. [DBEaver](https://dbeaver.io/download/) 7. [Postman](https://www.postman.com/downloads/) 8. [Jasper Report](https://sourceforge.net/projects/jasperstudio/) |  |



## Instalasi JDK

Java JDK adalah sebuah perangkat lunak yang digunakan untuk melakukan proses kompilasi dari kode java ke bytecode yang dapat dimengerti dan dapat dijalankan oleh JRE (Java Runtime Envirotment).

JDK wajib terinstall pada komputer yang akan melakukan proses pembuatan aplikasi berbasis java, namun tidak wajib terinstall di komputer yang akan menjalankan aplikasi yang dibangun dengan java.

A diagram of a software development kit

Description automatically generatedJDK adalah sebuah paket aplikasi yang berisi JVM (Java Virtual Machine) + JRE (Java Runtime Environment) + berbagai aplikasi untuk proses pembuatan kode program Java.

Isi dari JDK – Java Development Kit (sumber gambar quora.com)

**Download** dan Lakukan **instalasi** pada laptop teman-teman :

<https://www.oracle.com/java/technologies/javase/jdk11-archive-downloads.html>

**Mengatur Path di System WIndows**

Dalam pengertian secara umum di dalam dunia komputer, path adalah istilah untuk menyebut alamat sebuah file di dalam directory komputer. Misalnya kita memiliki dokumen belajar\_java.doc. Alamat pathnya bisa saja tersimpan di C:\MyDocument\belajar\_java.doc.

Namun jika disebut “mengatur path di sistem operasi Windows“, itu maksudnya kita menginput sebuah alamat folder agar cmd Windows bisa menjalankan aplikasi yang tersimpan dari mana saja.

**Memeriksa apakah PATH Java sudah aktif atau belum**

* Buka CMD
* Ketikkan java –version

A screenshot of a computer program

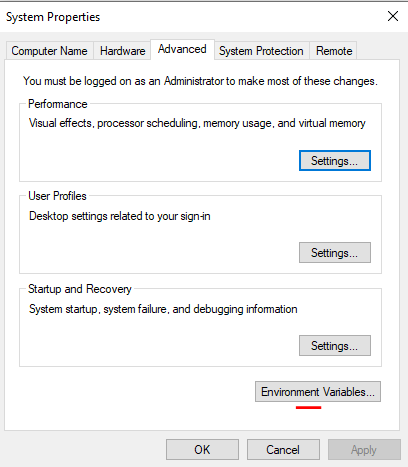
Description automatically generated

Gambar Version Java

Pada terminal diatas menunjukkan version java pada laptop teman-teman sudah terinstal. Jika terjadi eror, maka teman-teman perlu setting path java env di system windows

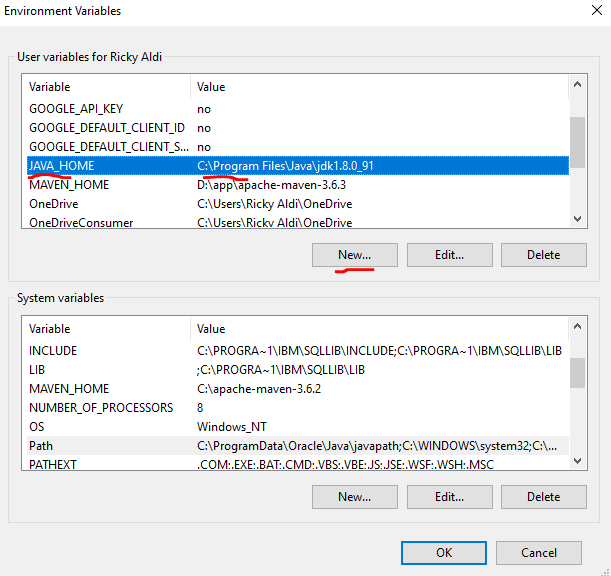
**Mengubah Pengaturan Path (environment variable) untuk Java JDK**

Ketikkan “**Advanced System Settings”** pada pencarian windows  
Pilih Environment Variabel



Gambar Advanced System Setting

Tambahkan Path JDK teman-teman pada user variable JAVA\_HOME pada gambar dibawah ini:



Gambar User Variabel

## Instalasi Intellij IDEA

***Berkenalan Dengan IntelliJ IDEA***

Intellij IDEA adalah sebuash IDE yang dikembangkan oleh Jetbrains. IntelliJ IDEA tersedia dalam 2 (dua) versi, yaitu versi Community Edition dengan lisensi *Apache 2 Licensed* yang bisa kita gunakan secara gratis dan versi Ultimate Edition untuk penggunaan komersial.

### install

**Silahkan download pada link berikut , dan pilih versi commmunity:**

<https://www.jetbrains.com/idea/download/#section=windows>

A screenshot of a computer

Description automatically generated

Gambar Intellij IDEA

### Fitur intellij

[Klik](https://www.dicoding.com/blog/berkenalan-dengan-intellij-idea/) or https://www.dicoding.com/blog/berkenalan-dengan-intellij-idea/

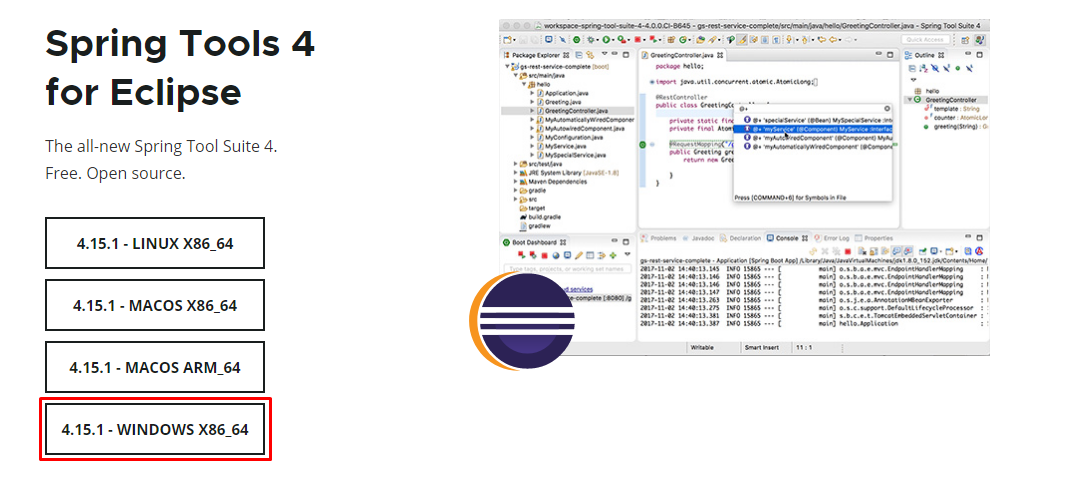
## Instalasi Eclipse

Ikuti langkah-langkah berikut

### url

<https://spring.io/tools/>

Pilih eclipse untuk OS windows



### Buat folder di C

Buatlah folder “Eclipse” di C, kemudian copi pasti file download di dalam folder tersebut.

A screenshot of a computer

Description automatically generated

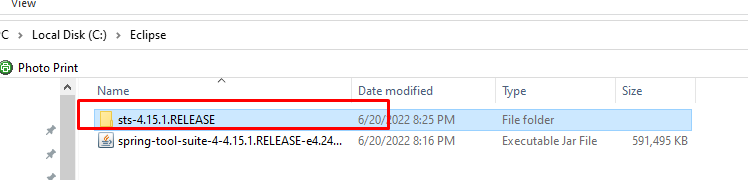
A screen shot of a computer

Description automatically generated

### Instalasi

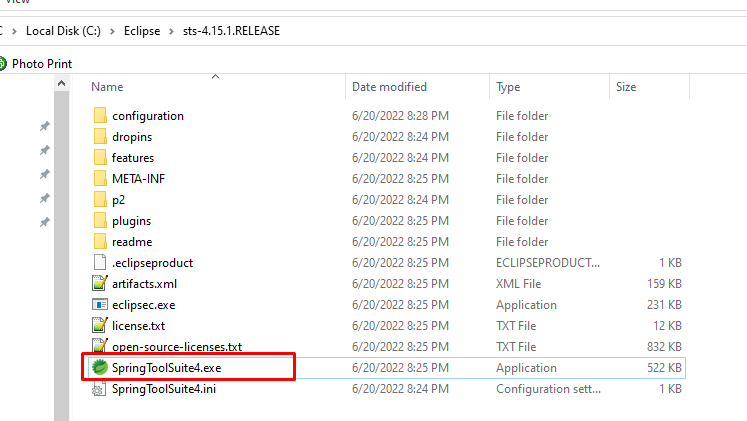
Cara install : klik 2 kali file tersebut. Kemudian lakukan perintah selanjutnya.

Sehingga akan membentuk sebuah folder “sts-4.15.1.RELEASE”, seperti gambar dibawah ini:



### Run eclipse

Agar dapat menggunakan eclipse, buka folder “sts-4.15.1.RELEASE”, kemudian klik 2 kali file “SpringToolSuite4.exe”, seperti gambar dibawah ini :



### Add Lombok to Eclipse

Klik kanan maven dependencies

Text

Description automatically generated with medium confidence

Pilih Lombok library

A screenshot of a computer

Description automatically generated with medium confidence

Klik kanan- pilih run application

A screenshot of a computer

Description automatically generated with medium confidence

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

**Penting: restart IDE , close dan running Kembali**

**Atau cara kedua:** java -jar Lombok.jar

<https://stackoverflow.com/questions/52780535/lombok-with-spring-tool-suite-4>

import ulang: klik control+spasi di @Data

## Instalasi PostgreSQL

**Apakah itu PostgreSQL ?**

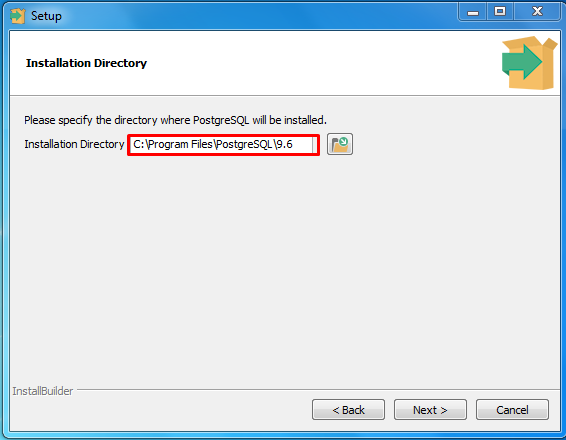
PostgreSQL adalah sistem database objek-relasional open source yang sangat kuat, dimana memiliki lebih dari 15 tahun fase pengembangan aktif dan arsitektur yang terbukti membuatnya mendapatkan reputasi yang baik untuk keandalan, integritas data, dll.

PostgreSQL mendukung empat bahasa prosedural standar, yang memungkinkan pengguna untuk menulis kode mereka sendiri dalam salah satu bahasa pemrograman dan itu dapat dieksekusi oleh server database PostgreSQL. Diantaranya adalah - Perl, Python, pgSQL. Selain itu, bahasa prosedural non-standar lainnya seperti PHP, V8, Ruby, Java, dll.

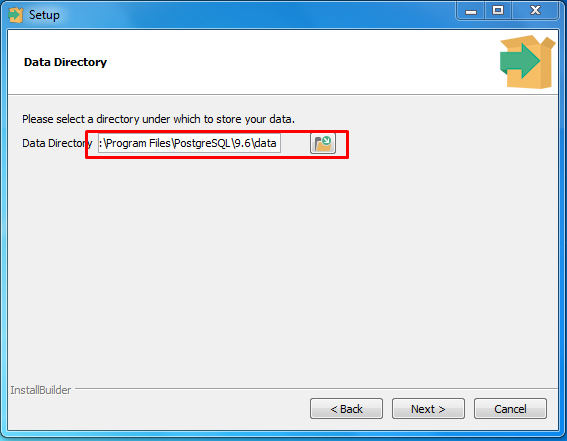
**Silahkan download dan install PostgreSQL pada link dibawah ini :**

<https://www.enterprisedb.com/downloads/postgres-postgresql-downloads>

**Cara Install PostgreSql Di Windows:**

* 1. Download Installer PostgreSql
  2. Lalu jalankan Installer seperti biasa.
  3. Pilih direktori installasi PostgreSql lalu klik **Next.**  
     [](https://cdn.tutorialpedia.net/wp-content/uploads/Screenshot_35.png)

#### Gambar : Instalasi PostgreSql 1

* 1. Pilih direktori Data. Klik Next**[](https://cdn.tutorialpedia.net/wp-content/uploads/Screenshot_36.png)**

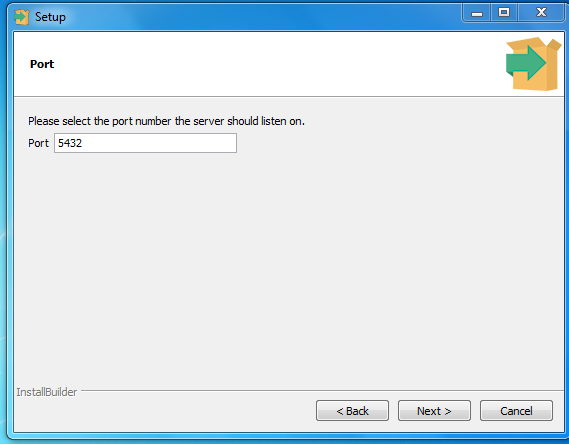
#### Gambar : Instalasi PostgreSql 2

* 1. Masukkan password yang nantinya digunakan untuk melakukan login ke server PostgreSql . lalu Klik **Next**[](https://cdn.tutorialpedia.net/wp-content/uploads/Screenshot_37.png)

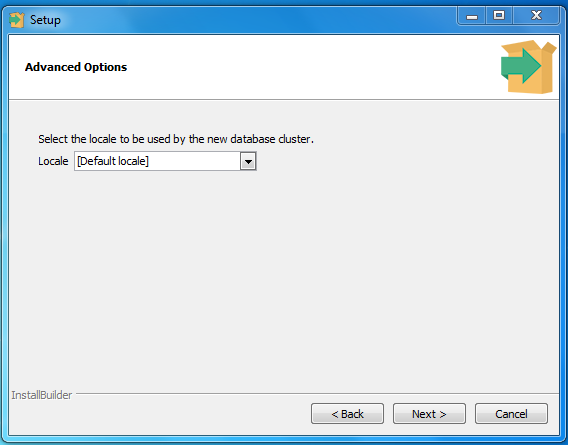
#### Gambar : Instalasi PostgreSql 3

* 1. Masukkan password yang nantinya digunakan untuk melakukan login ke server PostgreSql . lalu Klik **Next**[](https://cdn.tutorialpedia.net/wp-content/uploads/Screenshot_37.png)

#### Gambar : Instalasi PostgreSql 4

* 1. Pilih port PostgreSql yang nantinya akan digunakan. Port default biasanya menggunakan port 5432. Anda bisa mengubahnya sesuai selera apabila port tersebut sudah digunakan service lain. Klik **Next**[](https://cdn.tutorialpedia.net/wp-content/uploads/Screenshot_38.png)

#### Gambar : Instalasi PostgreSql 5

* 1. Pada tahap ini kita tidak perlu melakukan perubahan apa-apa, cukup klik **Next – Next** sampai selesai.  
     [](https://cdn.tutorialpedia.net/wp-content/uploads/Screenshot_39.png)

#### Gambar : Instalasi PostgreSql 6

* 1. Jika muncul **Stack Builder**abaikan saja dengan cara klik **Cancel**

Sekarang PostgreSql sudah berhasil terinstall pada komputer anda.

**Fitur-Fitur PostgreSQL**

* *View*
* *Transaction*
* Pilihan sub SQL
* Slony,
* Streaming replikasi
* DB mirror,
* SQL query kompleks

MVCC atau multiversion concurrency control, dan masih banyak lagi.

**Kelebihan PostgreSQL**

* **Terdapat fitur OO:**Mempunyai fitur OO berarti juga keunggulan dari PostgreSQL untuk dapat mendefinisikan tabel-tabel dan mewarisi table yang lain untuk dapat digunakan.
* **Memiliki Arstitektur Multiproses:** Arsitektur multiproses (forking) yang dimiliki oleh PostgreSQL menjadikan PostgreSQL mempunyai stabilitas yang tinggi.
* **Memiliki Kecepatan Meski Dalam Load Tinggi:** PostgreSQL didapuk memiliki kecepatan yang tinggi bahkan hingga mengalahkan kecepatan MySQL dalam hal [query](https://dosenit.com/php/fungsi-mysqli-query-php) dengan klausa [JOIN](https://dosenit.com/php/fungsi-string-join-pada-php) dengan tingkat yang kompleks
* **Memiliki Tipe Data Geometri:** PostgreSQL juga akan semakin memudahkan pengguna dikarenakan database system ini telah support berbagai tipe data geometri.
* **Menyediakan Seluruh Fitur Database:** PostgreSQL juga menyediakan bebagai macam fitur database yang pada umumnya juga terdapat pada produk-produk database komersial lain.

## Instalasi Postman

**Apa itu Postman ?**

**Postman** adalah sebuah aplikasi yang berfungsi sebagai REST Client untuk uji coba REST API. Postman biasa digunakan oleh developer pembuat API sebagai tools untuk menguji API yang telah mereka buat

**Postman** merupakan *tool* untuk melakukan proses *development* API, untuk saat ini sudah banyak fitur-fitur yang sangat membantu dalam proses *development* API, diantaranya :

* **Collection**

Pengelompokan *request* API yang bisa disimpan atau diatur dalam bentuk *folder*. Memudahkan untuk pengelompokan request sesuai dengan proyek yang di kerjakan.

* **Environment**

Semacam *config* untuk menyimpan *attribute dan attribute* tersebut dapat digunakan ataupun dimanipulasi dalam proses *request* API.

* **Response**

*Developer* dapat membuat Mockup API sebelum benar-benar mengimplementasikan ke dalam proyek.

**Silahkan** download dan install postman pada link dibawah ini :

<https://www.postman.com/downloads/> Instalasi Git

## Apa itu Git ?

* Git adalah salah satu sistem pengontrol versi *(Version Control System)* pada proyek perangkat lunak yang diciptakan oleh Linus Torvalds.
* Pengontrol versi bertugas mencatat setiap perubahan pada file proyek yang dikerjakan oleh banyak orang maupun sendiri.
* Git dikenal juga dengan *distributed revision control* (VCS terdistribusi), artinya penyimpanan database Git tidak hanya berada dalam satu tempat saja.

**Kenapa Git Penting Bagi Programmer?**

Beberapa menfaat yang akan kamu rasakan setelah bisa menggunakan Git.

* Bisa menyimpan seluruh versi source code;
* Bisa paham cara kolaborasi dalam proyek;
* Bisa ikut berkontribusi ke poryek open-source;
* Lebih aman digunakan untuk kolaborasi, karena kita bisa tahu apa yang diubah dan siapa yang mengubahnya.
* Bisa memahami cara *deploy* aplikasi modern;
* Bisa membuat blog dengan SSG.
* dan sebagainya…

**Apa itu Gitlab ?**

* Git adalah sistem versi kode sumber yang memungkinkan Anda melacak perubahan secara lokal dan mendorong atau menarik perubahan dari remote resources.
* GitLab, GitHub, dan Bitbucket : layanan yang menyediakan remote access  ke Git repositories. Selain meng-hosting kode Anda, layanan ini menyediakan fitur tambahan yang dirancang untuk membantu mengelola siklus pengembangan software. Fitur-fitur tambahan ini termasuk mengelola pembagian kode antara orang-orang yang berbeda, bug tracking, wiki space, dan alat-alat lain untuk ‘social coding’
* GitHub adalah layanan gratis yang tersedia untuk umum yang membutuhkan semua kode (kecuali Anda memiliki akun berbayar) dibuat terbuka. Siapa pun dapat melihat kode yang Anda push ke GitHub dan menawarkan saran untuk peningkatan. GitHub saat ini meng-host kode sumber untuk puluhan ribu proyek open source.

**Silahkan download an install Instalasi gitlab pada link berikut :**

<https://docs.gitlab.com/ee/topics/git/how_to_install_git/>

**Bagaimana Cek versi git ?**

Jika ingin mengetahui apakah git sudah terinstal, maka teman-teman dapat melakukan cek versi dengan cara sebagai beikut ini

* Buka terminal
* Ketikkan “git –version”

Sehingga akan menampilkan gambar berikut.

A screenshot of a computer program

Description automatically generated

Gambar Cek Versi Git

Berikut cara menambahkan file ke gitlab step-by-step:

### Register Gitlab

Teman-teman dapat register via link berikut : <https://gitlab.com/users/sign_up>

### Login Gitlab

Teman -teman dapat login pada link berikut : <https://gitlab.com/users/sign_in>

### Membuat project pada gitlab

* **Pilih new project**

A computer screen shot of a computer

Description automatically generated

Gambar new project

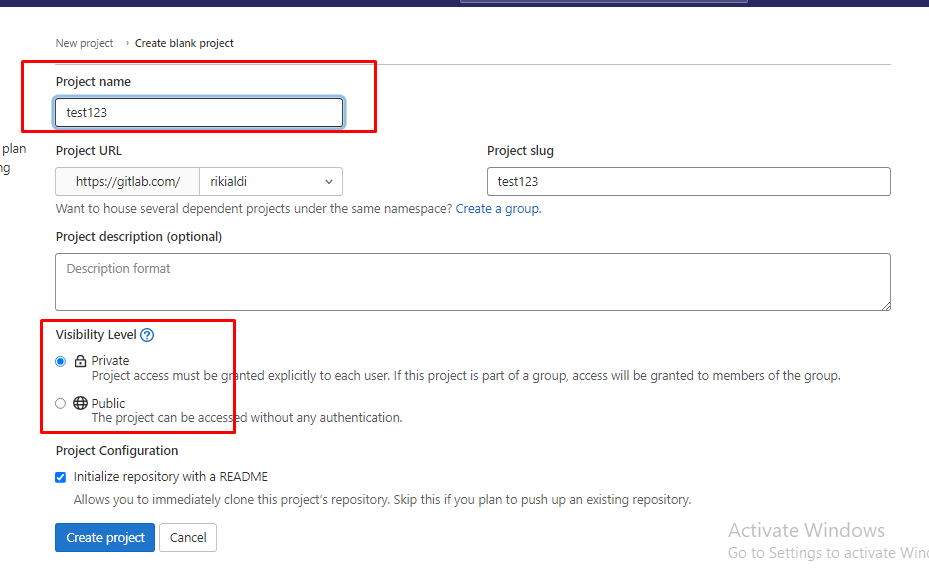
* **Create blank project**

A screenshot of a computer

Description automatically generated

Gambar create blank project

* **Isi nama project dan visibility level**



Gambar nama project dan visibility level

* **Project akan terbentuk**



Gitlab project

### Menambahkan project pada gitlab anda

* Membuat access token

Pilih preferences

A screenshot of a computer

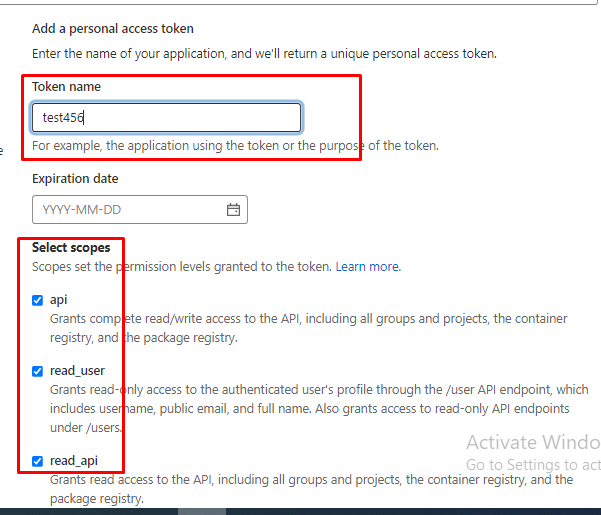
Description automatically generated

Pilih access token

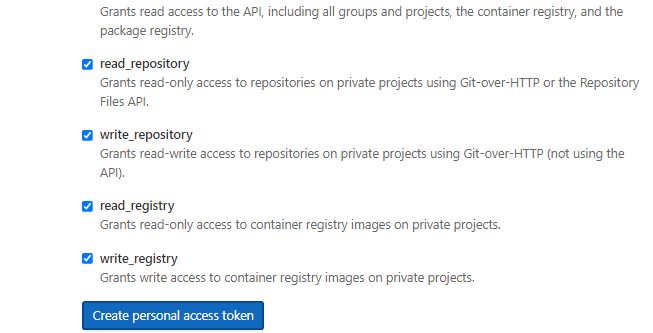
A screenshot of a computer

Description automatically generated

Isi token name dan select semua scopes



Create personal access token



Token sudah digenerate dan simpan token tersebut

A screenshot of a computer

Description automatically generated

* Membuat Project yang akan ditambahkan pada gitlab

Buatlah project di local anda, kemudian masuk pada terminal :

Ketikkan “cmd” pada menu filter windows anda

A screenshot of a computer

Description automatically generated

Sehingga tampilan di terminal seperti berikut

A screenshot of a computer

Description automatically generated

Add project ke gitlab

Jika project sudah ada di gitlab

git remote set-url origin [https://[user]:[token] @gitlab.com/[user]/](https://[user]:[token]%20@gitlab.com/%5buser%5d/)[namaproject]

**contoh**

git remote set-url origin [https://rikialdi:sdvnmdkfvndfvdfnvdfbdfb@gitlab.com/rikialdi/](https://rikialdi:[token%20anda%20masukkan%20disini]@gitlab.com/rikialdi/%5bnama%20project%20anda%5d)test123.git

Jika project belum ada di gitlab

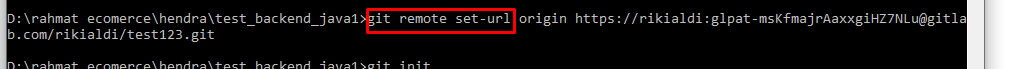
Step 1 : git init

Step 2: git remote add-url origin [https://[user]:[token] @gitlab.com/[user]/](https://[user]:[token]%20@gitlab.com/%5buser%5d/)[namaproject]

**contoh**

git remote set-url origin <https://rikialdi:sdvnmdkfvndfvdfnvdfbdfb@gitlab.com/rikialdi/test123.git>

misal project di gitlab sudah pernah ada, maka syntax nya seperti berikut:



Perintah **membuat branch baru** dengan nama test-satu:

“git checkout -b test-satu”

Perhatikan gambar dibawah ini :

A screen shot of a computer

Description automatically generated

Perintah **menambahkan** semua file :

“git add .”

Perhatikan gambar dibawah ini :

A black screen with white text

Description automatically generated

Perintah **commit** semua file :

“git commit -m “push project” ”

Perhatikan gambar dibawah ini :

A screen shot of a computer program

Description automatically generated

Perintah **push** semua file :

“git push -u origin test-satu”

Perhatikan gambar dibawah ini :

A screenshot of a computer screen

Description automatically generated

Project berhasil ditambahkan pada gitlab anda. Perhatikan gambar dibawah ini:

A screenshot of a computer

Description automatically generated

Perintah **clone(create branch baru dari branch yang ada) branch** test-satu menjadi test-dua:

“git checkout -b test-dua”

Perhatikan gambar dibawah ini :

A screen shot of a computer

Description automatically generated

Perintah **merge branch** . misal ingin merge branch test-dua menjadi test-satu.

Step 1 : kembali ke branch test-satu : git checkout test-satu

Step 2 : lakukan perintah merge : git merge test-dua

Perhatikan gambar berikut supaya lebih jelas

A screen shot of a computer program

Description automatically generated

### Perintah melihat status perubahan file

Perintah dengan “git status”. Perhatikan gambar dibawah ini :

A black screen with white text

Description automatically generated

Sekian pembahasan materi pada kesempatan ini, next materi kita akan membahas

* Variabel di java
* Type data di java
* Function di java
* Input & output di java

## Instalasi Pencil

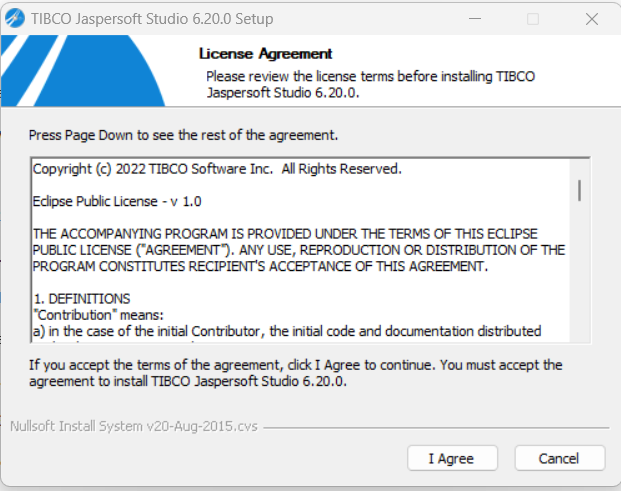
https://pencil.evolus.vn/Downloads.html

## Instalasi jasper report

### Download

https://community.jaspersoft.com/download

### Klik dua kali file jasper report .exe



### Pilih lokasi instalasi

A screenshot of a computer

Description automatically generated

### Tunggu proses instalasi sampai selesai

A screenshot of a software program

Description automatically generated

### Finish

A screenshot of a computer

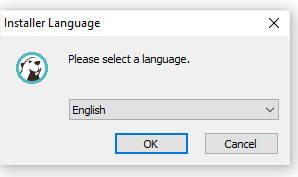
Description automatically generated

## Instalasi DBEaver

Anda dapat melakukan download DBever pada url berikut : <https://dbeaver.io/download/>.

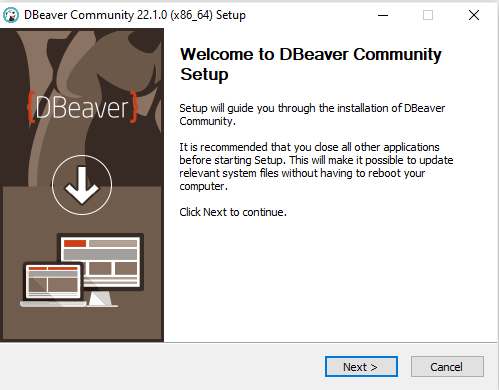
**Menginstall DBeaver**

Setelah proses pengunduhan selesai, selanjutnya kita akan menginstall DBeaver. Berhubung karena SQL Client ini open source, maka instalasinya sangat mudah. Pertama kita perlu membuka file unduhan DBeaver dengan klik kanan lalu open administrator. Pilih bahasa yang ingin digunakan, misal disini kita memilih english



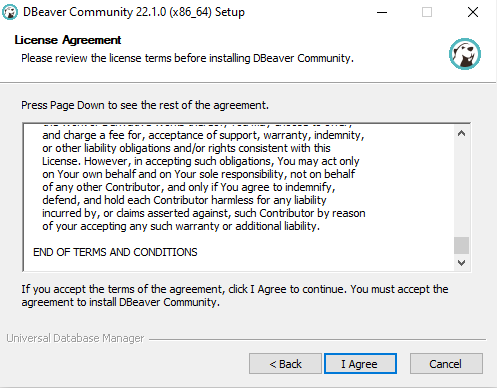
#### Gambar : Instalasi DBever 1

Setelah itu, pada bagian setup kita klik next saja.



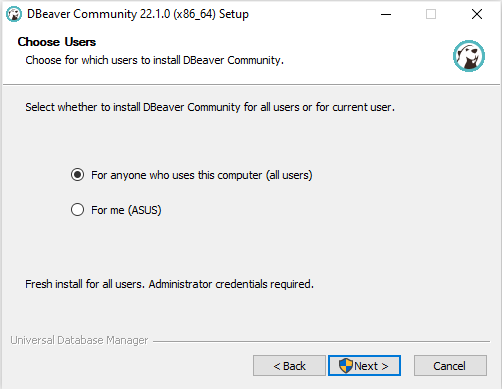
#### Gambar : Instalasi DBever 2

Bagian selanjutnya adalah mengenai lisensi, disarankan untuk membaca License Agreement yang disyaratkan oleh DBeaver. Jika setuju, klik I Agree.



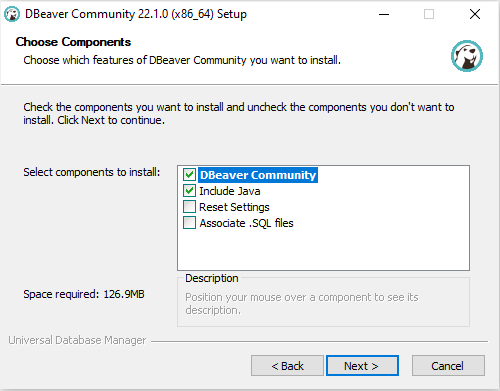
#### Gambar : Instalasi DBever 3

Pada pemilihan user, pilih œFor Anyone who uses this computer (all user).



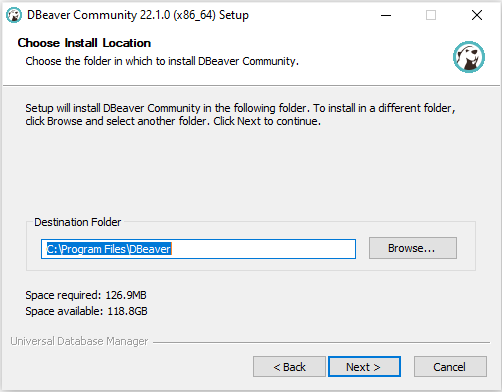
#### Gambar : Instalasi DBever 4

Pilih komponen yang ingin diinstall, silahkan dicentang sesuai kebutuhan. Disini akan dipilih komponen yang di centang secara default, yaitu DBeaver Community dan Include Java.



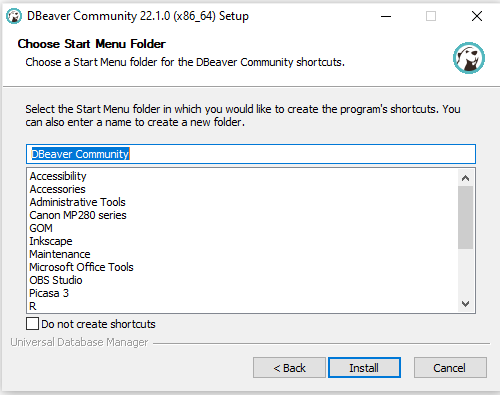
#### Gambar : Instalasi DBever 5

Langkah berikutnya memilih lokasi penginstalan. Sama seperti sebelumnya, lokasi penyimpanan kita pilih yang default saja.



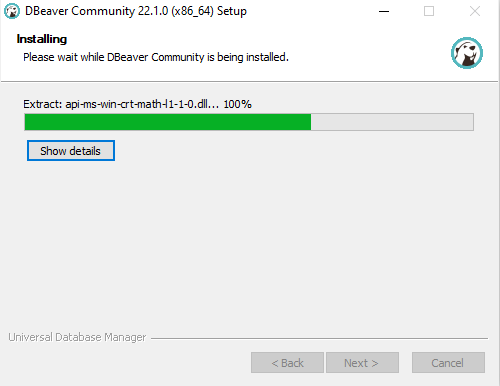
#### Gambar : Instalasi DBever 6

Berikutnya membuat membuat shortcut mode folder. Klik install untuk melanjutkan.



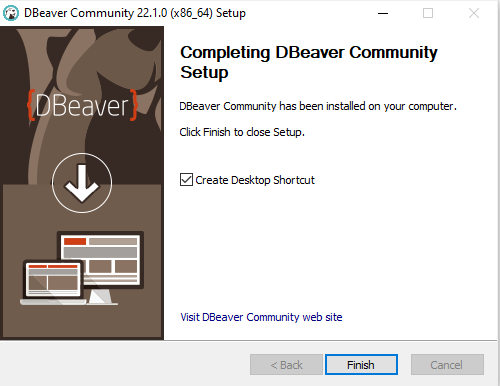
#### Gambar : Instalasi DBever 7

Tunggu hingga proses penginstalan selesai.



#### Gambar : Instalasi DBever 8

Setelah proses penginstalan selesai akan muncul pesan œDBeaver Community has been installed on Your Computer. Centang œCreate Desktop Shortcut untuk membuat shortcut.

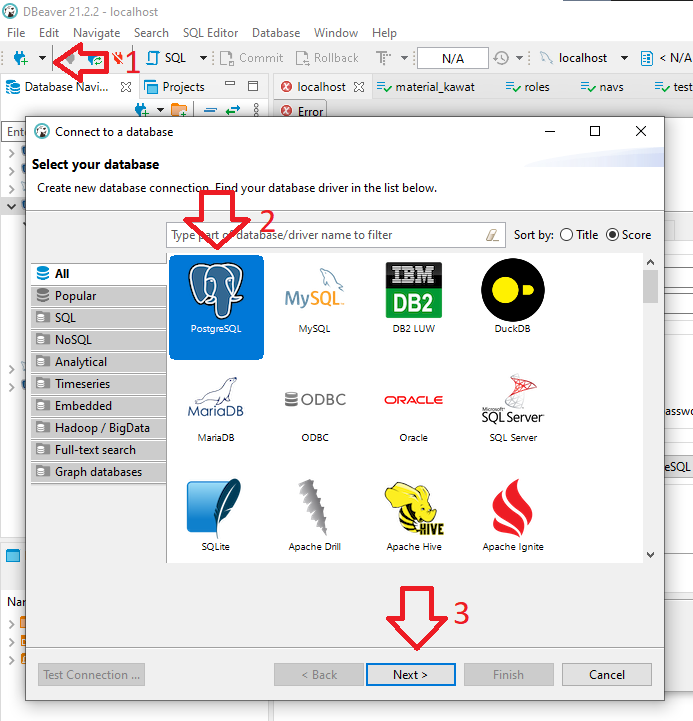


#### Gambar : Instalasi DBever 9

## Setup Koneksi PostgreSql dengan Dbever

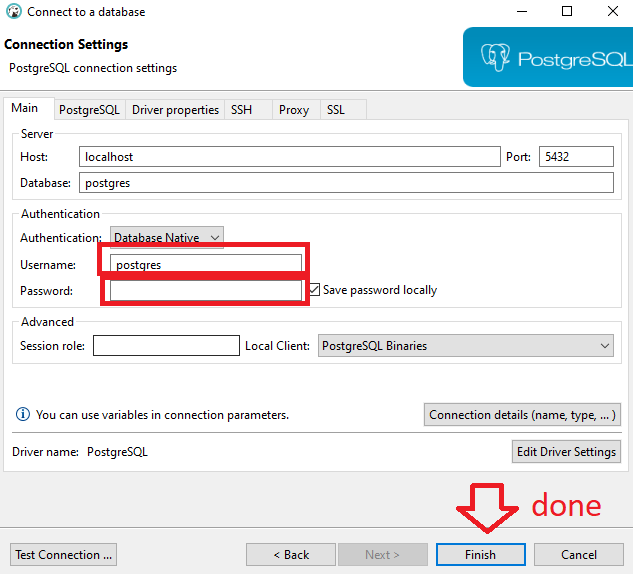
**Configuration Dbeaver dengan PostgreSql**

* 1. Buka Dbever
  2. Pilih icon koneksi kiri atas

****

#### Gambar : Koneksi DBever dengan PostgreSql 1

* 1. Kemudian masukkan username dan password PostgreSql

****

#### Gambar : Koneksi DBever dengan PostgreSql 2

* 1. Selesai..

## install Maven

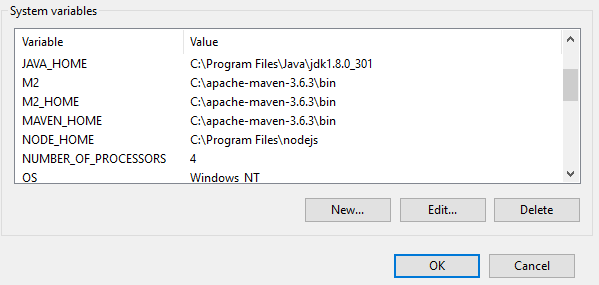
<https://maven.apache.org/download.cgi>

### download pada bagian ini :

Graphical user interface, application

Description automatically generated

### daftarkan pada systemm env:



Graphical user interface, text

Description automatically generated

### Jika terjadi eror: silhkan tambahkan JAVA\_HOME

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated

https://stackoverflow.com/questions/45119595/how-to-add-maven-to-the-path-variable

# CH1 TOP 2 - Java Introduction

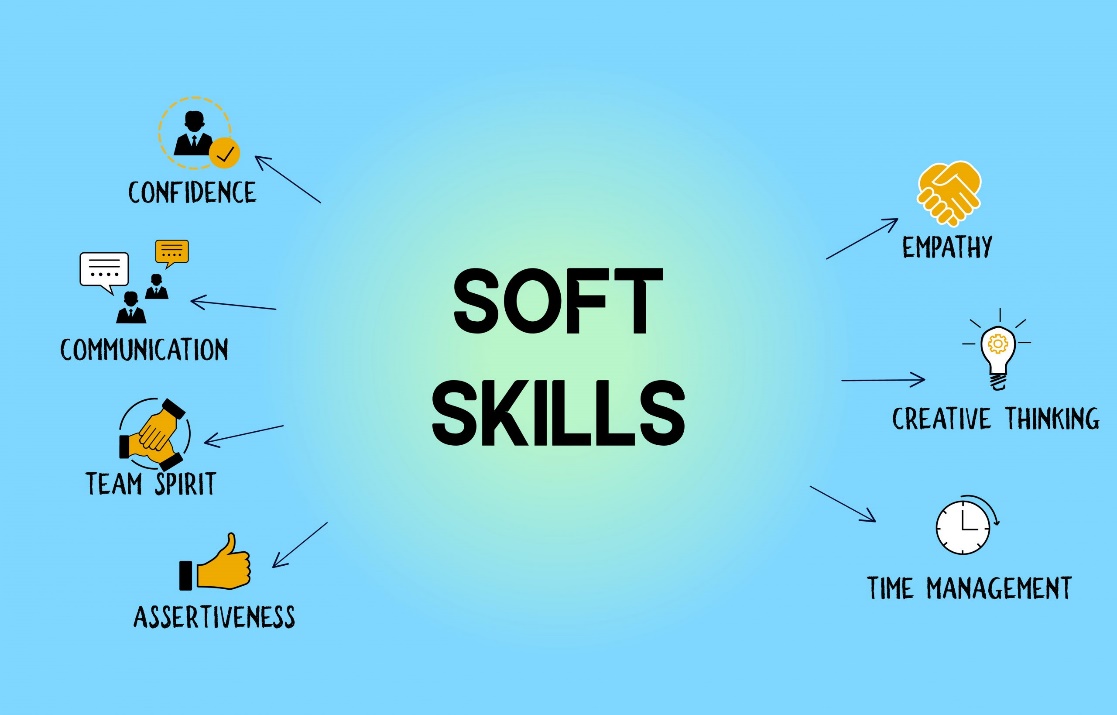
|  |  |
| --- | --- |
| 1.1.1 | [Pengenalan Back End](https://glints.com/id/lowongan/pekerjaan-back-end-developer/) |
| 1.1.2 | BE Engineer Skill Set and Day to Day Activities |
| 1.1.3 | [Getting Started to BE Java Tools](#_Instalasi_Sebelum_Belajar) |
| 1.1.4 | Java Introduction |
| 1.1.5 | [Java Ecosystem (JVM, JRE, JDK)](https://softscients.com/2020/11/12/perbedaan-dari-jdk-jre-jvm-pada-teknologi-java/) |
| 1.1.6 | [IDE dan Terminal](https://www.dicoding.com/blog/berkenalan-dengan-intellij-idea/) |
| 1.1.7 | [Tools Instalation (Java, Maven, IntelliJ, Gitlab)](#_Instalasi_Sebelum_Belajar) |
| 1.1.9 | [Hello World dengan Java](#_Hello_word) |
| 1.1.8 | [Apache Maven Basic](https://www.petanikode.com/java-maven/) |

1. [Mengenal Sejarah Java](https://glints.com/id/lowongan/bahasa-pemrograman-java/#mengenal-sejarah-java)
2. [Pengertian Bahasa Pemrograman Java](https://glints.com/id/lowongan/bahasa-pemrograman-java/#pengertian-bahasa-pemrograman-java)
3. [Mengapa Java Digunakan?](https://glints.com/id/lowongan/bahasa-pemrograman-java/#mengapa-java-digunakan)
4. [Penggunaan Bahasa Pemrograman Java](https://glints.com/id/lowongan/bahasa-pemrograman-java/#penggunaan-bahasa-pemrograman-java)
5. [Komponen Penting Java](https://glints.com/id/lowongan/bahasa-pemrograman-java/#komponen-penting-java)
6. [Contoh Java Framework](https://glints.com/id/lowongan/bahasa-pemrograman-java/#contoh-java-framework)

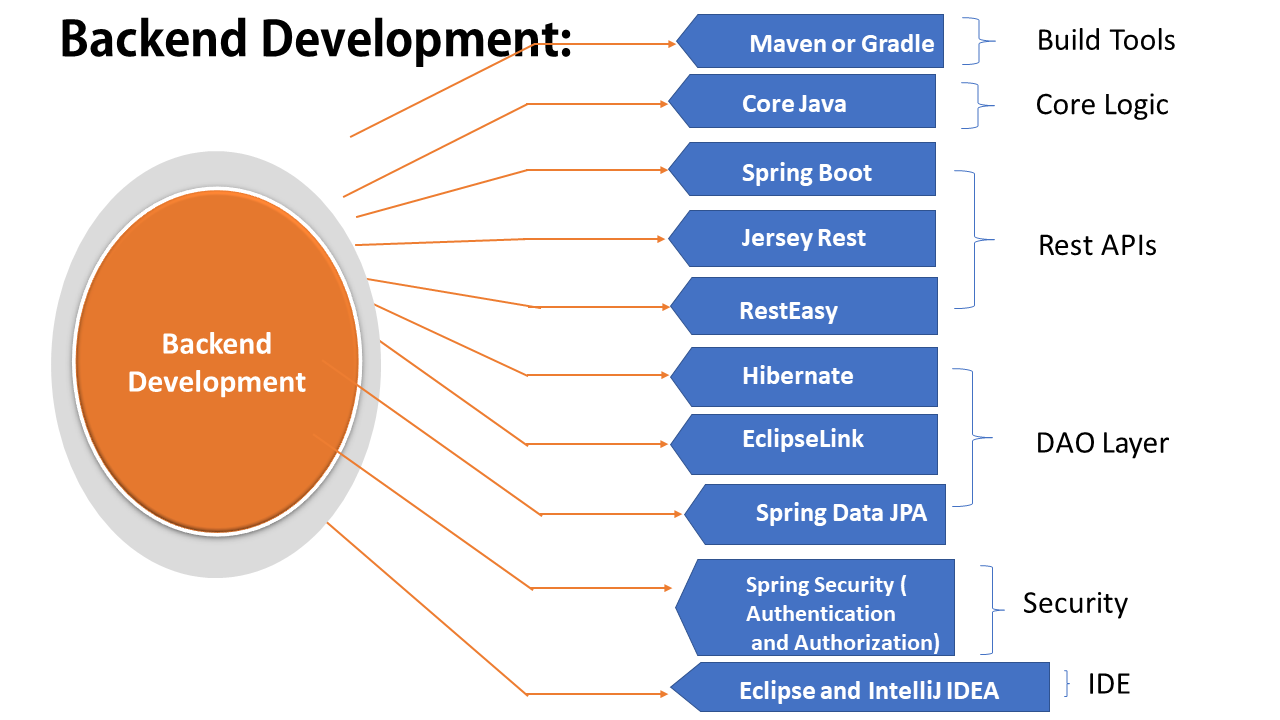
## Situasi dunia kerja sebagai BackEnd Enginering

<https://glints.com/id/lowongan/skill-software-developer/>

### soft skill



### hardskill



## Latihan Membuat Project java dengan intellij

## Hello word

[LINK](https://dutormasi.com/2019/10/membuat-program-hello-java-di-intellij-ide.html)

## [Deploy Gitlab](#_GIT)

[LINK](#_Get_Personal_Access)

## Branch Hello Word

[LINK](https://gitlab.com/rikialdi/java-fundamental-final/-/tree/sesi1-hello-word?ref_type=heads)

# CH1 TOP 3 - Data Types and Variables

## Materi

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1.2.1 | Java program structure | Asynchronous | 1. Mengidentifikasi struktur code pada program Java | **Day 2:** - Paparan materi - Hands on Git Clone dan Git pull dari code yang di demokan pada Day 1 - Hands on Hello World dengan berbagai variasi (dengan menggunakan variable, memanfaatkan berbagai tipe data, memanfaatkan method call, hingga penggunaan input) - Penugasan: Buat aplikasi prompting menu sederhana **- Hands ON : Define Type Data pada suatu variabel Sesuai dengan fungsinya**  **Challenge Progress:** - Ada baiknya fasil disini sudah memberikan sneak peek tentang challenge 1 kepada students - Menanyakan ke student progress pengerjaan challenge dan hambatan yang ditemukan |
| 1.2.2 | Konsep Java Syntax dan Convention | Synchronous | 2. Menjelaskan aturan-aturan syntax |
| 1.2.3 | Data Types | Asynchronous | 3. Menguraikan tipe data primitive dan non primitive pada Java |
| 1.2.4 | Variable | Synchronous | 4. Mengidentifikasi variable yang biasa dikenal di Java |
| 1.2.5 | Methods | Synchronous | 5. Membuat method dengan macam-macam tipe data |
| 1.2.6 | Basic Input-Output | Synchronous | 6. Melakukan input dan output data yang melingkupi scanner dan printout |
| 1.2.7 | Runtime Error & Compile Time Error | Synchronous | 7. Memahami perbedaan runtime error dan compile time error |
| 1.2.8 | Debugging Error with IDE | Synchronous | 8. Melakukan debugging eror melalui IDE |

## Java program structure and Konsep Java Syntax dan Convention

### Referensi

[Klik](https://www.petanikode.com/java-sintaks/) or https://www.petanikode.com/java-sintaks/

### Pembahasan

Setiap bahasa pemrograman memiliki struktur dan aturan penulisan sintaks yang berbeda-beda.

Java merupakan bahasa pemrograman yang dikembngkan dari bahasa C dan tentunya akan banyak mengikuti gaya penulisan C.

Saat baru pertama kali melihat program Java, mungkin kamu akan bertanya-tanya.

Ini apa? itu apa?

Contoh:

Coba perhatikan program berikut:

package com.petanikode.program;

*class* Program {

*public* *static* void main(String args[]){

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

}

}

Banyak hal yang kita belum ketahui.

Apa itu **package**?

Apa itu **class**?

…dan mengapa ditulis seperti itu?

Oleh sebab itu, kita perlu belajar sintaks dasar dan struktur program Java.

Mari kita mulai…

Struktur Dasar Program Java

Struktur program Java secara umum dibagi menjadi 4 bagian:

* Deklarasi Package
* Impor Library
* Bagian Class
* Method Main

Mari kita lihat contohnya:

package com.petanikode.program; //<- 1. deklarasi package

import java.io.File; //<- 2. Impor library

*class* Program { //<- 3. Bagian class

*public* *static* void main(String args[]){ //<- 4. Method main

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

}

}

Mari kita bahas, satu-per-satu…

1. Deklarasi Package

Package merupakan sebuah folder yang berisi sekumpulan program Java.

Deklarasi package biasanya dilakukan saat membuat program atau aplikasi besar.

Contoh deklarasi package:

package com.petanikode.program;

Biasanya nama package mengikuti nama domain dari sebuah vendor yang mengeluarkan program tersebut.

Pada contoh di atas, **com.petanikode** adalah nama domain dari petani kode.

Aturannya: nama domain dibalik, lalu diikuti nama programnya.

Bagaimana kalau kita tidak mendeklarasikan package?

Boleh-boleh saja dan programnya akan tetap bisa jalan.

Tapi nanti saat produksi, misalnya saat [membuat aplikasi Android](https://www.petanikode.com/topik/android/).

Kita wajib mendeklarasikan package.

2. Bagian Impor

Pada bagian ini, kita melakukan impor library yang dibutuhkan pada program.

Library merupakan sekumpulan class dan fungsi yang bisa kita gunakan dalam membuat program.

Contoh impor library:

import java.util.Scanner;

Pada contoh tersebut, kita mengimpor class **Scanner** dari package **java.util**.

3. Bagian Class

Java merupakan bahasa pemrograman yang menggunakan paradigma OOP (Object Oriented Programming).

Setiap program harus dibungkus di dalam class agar nanti bisa dibuat menjadi objek.

Kalau kamu belum paham apa itu OOP?

Cukup pahami class sebagai deklarasi nama program.

*class* NamaProgram {

*public* *static* void main(String args[]){

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

}

}

Ini adalah blok class.

Blok class dibuka dengan tanda kurung kurawal **{** kemudian ditutup atau diakhiri dengan **}**.

Di dalam blok class, kita dapat mengisinya dengan method atau fungsi-fungsi dan juga variabel.

Pada contoh di atas, terdapat method **main()**.

4. Method Main

Method **main()** atau fungsi **main()** merupakan blok program yang akan dieksekusi pertama kali.

Ini adalah entri point dari program.

Method **main()** wajib kita buat. Kalau tidak, maka programnya tidak akan bisa dieksekusi.

Contoh method **main()**.

*public* *static* void main(String args[]){

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

}

Penulisannya harus seperti ini…

Method **main()** memiliki parameter **args[]**. Parameter ini nanti akan menyimpan sebuah nilai dari argumen di command line.

Silahkan baca: [Apa Fungsi String[] args pada Pemrograman Java?](https://www.petanikode.com/java-args/)

Lalu di dalam method **main()**, terdapat statement atau fungsi:

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

Ini adalah fungsi untuk menampilkan teks ke layar monitor.

Statement dan Ekspresi pada Java

Statement dan eksrepsi adalah bagian terkecil dalam program. Setiap statement dan ekspresi di Java, harus diakhiri dengan titik koma (**;**).

Contoh statemen dan ekspresi:

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

System.out.println("Apa kabar?");

var x = 3;

var y = 8;

var z = x + y;

Statemen dan ekspresi akan menjadi instruksi yang akan dikerjakan oleh komputer.

Pada contoh di atas, kita menyuruh komputer untuk menampilkan teks **"Hello World"**, dan **"Apa kabar?"**.

Lalu kita menyuruhnya untuk menghitung nilai **x + y**.

Blok Program Java

Blok program merupakan kumpulan dari statement dan ekspresi yang dibungkus menjadi satu.

Blok program selalu dibuka dengan kurung kurawal **{** dan ditutup dengan **}**.

Contoh blok program:

// blok program main

*public* *static* void main(String args[]){

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

System.out.println("Hello Kode");

// blok program if

if( true ){

System.out.println('True');

}

// blok program for

for ( int i = 0; i<10; i++){

System.out.println("Perulangan ke"+i);

}

}

Intinya: jika kamu menemukan kurung **{** dan **}**, maka itu adalah sebuah blok program.

Blok program dapat juga berisi blok program yang lain (nested).

Pada contoh di atas, blok program **main()** berisi blok if dan for.

## Variabel

Variable merupakan tempat penyimpanan atau penampung nilai atau data di dalam memori. Variabel terdiri dari terdiri dari tipe data dan nama variabel. Tipe data menentukan jenis nilai atau data yang akan disimpan, sedangkan nama variabel menjadi pengenal (identifier), seperti halnya orang akan dipanggil dengan nama yang ia miliki, begitu-pun variabel.

**Deklarasi Variabel**

|  |
| --- |
| typedata variable = value; |

**Contoh program sbb :**

|  |
| --- |
| String namaVariabel = “riki”; |

## Type Data

Tipe data adalah jenis data yang tersimpan dalam variabel.

A diagram of data types

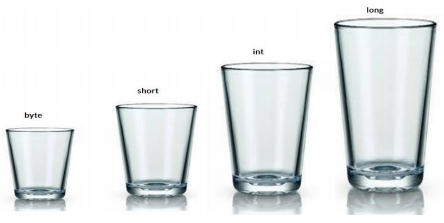
Description automatically generated

### Tipe Data Primitif

Tipe data primitif dapat digunakan untuk menyimpan satu nilai dalam satu waktu ke dalam sebuah variable. Tipe data primitif tersebut diantaranya sbb :

* byte
* short
* int
* long
* float
* double
* char
* boolean

byte, short, int and long merupakan tipe data primitif yang dikelompokkan kedalam tipe data integer, maksudnya byte, short, int and long variable dapat menampung data dalam bilangan integer, masing masing memiliki jangkauan yang berbeda. Seluruh tipe data integer dapat menampung bilangan integer positif dan negatif.



1. byte

Ukuran untuk tipe data byte ini adalah 8 bit. Jangkauan nilai ini adalah antara -128 sampai +127. Variable bertipe data byte di deklarasikan dengan keyword byte.

|  |
| --- |
| byte num1 = 120; |

1. Short

Ukuran untuk tipe data short ini adalah 16 bit. Jangkauan nilai ini adalah antara -32768 sampai +32767. Variable bertipe data short di deklarasikan dengan keyword short.

|  |
| --- |
| short num2 = 10000; |

1. Int

Ukuran untuk tipe data int ini adalah 32 bit. Jangkauan nilai ini adalah antara –2,147,483,648 to +2,147,483,647. Variable bertipe data int di deklarasikan dengan keyword int.

|  |
| --- |
| int num3 = 123; |

1. Long

Ukuran untuk tipe data long ini adalah 64 bit. Jangkauan dari tipe data ini cocok untuk mendeklarasikan nilai-nilai yang besar. Variable bertipe data int di deklarasikan dengan keyword long.

|  |
| --- |
| long num4 = 1234567891; |

Tabel Jangkauan Tipe Data Integer:

|  |  |  |
| --- | --- | --- |
| **Tipe Data** | **Ukuran (bit)** | **Range** |
| Byte | 8 | -128 s.d. 127 |
| Short | 16 | -32768 s.d. 32767 |
| Int | 32 | -2147483648 s.d. 2147483647 |
| long | 64 | -9223372036854775808  s.d. 9223372036854775807 |
|  |  |  |

Untuk nilai float dan double merupakan tipe data floating dengan kapasitas penyimpanan yang berbeda.

1. Float

Ukuran untuk tipe data float ini adalah 32 bit. Jangkauan nilai ini adalah antara –1.4e–045 sampai 3.4e+038. Variable bertipe data float di deklarasikan dengan keyword float. float merupakan single precision. Variables cocok untuk merepresentasikan nilai dollars dan nilai cents.

|  |
| --- |
| float num5 = 10000f; |

1. Double

Ukuran untuk tipe data double ini adalah 64 bit. Jangkauan nilai ini adalah antara 4.9e–324 sampai 1.8e+308. Variable bertipe data float di deklarasikan dengan keyword float. float merupakan single precision. Variables cocok untuk merepresentasikan nilai ke akuratan nya lebih tinggi.

|  |
| --- |
| double num6 = 0.122; |

Tabel Jangkauan Tipe Data floating point:

|  |  |  |
| --- | --- | --- |
| **Tipe Data** | **Ukuran (bit)** | **Range** |
| float | 32 | 1.4e-045 to 3.4e+038 |
| double | 64 | 4.9e-324 to 1.8e+308 |
|  |  |  |
|  |  |  |
|  |  |  |

1. Char

tipe data char ini dapat mengisi variabel dengan satu huruf dalam satu waktu, char di java berbeda dengan char pada bahasa c/c++. Pada Bahasa c/c++ ukurannya 8 bits sedangkan di java 16 bits dalam bentuk Unicode Range nya 0 sampai 65535 dan tidak ada negative char. Variable dengan tipe data ini di deklarasikan dengan keyword char.

|  |
| --- |
| char num7 = '1';  char num8 = '%'; |

1. Boolean

variable dengan tipe data boolean digunakan untuk menyimpan data logika. Dimana nilai yang disimpan berupa nilai true atau false. Tipe data boolean dideklarasikan dengan keyword boolean.

|  |
| --- |
| boolean num9 = false; |

### Tipe Data Non Primitif (Reference Tipe Data)

Pada tipe data non primitif dariabel selalu menyimpan data berupa alamat data. Selalu di sebut dengan tipe data reference karena merifer ke lokasi memori di mana data di simpan. Tipe data primitif ini diantaranya class, array, enum dll.

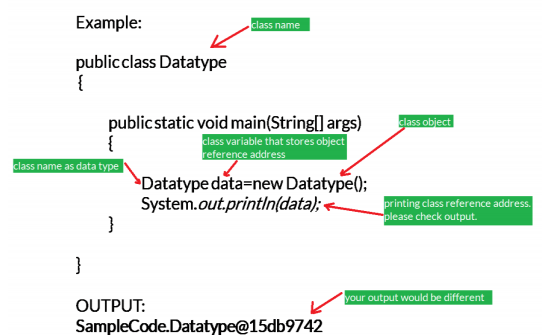
1. Class

Class adalah sebuah tipe data non primitif. Digunakan untuk menyimpan alamat rujukan dari sebuah objek. Tipe data class di deklarasikan menggunakan class name.

Class di bagi kedalam 2 bagian

* Class yang di defenisikan oleh user (UDF)
* Class yang sudah di siapkan olej jdk.

Contoh class yang di definisikan oleh user sbb :



Contoh class yang sudah ada dalam jdk :

Semua class yang sudah di sediakan oleh java, selalu berawalan huruf besar, jadi berbeda dengan tipe data primitif, dimana class tipe data memiliki method yang dapat dioperasikan untuk beberapa hal. Diantaranya sbb:

* Byte
* Short
* Integer
* Long
* Float
* Double
* Character
* Boolean
* String dll

### Latihan

[Refer Latihan dibawah](#_Pertanyaan_Dasar:Type_Data:)

## Methods

Method/fungsi pada bahasa pemrograman Java adalah kumpulan baris kode yang dikelompokkan untuk menjalankan tugas tertentu.

### Cara Membuat Fungsi di Java

Fungsi harus dibuat atau ditulis di dalam *class*.

Struktur dasarnya seperti ini:

|  |
| --- |
| *static* TypeDataKembalian namaFungsi(){  // statemen atau kode fungsi  } |

Penjelasan:

* Kata kunci static, artinya kita membuat fungsi yang dapat dipanggil tanpa harus membuat instansiasi objek.
* TypeDataKembalian adalah tipe data dari nilai yang dikembalikan setelah fungsi dieksekusi.
* namaFungsi() adalah nama fungsinya. Biasanya ditulis dengan huruf kecil di awalnya. Lalu, kalau terdapat lebih dari satu suku kata, huruf awal di kata kedua ditulis kapital.

**Contoh**

|  |
| --- |
| *static* void ucapSalam(){  System.out.println("Selamat Pagi");  } |

Tipe data void artinya kosong, fungsi tersebut tidak mengebalikan nilai apa-apa.

### Cara Memanggil/Eksekusi Fungsi

Setelah kita membuat fungsi, selanjutnya kita akan mengeksekusi fungsinya.

Fungsi dapat dipanggil dari fungsi main atau dari fungsi yang lainnya.

Contoh pemanggilan fungsi dalam dalam funsgi main:

|  |
| --- |
| *public* *static* void main(String[] args){  ucapSalam();  } |

Maka akan menghasilkan output:

|  |
| --- |
| Selamat Pagi |

### Fungsi dengan Parameter

Parameter adalah variabel yang menampung nilai untuk diproses di dalam fungsi. Parameter berperan sebagai input untuk fungsi.

Struktur dasarnya seperti ini:

|  |
| --- |
| *static* TipeData namaFungsi(TipeData namaParameter, TipeData namaParameterLain){  // kode fungsi  } |

**Penjelasan**:

* Parameter ditulis di antara tanda kurung (...);
* Parameter harus diberikan tipe data;
* Bila terdapat lebih dari satu parameter, maka dipisah dengan tanda koma.

Contoh fungsi yang memiliki parameter:

|  |
| --- |
| *static* void ucapin(String ucapan){  System.out.println(ucapan);  } |

Pada contoh tersebut, kita membuat parameter bernama ucapan dengan tipe String. Sehingga kita bisa menggunakan variabel ucapan di dalam fungsi.

Cara pemanggilan fungsi yang memiliki parameter:

|  |
| --- |
| *public* *static* void main(String[] args){  ucapin (“apa kabar”);  } |

### Fungsi yang Mengembalikan Nilai

Setelah fungsi memproses data yang diinputkan melalui parameter, selanjutnya fungsi harus mengembalikan nilai agar dapat diolah pada proses berikutnya.

Pengembalian nilai pada fungsi menggunakan kata kunci return.

Contoh:

|  |
| --- |
| *static* int luasPersegi(int sisi){  int luas = sisi \* sisi;  return luas;  } |

Pada contoh tersebut, kita membuat sebuah parameter bernama sisi. Kemudian fungsi akan mengembalikan nilai dengan tipe int (integer) dari variabel luas.

Contoh pemanggilanya:

|  |
| --- |
| System.out.println("Luas " + luasPersegi(5)); |

Hasil Output:

|  |
| --- |
| Luas adalah 25 |

### Pemanggilan Fungsi di Fungsi Lain

Fungsi-fungsi dapat saling memanggil untuk memproses data.

Contoh, sebuah program Kalkulator Bangun Ruang memiliki fungsi-fungsi: luasPersegi(), luasPersegiPanjang(), luasSegitiga(), luasBalok(), luasKubus() dsb.

Fungsi-fungsi tersebut dapat saling membantu, contoh fungsi luasKubus() membutuhkan fungsi luasPersegi().

Rumus:

Luas Kubus = 6 \* luasPersegi;

Luas Persegi = sisi \* sisi;

Maka programnya bisa dibuat seperti ini:

|  |
| --- |
| // membuat fungsi luasPersegi()  *static* int luasPersegi(int sisi){  return sisi \* sisi;  }  // membuat fungsi luasKubus()  *static* int luasKubus(int sisi){    // memanggil fungsi luasPersegi  return 6 \* luasPersegi(sisi);  } |

### Fungsi Static dan Non-Static

Pada contoh-contoh diatas, kita menggunakan kata kunci static sebelum membuat fungsi.

Kata kunci static akan membuat fungsi dapat dieksekusi langsung, tanpa harus membuat instansiasi objek dari class.

Contoh

|  |
| --- |
| // Fungsi non-static  void makan(String makanan){  System.out.println("Hi!");  System.out.println("Saya sedang makan " + makanan);  }    // fungsi static  *static* void minum(String minuman){  System.out.println("Saya sedang minum " + minuman);  } |

Cara pemanggilan

|  |
| --- |
| // fungsi main  *public* *static* void main(String[] args) {    // pemanggilan fungsi static  minum("Kopi");      // mambuat instansiasi objek saya dari class FungsiStatic  FungsiStatic saya = new FungsiStatic();  // pemanggilan fungsi non-static  saya.makan("Nasi Goreng");    } |

Apabila kita tidak membuat objek untuk memanggil fungsi *non-static*, maka akan terjadi error.

### Latihan

[Refer Latihan dibawah](#_Metod_untuk_Kalkulasi:)

## Basic Input-Output

Seperti yang kita ketahui, program komputer terdiri dari tiga komponen utama, yaitu: input, proses, dan output.

* Input: nilai yang kita masukan ke program
* Proses: langkah demi langkah yang dilakukan untuk mengelola input menjadi sesuatu yang berguna
* Output: hasil pengolahan

Java sendiri sudah menyediakan tiga class untuk mengambil input:

* Class Scanner;
* Class BufferReader;
* dan Class Console.

### Scanner

*Scanner* merupakan *class* yang menyediakan fungsi-fungsi untuk mengambil input dari keyboard.

Agar kita bisa menggunakan Scanner, kita perlu mengimpornya ke dalam kode:

import java.util.Scanner;

Contoh

|  |
| --- |
| // mengimpor Scanner ke program  import java.util.Scanner;  *public* *class* DataKaryawan {  *public* *static* void main(String[] args) {  // deklarasi variabel  String nama, alamat;  int usia, gaji;  // membuat scanner baru  Scanner keyboard = new Scanner(System.in);  // Tampilkan output ke user  System.out.println("### Pendataan Karyawan PT. XXX ###");  System.out.print("Nama karyawan: ");  // menggunakan scanner dan menyimpan apa yang diketik di variabel nama  nama = keyboard.nextLine();  // Tampilkan outpu lagi  System.out.print("Alamat: ");  // menggunakan scanner lagi  alamat = keyboard.nextLine();  System.out.print("Usia: ");  usia = keyboard.nextInt();  System.out.print("Gaji: ");  gaji = keyboard.nextInt();  // Menampilkan apa yang sudah simpan di variabel  System.out.println("--------------------");  System.out.println("Nama Karyawan: " + nama);  System.out.println("Alamat: " + alamat);  System.out.println("Usia: " + usia + " tahun");  System.out.println("Gaji: Rp " + gaji);  }  } |

**Point yang harus diperhatikan:**

penggunakan fungsi untuk mengambil data bergantung dari tipe data yang digunakan.

Misal, tipe datanya adalah String, maka fungsi atau method yang dipakai adalah nextLine().

Begitu juga dengan tipe data lain, Integer menggunakan nextInt(), Double menggunakan nextDouble(), dsb.

### BufferReader

Class BufferReader sebenarnya tidak hanya untuk mengambil input dari keyboard saja.

Class ini juga dapat digunakan untuk membaca input dari file dan jaringan.

Class ini terletak di dalam paket java.io.

Silahkan diimpor untuk dapat menggunakan class BufferReader.

import java.io.BufferedReader;

**Contoh**

|  |
| --- |
| *public* *static* void main(String[] args) *throws* IOException {  String nama;  // Membuat objek inputstream  InputStreamReader isr = new InputStreamReader(System.in);  // membuat objek bufferreader  BufferedReader br = new BufferedReader(isr);  // Mengisi varibel nama dengan Bufferreader  System.out.print("Inputkan nama: ");  nama = br.readLine();  // tampilkan output isi variabel nama  System.out.println("Nama kamu adalah " + nama);  } |

**Perlu diperhatikan**

Class BufferReader tidak bisa bekerja sendirian. Dia juga butuh teman yaitu: class InputStreamReader dan class IOException.

### Perbedaan

Perbedaan BufferReader dengan Scanner terlihat dari fungsi atau method yang dipakai.

Scanner menggunakan next(), sedangkan BufferReader  menggunakan readLine().

Lalu untuk tipe data integer, BufferReader  menggunakan fungsi read() saja.

### Mengambil Input dengan Class Console

Class Console hampir sama dengan BufferReader. Dia juga menggunakan fungsi readLine() untuk mengambil input.

Akan tetapi…

Class ini hanya bisa digunakan di lingkungan console saja, seperti Terminal dan CMD.

Untuk menggunakan class ini, kita perlu mengimpornya terlebih dahulu.

import java.io.Console;

**Contoh**

|  |
| --- |
| import java.io.Console;  *public* *class* InputConsole {  *public* *static* void main(String[] args) {  String nama;  int usia;  // membuat objek console  Console con = System.console();  // mengisi variabel nama dan usia dengan console  System.out.print("Inputkan nama: ");  nama = con.readLine();  System.out.print("Inputkan usia: ");  usia = Integer.parseInt(con.readLine());  // mengampilkan isi variabel nama dan usia  System.out.println("Nama kamu adalah: " + nama);  System.out.println("Saat ini berusia " + usia + " tahun");  }  } |

**Perhatikan**: dalam kode di atas, kita menggunakan fungsi Integer.parseInt(con.readLine()) untuk tipe data integer. Artinya, kita merubah tipe data String ke Integer.

Karena Console tidak memiliki nilai kembalian berupa integer untuk fungsi read().

Setelah itu, lakukan kompilasi melalui Terminal atau CMD. Buka terminal, kemudian masuk ke direktori tempat menyimpan kodenya.

Ketik perintah: javac InputConsole.java untuk melakukan kompilasi.

Setelah itu, ketik perintah java InputConsole untuk menjalankannya:

A screen shot of a computer

Description automatically generated

### Menampilkan Output

Kita sudah mengenal beberapa cara mengambil input dari keyboard untuk program berbasis teks.

Sekarang bagaimana dengan outputnya?

Ada beberapa fungsi yang sudah disediakan oleh Java:

* **Fungsi System.out.print()**
* **Fungsi System.out.println()**
* **Fungsi System.out.format()**

Apa saja perbedaan dari fungsi-fungsi tersebut?

* **Fungsi print() vs println()**

Fungsi print() dan println() sama-sama digunakan untuk menampilkan teks.

Lalu apa bedanya?

Fungsi print() akan menampilkan teks apa adanya. Sedangkan println() akan menampilkan teks dengan ditambah baris baru.

**Contoh**

|  |
| --- |
| *public* *static* void main(String[] args) {  System.out.print("ini teks yang dicetak dengan print()");  System.out.println("sedangkan ini teks yang dicetak dengan println()");  System.out.print("pake print() lagi");  } |

Output

A screen shot of a computer

Description automatically generated

### Referensi

https://www.petanikode.com/java-prosedur-dan-fungsi/

### Latihan

[Refer Latihan dibawah](#_Input_dan_Output:)

## Runtime Error & Compile Time Error

### Apa itu Compile Time Error ?

Error ini terjadi ketika pelanggaran aturan yang ada dalam sintaks. Compile Time Error menunjukkan sesuatu yang perlu kita perbaiki sebelum mengkompilasi kode. Kompiler dapat dengan mudah mendeteksi kesalahan ini. Itulah alasan mengapa kami menyebutnya kesalahan waktu kompilasi.

Berikut ini adalah kesalahan yang paling sering terjadi pada Compile Time Error:

* Terminator- hilang titik koma.
* Tanda kurung hilang.
* Mencetak nilai keseluruhan variabel tanpa deklarasi.

### Apa itu Runtime Error?

Runtime Error ini terjadi selama eksekusi program run-time setelah kompilasi berhasil. Kesalahan pembagian adalah salah satu kesalahan paling umum pada runtime error. Itu terjadi karena pembagian dengan nol. Sangat sulit bagi kompiler untuk menemukan kesalahan runtime karena tidak dapat menunjukkan garis yang tepat di mana kesalahan tersebut terjadi.

### Perbedaan Antara Compile Time Error dan Runtime Error :

| **Parameter** | **Compile Time Error** | **Runtime Error** |
| --- | --- | --- |
| Deteksi | Kompiler dapat dengan mudah mendeteksi kesalahan waktu kompilasi selama pengembangan kode. | Kompiler tidak dapat dengan mudah mendeteksi kesalahan runtime. Jadi, kita perlu mengidentifikasinya selama eksekusi kode. |
| Referensi | Kesalahan waktu kompilasi umumnya mengacu pada kesalahan yang sesuai dengan semantik atau sintaksis. | Kesalahan runtime mengacu pada kesalahan yang kami temui selama eksekusi kode selama runtime. |
| Fiksasi | Kami dapat dengan mudah memperbaiki kesalahan waktu kompilasi selama pengembangan kode. | Kompiler tidak dapat mengidentifikasi kesalahan runtime. Tapi kita bisa memperbaikinya setelah eksekusi kode dan identifikasi kode sebelumnya. |

### Referensi

https://dotedu.id/perbedaan-antara-compile-time-error-dan-runtime-error/

## Debugging Error with IDE

Cara Debugging with ide intellij , [KLIK](https://www.jetbrains.com/help/idea/debugging-your-first-java-application.html#analyzing-state) or https://www.jetbrains.com/help/idea/debugging-your-first-java-application.html#analyzing-state

1. [Apa itu Debug?](https://www.jagoanhosting.com/blog/debug-adalah/#apa-itu-debug)
2. [Perbedaan Testing dan Debugging](https://www.jagoanhosting.com/blog/debug-adalah/#perbedaan-testing-dan-debugging)
3. [Alasan Melakukan Debug](https://www.jagoanhosting.com/blog/debug-adalah/#alasan-melakukan-debug)
4. [Manfaat Debug](https://www.jagoanhosting.com/blog/debug-adalah/#manfaat-debug)
   1. [1. Meningkatkan keamanan sistem](https://www.jagoanhosting.com/blog/debug-adalah/#1-meningkatkan-keamanan-sistem)
   2. [2. Mengetahui fungsi yang tidak terpakai atau mengalami perubahan](https://www.jagoanhosting.com/blog/debug-adalah/#2-mengetahui-fungsi-yang-tidak-terpakai-atau-mengalami-perubahan)
   3. [3. Menyesuaikan CMS](https://www.jagoanhosting.com/blog/debug-adalah/#3-menyesuaikan-cms)
   4. [4. Mencegah terjadinya kesalahan dalam penulisan kode](https://www.jagoanhosting.com/blog/debug-adalah/#4-mencegah-terjadinya-kesalahan-dalam-penulisan-kode)
   5. [5. Menyelaraskan add on tambahan dengan sistem core terbaru](https://www.jagoanhosting.com/blog/debug-adalah/#5-menyelaraskan-add-on-tambahan-dengan-sistem-core-terbaru)
5. [Cara Kerja Debug](https://www.jagoanhosting.com/blog/debug-adalah/#cara-kerja-debug)
   1. [1. Memproduksi bug](https://www.jagoanhosting.com/blog/debug-adalah/#1-memproduksi-bug)
   2. [2. Mengidentifikasi error](https://www.jagoanhosting.com/blog/debug-adalah/#2-mengidentifikasi-error)
   3. [3. Menemukan lokasi bug](https://www.jagoanhosting.com/blog/debug-adalah/#3-menemukan-lokasi-bug)
   4. [4. Menganalisis](https://www.jagoanhosting.com/blog/debug-adalah/#4-menganalisis)
   5. [5. Membuktikan analysis error](https://www.jagoanhosting.com/blog/debug-adalah/#5-membuktikan-analysis-error)
   6. [6. Lakukan pada semua error](https://www.jagoanhosting.com/blog/debug-adalah/#6-lakukan-pada-semua-error)
   7. [7. Perbaikan dan validasi skrip](https://www.jagoanhosting.com/blog/debug-adalah/#7-perbaikan-dan-validasi-skrip)
6. [Tips Debug untuk Pemula](https://www.jagoanhosting.com/blog/debug-adalah/#tips-debug-untuk-pemula)
   1. [1. Aktifkan WP\_DEBUG](https://www.jagoanhosting.com/blog/debug-adalah/#1-aktifkan-wp_debug)
   2. [2. Aktifan WPDB Error Reporting](https://www.jagoanhosting.com/blog/debug-adalah/#2-aktifan-wpdb-error-reporting)
   3. [3. Cek Error Log](https://www.jagoanhosting.com/blog/debug-adalah/#3-cek-error-log)

## Latihan Aplikasi Kalkulator

### Pertanyaan Dasar:Type Data:

Buatlah sebuah program kalkulator sederhana menggunakan Java. Program ini harus mampu melakukan operasi penambahan, pengurangan, perkalian, dan pembagian. Pastikan Anda menggunakan variabel dan metode yang sesuai.

|  |
| --- |
| double angka1, angka2;  char operator; |

### Scanner Input Output

|  |
| --- |
| Scanner input = new Scanner(System.*in*);  double angka1, angka2;  char operator;   System.*out*.print("Masukkan angka pertama: ");  angka1 = input.nextDouble();  System.*out*.print("Masukkan angka kedua: ");  angka2 = input.nextDouble(); |

### Input dan Output:

Buat program Java yang meminta pengguna untuk memasukkan dua angka dan operator matematika (+, -, \*, /). Program tersebut harus menampilkan hasilnya.

|  |
| --- |
| Scanner input = new Scanner(System.*in*);  double angka1, angka2;  char operator;   System.*out*.print("Masukkan angka pertama: ");  angka1 = input.nextDouble();   System.*out*.print("Pilih operator (+, -, \*, /): ");  operator = input.next().charAt(0);   System.*out*.print("Masukkan angka kedua: ");  angka2 = input.nextDouble();   double hasil = *hitung*(angka1, angka2, operator);  System.*out*.println("Hasil: " + hasil); |

### Metod untuk Kalkulasi:

Buat metode terpisah dalam program Anda yang bertanggung jawab untuk melakukan perhitungan matematika. Metode ini harus menerima dua angka dan operator sebagai parameter dan mengembalikan hasil perhitungan.

|  |
| --- |
| public static double hitung(double angka1, double angka2, char operator) {  double hasil = 0.0;   switch (operator) {  case '+':  hasil = angka1 + angka2;  break;  case '-':  hasil = angka1 - angka2;  break;  case '\*':  hasil = angka1 \* angka2;  break;  case '/':  if (angka2 != 0) {  hasil = angka1 / angka2;  } else {  System.*out*.println("Error: Pembagian dengan nol tidak diizinkan.");  System.*exit*(1);  }  break;  default:  System.*out*.println("Error: Operator tidak valid.");  System.*exit*(1);  }   return hasil;  } |

### Contoh Output

A screen shot of a computer

Description automatically generated

### Code

|  |
| --- |
| package application.chapter1.sesi3;  import java.util.Scanner;  public class Kalkulator {  public static void main(String[] args) {  Scanner input = new Scanner(System.*in*);  double angka1, angka2;  char operator;   System.*out*.print("Masukkan angka pertama: ");  angka1 = input.nextDouble();   System.*out*.print("Pilih operator (+, -, \*, /): ");  operator = input.next().charAt(0);   System.*out*.print("Masukkan angka kedua: ");  angka2 = input.nextDouble();   double hasil = *hitung*(angka1, angka2, operator);  System.*out*.println("Hasil: " + hasil);  }   public static double hitung(double angka1, double angka2, char operator) {  double hasil = 0.0;   switch (operator) {  case '+':  hasil = angka1 + angka2;  break;  case '-':  hasil = angka1 - angka2;  break;  case '\*':  hasil = angka1 \* angka2;  break;  case '/':  if (angka2 != 0) {  hasil = angka1 / angka2;  } else {  System.*out*.println("Error: Pembagian dengan nol tidak diizinkan.");  System.*exit*(1);  }  break;  default:  System.*out*.println("Error: Operator tidak valid.");  System.*exit*(1);  }   return hasil;  } } |

### Branch

<Klik> or https://gitlab.com/rikialdi/java-fundamental-final/-/tree/sesi2-type-data-kalkulator?ref\_type=heads

# Session: GIT

## Apa itu git dan jenisnya

[LINK](https://www.dewaweb.com/blog/mengenal-git/)

## Gitlab

### Register

[LINK](https://gitlab.com/)

### Create projet in gitlab

[LINK](cara%20membuat%20project%20di%20gitlab)

### Get Personal Access Token and Remote Gitlab

[LINK](https://docs.gitlab.com/ee/user/profile/personal_access_tokens.html) atau [LINK](https://medium.com/@stanleywijaya/tutorial-upload-file-ke-gitlab-memakai-access-token-lewat-cmd-f72ca7593966)

### Push Project to gitlab

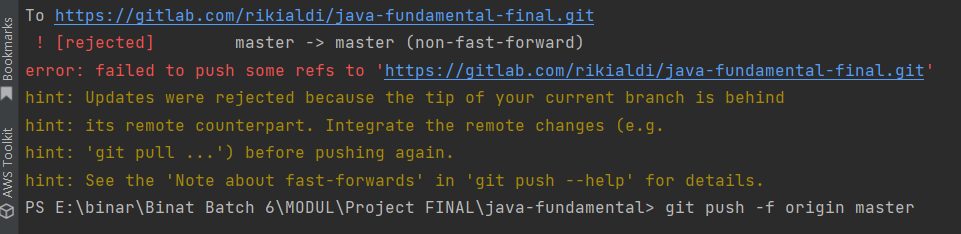
|  |
| --- |
| 1. git init 2. git remote add origin <https://rikialdi:tokenanda@gitlab.com/rikialdi/java-fundamental-final.git>   //jika sudah terdaftar  git remote set-url origin [https://rikialdi: tokenanda @gitlab.com/rikialdi/ java-fundamental-final.git](https://rikialdi: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@github.com/rikialdi/test11.git)  git remote add origin [https://rikialdi:[namatoken]@gitlab.com/nama\_user/[namarepository].git](https://rikialdi:[namatoken]@github.com/nama_user/%5bnamarepository%5d.git)   1. git add . 2. git commit -m “deksripsi commit” 3. git push -u origin master |

### Perintah Git

|  |  |
| --- | --- |
| git reset --hard | Urungkan perubahan/ menghapus /membatalkan semua perubahan di branch local. Perubahan akan kembali sebelumnya. Hati2 menggunakan ini, |
| Git diff | Melihat perubahan |
| Git status | Melihat file yang berubah |
| Git checkout nambranch | Pindah branch |
| Git checkout **-b** newBranch | Membuat branch baru dari branch sebelumnya, misal nya branch master |
|  |  |
|  |  |
|  |  |

## Eror and Solusi

### master -> master (non-fast-forward)



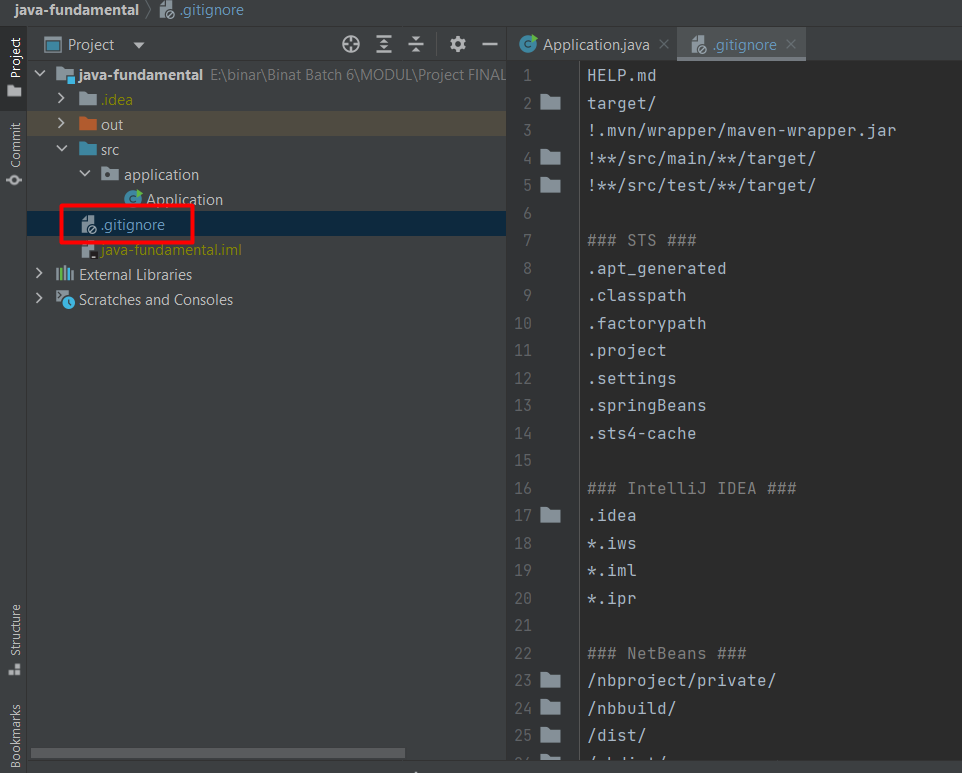
Solusi : git push -f origin master

Outputnya : akan melakukan perubahan dari local ke gitlab, ini perlu diketahui: warning, perubahan yang ada di gitlab akan kehapus semuanya dan akan di replace dari file local

### .gitignore

File ini penting sekali, bertujuan melakukan pengecualian push file ke gitlab.

Contoh : tidak melakuka npush pada file .idea



## Documentation Gitlab

[LINK](https://docs.gitlab.com/ee/)

## Setting git username

|  |
| --- |
| https://docs.github.com/en/get-started/getting-started-with-git/setting-your-username-in-git |

## Setting User Password GIT di Windows

URL : https://linuxhint.com/change-git-password-in-windows/

# CH1 TOP 4 - Operator, Conditional \_ Looping

## Materi

* Tipe Reference Data pada Java
* Konsep dasar Array
* Operasi dasar matematis dan perbandingan
* Klasifikasi Expression, Statement dan Block
* Penggunaan If else dan Ternary operation
* Logic flow dari Conditional
* Logic flow dari Looping

## Tipe Reference Data pada Java

[Refer Section](#_Type_Data)

## Konsep dasar Array

### Array

Array juga termasuk tipe data non primitif. Array merupakan sebuah solusi untuk mendeklarasikan sejumlah Variabel secara tepat. Pemakaian Variabel Array akan menghemat waktu penyebutan nama Variabel. Variabel Array adalah sejumlah Variabel dengan nama yang sama.

Ilustrasi Array:

A diagram of a number

Description automatically generated with medium confidence

Catatan: No. Indeks/No. Element di awali angka 0 (nol) dan diakhiri n-1 dimana n adalah panjang array.

### Deklarasi Array 1 Dimensi

Beberapa cara mendeklarasikan sebua variabel array, yaitu:

**Deklarasi**

|  |
| --- |
| TypeData[] namaArray; contoh int[] angka; |

**Deklarasi dan Inisialisasi**

|  |
| --- |
| typeData[]namaarray = new typedata[jumlah elemen]  Contoh  int[] angka1 = new int[5]; |

**Deklarasi otomatis**

|  |
| --- |
| typeData[]namaarray = {daftar elemen}  Contoh  int[] angka1 = {1,2,3,4,5,6}; |

**Contoh program array 1 dimensi**

|  |
| --- |
| public static void main (String args[])  {  //Deklarasi array  String[] mahasiswa = {"Rini","Aldi","Acep","Putra"};  obj : {  “id” : 1, “data1” :[{  “id”:””,  “nama”:”nama”}]  }  //Menampilkan isi array pada indeks ke-2  System.out.println(mahasiswa[2]);  } |

### Array Multidimensi

Selain berupa sederetan variabel satu dimensi, kita dapat pula membuat array yang berukuran lebih dari satu dimensi atau disebut array multi-dimensi. Pada bagian ini kita mencoba mencontohkan bentuk array dua dimensi sbb:  
  
**typedata[][] namarray = new typedata [jumlah baris][jumlah kolum]**

**Contoh**

String[][] meja = new String[2][3];

Contoh program

|  |
| --- |
| public static String cetakArray(int bariske, int columke){  String[][] kontak = {  {"Lili","08111"},  {"Lala","08122"},  {"Maya","08133"}  };  return kontak[bariske][columke];  } |

## Math

### Referensi

<Klik> or https://www.petanikode.com/java-math/

### Apa itu Class Math?

Class Math adalah class yang berisi method dan konstanta untuk melakukan perhitungan numerik seperti eksponensial, logaritma, akar kuadrat, dan fungsi trigonometri.1

Menurut saya, class ini sangat bermanfaat saat kita membuat program yang membutuhkan perhitungan matematis.

Cara menggunakan class Math di program

Karena class Math sudah menjadi class build-in atau bawaan di Java, maka kita tidak perlu mengimpor untuk menggunakannya.

Kita bisa langsung menggunaknnya seperti ini:

|  |
| --- |
| Math.PI; // konstanta PI  Math.pow(2,4); // fungsi pangkat di Math |

Contoh:

|  |
| --- |
| *public* *static* void main(String[] args) {  float keliling;  float PI;  int r;    r = 7;    keliling = (float) (2 \* Math.PI \* r);  System.out.println("Keliling: " + keliling);  } |

Pada contoh ini kita menggunakan konstanta PI dari class Math yakni dengan menulis Math.PI.

Selain konstanta PI, masih banyak isi yang lainnya seperti fungsi-fungsi matematika dasar.

## Equality

### Referensi

[Klik](https://codepolitan.com/blog/membandingkan-dua-buah-string-di-java-5987344c60f3d) or <https://codepolitan.com/blog/membandingkan-dua-buah-string-di-java-5987344c60f3d>

### Pembahasan

* Membandingkan dua String menggunakan metode equals di Java

Metode equals() membandingkan dua String berdasarkan kontennya. Sehingga, apabila dua buah String memiliki huruf-huruf yang sama, diurutan yang sama dan besar-kecil yang sama maka dua String ini akan dianggap sama. Oleh karena itu "Codepolitan" dan "Codepolitan" akan mengembalikan nilai true sedangkan "Codepolitan" dan "CODEPOLITAN" akan mengembalikan nilai false.

* Membandingkan String dengan equalsIgnoreCase

Dibandingkan dengan equals, metode equalsIgnoreCasemembandingkan dua String dengan mengabaikan besar-kecilnya. Oleh karena itu, String "Codepolitan" dan "CODEPOLITAN" adalah dua buah String yang sama menurut equalsIgnoreCase dan menghasilkan nilai true.

* Membandingkan String menggunakan compareTo

Berbeda dengan dua metode sebelumnya, compareTo membandingkan dua buah String berdasarkan urutan leksikografisnya (posisi kamus). Penulisan compareTo adalah dengan menuliskan "google".compareTo("codepolitan"). Metode ini akan mengembalikan sebuah nilai int yang dapat diartikan sebagai berikut:

Jika nilai yang dikembalikan kurang dari nol, maka String yang pertama memiliki posisi kamus lebih dulu.

Jika nilai yang dikembalikan sama dengan nol maka kedua String memiliki posisi kamus yang sama.

Jika nilai yang dikembalikan lebih dari nol maka String yang kedua (yang dikirimkan ke parameter compareTo) memiliki posisi kamus lebih dulu.

* Membandingkan String menggunakan compareToIgnoreCase

Metode ini mengabaikan besar-kecil String seperti halnya equalsIgnoreCase() dan mengembalikan nilai yang sama dengan metode compareTo().

* Jangan menggunakan "==" untuk membandingkan String

Banyak programmer Java membuat kesalahan dengan menggunakan "==" untuk melakukan perbandingan String. Simbol "==" membandingkan referensi dua variabel apakah menunjuk ke objek yang sama di heap Java. Selalu gunakan salah satu metode di atas untuk membandingkan konten dua buah String di Java.

* Source Code

Berikut ini contoh program Java yang membandingkan dua buah String.

|  |
| --- |
| public class StringComparisonExample {  public static void main(String args[]) {  String tv = "Bravia";  String television = "Bravia";  *// Contoh perbandingan String dengan equals*  if (tv.equals(television)) {  System.out.println("Both tv and television contains same letters and equal by equals method of String");  }  *// Contoh perbandingan String dengan compareTo*  if (tv.compareTo(television) == 0) {  System.out.println("Both tv and television are equal using compareTo method of String");  }  television = "BRAVIA";  *// Contoh perbandingan String dengan equalsIgnoreCase*  if (tv.equalsIgnoreCase(television)) {  System.out.println("tv and television are equal by equalsIgnoreCase method of String");  }  *// Contoh perbandingan String dengan compareToIgnoreCase*  if (tv.compareToIgnoreCase(television) == 0) {  System.out.println("tv and television are same by compareToIgnoreCase of String");  }  String sony = "Sony";  String samsung = "Samsung";  *// Contoh perbandingan String dengan compareTo*  if (sony.compareTo(samsung) > 0) {  System.out.println("Sony comes after Samsung in lexicographical order");  } else if (sony.compareTo(samsung) < 0) {  System.out.println("Sony comes before Samsung in lexicographical order");  }  }  } |

Berikut ini keluaran program Java di atas:

Both tv and television contains same letters and equal by equals method of String

Both tv and television are equal using compareTo method of String

tv and television are equal by equalsIgnoreCase method of String

tv and television are same by compareToIgnoreCase of String

Sony comes after Samsung in lexicographical order

## Operators Operasi dasar matematis dan perbandingan

Ada enam jenis kelompok operator dalam pemrograman Java:

* Operator Arithmetic
* Operator Relational
* Operator Logical
* Operator Assignment
* Operator Ternary
* Operator Bitwise

### Operator Artimatika

Operator aritmatika digunakan untuk melakukan operasi aritmatika.

Contoh : int M=100, N=50;

Result = M + n;  
M+=n;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operator** | **meaning** | **description** | **example** | **result** |
| + | Addition | Adds numbers or variables or both | M+N | 150 |
| - | Subtraction | subtract numbers or variables or both | M-N | 50 |
| \* | Multiplication | Multiply numbers or variables or both | M\*N | 5000 |
| / | Division | Divides numbers or variables or both and returns quotient of division | M/N | 2 |
| % | Modulus | modulus division of numbers or variables or both and returns remainder of division | M%N | 0 |
| ++ | Increment | Increments value of variable by 1 | ++M | M=101 |
| - - | Decrements | Decrements value of variable by 1 | --M | M=99 |
| += | Addition assignment | Adds left hand side variable with right hand side variable and stores result in left hand side variable. Its is equivalent to M=M+N | M+=N | M=150; |
| -= | Substraction assignment | Substracts right hand side variable from left side variable and stores result in left hand side variable. Its is equivalent to M=M- | M-=N | M=50; |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | N |  |  |
| \*= | Multiplication assignment | Multiplies left hand side variable with right hand side variable and stores result in left hand side variable. Its is equivalent to M=M\*N | M\*=N | M=5000; |
| /= | Division assignment | Divides left hand side variable with right hand side variable and stores quotient of division in left hand side variable. Its is equivalent to M=M/N | M/=N | M=2; |
| %= | Modulus assignment | Divides left hand side variable with right hand side variable and stores remainder of division in left hand side variable. Its is equivalent to M=M%N | M%=N | M=0; |

Contoh Program sbb:

|  |
| --- |
| int penjumlahan = 100 + 50;  int pengurangan = 100 - 50;  int A = 10;  int B = 5;  int perkalian = A \* B;  System.out.println("Hasil jumlah = " + penjumlahan);  System.out.println("Hasil kurang = " + pengurangan);  System.out.println("Hasil kali = " + perkalian);  System.out.println("Hasil bagi = " + A / B);  System.out.println("Hasil sisa bagi = " + A % B); |

**Latihan**

* Program menghitung keliling persegi panjang.

Rumus : K = 2 (p + l)

K: Keliling

p: Panjang

l: lebar

### Operator Relational

Relational operator digunakan untuk membandingkan antar dua operand yang saling berhubungan. Pada tabel berikut dicontohkan variables A dan B di definisikan integer dan diberi nilai integer dan menghasilkan nilai logika true atau false

Contoh

**int A=100, B=100;**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operator** | **meaning** | **description** | **example** | **result** |
| == | Equal to | If value of left hand side variable is equal to value of right hand side then returns true else returns false | A==B | 100==100    true |
| != | Not equal to | If value of left hand side variable is not equal to value of right hand side then returns true else returns false | A!=B | 100!=100    false |
| < | Less than | If value of left hand side variable is less than value of right hand side then returns true else returns false | A< B | 100<100    false |
| <= | Less than equal to | If value of left hand side variable is less than or equal to the value of right hand side then returns true else returns false | A<= B | 100<100    true |
| > | Greater than | If value of right hand side variable is greater than value of left hand side then returns true else returns false | B>A | 100>100    false |
| >= | Greater than equal to | If value of right hand side variable is greater than or equal to the value of left hand side then returns true else returns false | B>=A | 100>100 true |

**Contoh program**

|  |
| --- |
| int a = 10;  int b = 20;  boolean hasil = a == b;  boolean hasil2 = a != b;  boolean hasil3 = a > b;  boolean hasil4 = a < b;  boolean hasil5 = a >= b;  boolean hasil6 = a <= b;  System.out.println("Hasil1 = " + hasil);  System.out.println("Hasil2 = " + hasil2);  System.out.println("Hasil3 = " + hasil3);  System.out.println("Hasil4 = " + hasil4);  System.out.println("Hasil5 = " + hasil5);  System.out.println("Hasil6 = " + hasil6); |

### Operator Logical

Operator logical di gunakan untuk membandingkan nilai dari dua operand yang saling berelasi. Pada table berikut dicontohkan variables A dan B di definisikan dengan nilai boolean.

**boolean A=true, B=false;**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operator** | **meaning** | **description** | **example** | **result** |
| && | Logical AND | Performs logical AND operation on | A&&B | true |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | two operands. If both operands are true then returns true else returns false |  | false    false |
| || | Logical OR | Performs logical OR operation on two operands. If one or more than operand are true then returns true else returns false | A||B | true false    true |
| ! | Logical NOT | Converts true value to false and vice versa | !A | !true    false |

**Contoh**

|  |
| --- |
| int nilaiA = 70;  int nilaiB = 80;  int nilaiC = 90  boolean a = nilaiA > nilaiB && nilaiB < nilaiC || nilaiC < nilaiB;  boolean b = nilaiA < nilaiB || nilaiB < nilaiC && nilaiC < nilaiB;  //Jika dijalankan  System.out.println(a); //hasilnya: false  System.out.println(b); //hasilnya: true |

### Operator Assignment

Assignment operator di gunakan untuk mengisi nilai yang ada di kanan assign dan menyimpannya ke variable. Symbol sign adalah (=). Secara umum deklarasi assign sbb :

**variable = expression or values;**

**Contoh**:

int a=20; // Equal operator assigns 20 to variable a.

### Operator Ternary

Ternary operator menyerupai if else statement didalam java. Secara umum deklarasi operator ini sbb :

**Syntax**

|  |
| --- |
| **Expression1 ? Expression2 : Expression 3 ; OR**  **Expression1 ? value1 : value2;**  Maksudnya adalah :  If Expression1 benar true, maka value1 yang akan di return else value2 yang akan return. |

**Contoh** :

System.out.println( 10>5? “true” : “false” );

**Output**:

True

**Contoh lain**

|  |
| --- |
| boolean suka = true;  String jawaban;  // menggunakan operator ternary  jawaban = suka ? "iya" : "tidak";  // menampilkan jawaban  System.out.println(jawaban); |

### Operator Bitwise

Operator bitwise merupkan operator yang digunakan untuk operasi bit (biner)  
  
Operator ini berlaku untuk tipe data int, long, short, char, dan byte.

Operator ini akan menghitung dari bit-ke-bit.

Misalnya, kita punya variabel a = 60 dan b = 13.

Bila dibuat dalam bentuk biner, akan menjadi seperti ini:

|  |
| --- |
| a = 00111100  b = 00001101 |

Konsepnya memang hampir sama dengan opeartor Logika. Bedanya, Bitwise digunakan untuk biner.

Contoh

int M=100, N=50;

nilai binary M=100 is **1100100**

nilai binary N= 50 is **0110010**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operator** | **meaning** | **description** | **example** | **result** |
| ~ | Bitwise NOT | Performs one's complement operation | ~M | 1100100 (M)    0011011(~M) |
| & | Bitwise AND | Performs bitwise AND i.e bit by bit binary multiplication between  two variables or numbers or both | M & N | 1100100(M)  0110010(N)    0100000(&) |
| | | Bitwise OR | Performs bitwise OR between two variables or numbers or both | M | N | 1100100(M)  0110010(N)    1110110(|) |
| ^ | Bbitwise EX- OR | Performs bitwise EX-OR between two variables or numbers or both | M ^ N | 1100100(M)  0110010(N)    1010110(^) |
| >> | Right shift | The right shift operator, >> , shifts all of the bits in a value to the right a specified number of times.  Syntax: value >> shift number | M>>1 | 1100100 ( M )    0110010(M>>1) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| << | Left shift | The left shift operator, << , shifts all of the bits in a value to the left a specified number of times.  Syntax: value << shift number | M<<1 | 01100100 ( M )    11001000(M<<1) |
| &= | Bitwise AND assignment | Performs bitwise AND left hand side variable and right hand side variable and assigns result to left hand side variable  M&=N is equvivalent to M=M&N; | M&=N | 1100100(M)  0110010(N)    0100000(&) |
| |= | Bitwise OR assignment | Performs bitwise OR left hand side variable and right hand side variable and assigns result to left hand side variable  M|=N is equvivalent to M=M|N; | M|=N | 1100100(M)  0110010(N)    1110110(|) |
| ^= | Bitwise EX- OR  assignment | Performs bitwise EX-OR left hand side variable and right hand side variable and assigns result to left hand side variable  M^=N is equvivalent to M=M^N; | M^=N | 1100100(M)  0110010(N)    1010110(^) |
| <<= | Left shift assignment | The left shift operator, << , shifts all of the bits of left hand side variable to the left a specified number of times and stores result in left hand side variable  Syntax: value << shift number  M<<=N is equvivalent to M=M << N; | M<<= N | 01100100 ( M )    11001000(M<<1) |
| >>= | right shift assignment | The right shift operator, >> , shifts all of the bits of left hand side variable to the right a specified number of times and stores result in left hand | M>>= N | 1100100 ( M ) |

Contoh Program sbb :

|  |
| --- |
| 1. int a = 181; int b = 108; int hasil; 2. hasil = a & b; 3. System.out.println("Hasil dari a & b : " + hasil ); 4. hasil = a | b; 5. System.out.println("Hasil dari a | b : " + hasil ); 6. hasil = a ^ b; 7. System.out.println("Hasil dari a ^ b : " + hasil ); 8. hasil = ~a; 9. System.out.println("Hasil dari ~a : " + hasil ); 10. hasil = a >> 1; 11. System.out.println("Hasil dari a >> 1 : " + hasil ); 12. hasil = b << 2; 13. System.out.println("Hasil dari b << 2 : " + hasil ); |

**Hasil**

Hasil dari a & b : 36

Hasil dari a | b : 253

Hasil dari a ^ b : 217

Hasil dari ~a : -182

Hasil dari a >> 1 : 90

Hasil dari b << 2 : 432

Dalam contoh di atas saya mendefinisikan 2 variabel: **a** dan **b**. Kemudian memberikan nilai awal **181** dan **108**. Jika di konversi ke dalam bentuk biner, keduanya berisi angka berikut:

a = 181 (desimal) = 10110101 (biner)

b = 108 (desimal) = 01101100 (biner)

Di baris 8 saya melakukan operasi **&** (AND) terhadap kedua variabel. Operasi bitwise “and” ini akan memproses bit per bit dari kedua variabel, jika kedua bit sama-sama 1, maka hasilnya juga 1, selain kondisi tersebut, nilai akhirnya adalah 0. Berikut perhitungan bitwise “and”:

a = 10110101

b = 01101100

a & b = 00100100 = 36 (desimal)

Tampilan perintah **System.out.println()** di baris 9 memperlihatkan isinya sama dengan perhitungan kita, yakni 36 (dalam bentuk desimal).

Di baris 11 terdapat operasi **|** (OR), hasilnya akan bernilai 0 jika kedua bit bernilai 0, selain itu nilai bit akan di set menjadi 1. Berikut cara perhitungan bitwise “or”:

a = 10110101

b = 01101100

a | b = 11111101 = 253 (desimal)

Di baris 14 terdapat operasi **^** (XOR), hasilnya akan bernilai 1 jika salah satu dari kedua variabel bernilai 1 (namun tidak keduanya). Atau dengan kata lain jika kedua bit berlainan, hasilnya 1 tapi kalau sama-sama 0 atau sama-sama 1, hasilnya 0.

Berikut cara perhitungan bitwise “xor”:

a = 10110101

b = 01101100

$a ^ $b = 11011001 = 217 (desimal)

Di baris 17 saya membuat operasi **~** atau not, yang akan membalikkan nilai bit sebuah variabel dari 0 menjadi 1, dan 1 menjadi nol. Namun perhitungan bit not ini sedikit membingungkan karena jika kita hanya membalikkan seluruh bit saja, hasilnya tidak sesuai dengan apa yang dihitung oleh compiler Java:

a = 10110101

~a = 01001010 = 74 (desimal) ==> salah ???

Dari hasil menjalankan program, dapat dilihat bahwa ~a = -182, dari manakah datangnya angka -182?

Ini berkaitan dengan cara compiler bahasa Java menyimpan angka biner (dan juga hampir semua bahasa pemrograman komputer modern). Angka biner di dalam bahasa Java disimpan dalam format “Two’s complement”. Penjelasan tentang “Two’s complement” ini cukup panjang, jika tertarik saya sudah membahasnya lengkap di buku [Pascal Uncover](https://www.duniailkom.com/pascal-uncover-panduan-belajar-pascal-untuk-pemula/), atau bisa ke [Two’s complement](https://en.wikipedia.org/wiki/Two%27s_complement) [Wikipedia](https://en.wikipedia.org/wiki/Two%27s_complement).

Secara singkat, rumusnya adalah -a - 1, sehingga ~a = -181 - 1 = - 182 (desimal)

## Expression, Statement dan Block

### Ekspresi

Ekspresi merupakan suatu bentuk yang menghasilkan suatu nilai. Perhatikan ekspresi berikut ini:

|  |  |
| --- | --- |
| 1  2 | int a; //mendefinisikan variabel bernama a  a = 10 + 5; //10+5 adalah ekspresi karena menghasilkan nilai suatu nilai yaitu 15 |

### Statement

Statement adalah kalimat lengkap dari bahasa pemrograman. Di java, Statement diakhiri oleh tanda semikolon (;). Ada tiga jenis statement yaitu:

Assignment statement

Increment dan dercement statement

Method invocation

Object creation statement

Berikut ini adalah contoh-contoh dari jenis-jenis statement di atas:

|  |  |
| --- | --- |
| 1  2  3  4  5 | byte skor = 13; //ini adalah assignment statement  nilai++; //ini adalah increment statement  nilai--; //ini adalah decrement statement  System.out.println(“Halo saya pauzan”); //ini adalah method invocation statement  Age age = new Age(); //ini adalah object creation statement |

### Blok

Blok adalah kumpulan statement, bisa lebih dari 1, 2, 3 dst statement. Blok dimulai dengan tanda kurung kurawal { diakhiri dengan }. Perhatikan contoh berikut ini:

|  |  |
| --- | --- |
| 1  2  3  4  5 | public static void main(String[] args) { **//blok dimulai**    int angka = 10;    angka++;    System.out.println (angka);  } **//akhir dari blok** |

Perhatikan bahwa blok dari public static void main(String[] args){} terdiri dari banyak statement. Jadi setiap kita menemukan tanda {} maka hal tersebut adalah blok.

## Struktur Percabangan

**Statement seleksi Java antara lain sbb** :

* Struktur **if**
* Struktur **if-else**
* Struktur **if-else-if**
* Struktur **nested-if**
* Struktur **switch-case**

Statement tersebut memberikan programmer kemampuan untuk mengendalikan alur dari eksekusi program berbasis kondisi yang diketahui pada saat waktu program berjalan.

### Struktur if-else-if

Sturktur if digunakan untuk menentukan sebuah statement (atau blok statemen) yang akan di eksekusi jika dan hanya jika persyaratan Boolean (Boolean statement) bernilai **true**.

**Berikut format dasar penulisan kondisi IF ELSE IF dalam bahasa Java:**

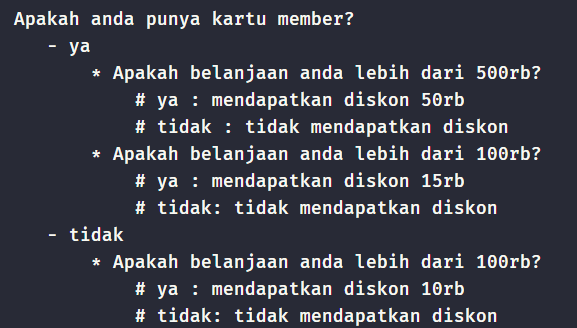
|  |
| --- |
| if (condition\_1) {    // Kode program yang dijalankan jika condition\_1 berisi nilai True  }  else if (condition\_2) {    // Kode program yang dijalankan jika condition\_2 berisi nilai True  }  else if (condition\_3) {    // Kode program yang dijalankan jika condition\_3 berisi nilai True  }  else  {    // Kode program yang dijalankan jika semua kondisi tidak terpenuhi  } |

**Contoh porgram**

|  |
| --- |
| int time = 20;  if(time < 10){  System.out.println("good morning");  }else if(time >20){  System.out.println("good day");  }else{  System.out.println("good evening");  } |

### Struktur nested-if

Percangan di dalam if ada if lagi**. asumsi**



**Contoh**

|  |
| --- |
| int belanjaan = 600000;  int diskon ,bayar ;  String kartu= "ya";  // proses  if (kartu.equalsIgnoreCase("ya")) {  if (belanjaan > 500000) {  diskon = 50000;  } else if (belanjaan > 100000) {  diskon = 15000;  } else {  diskon = 0;  }  } else {  if (belanjaan > 100000) {  diskon = 5000;  } else {  diskon = 0;  }  }  // total yang harus dibayar  bayar = belanjaan - diskon;  // output  System.out.println("Total Bayar: Rp " + bayar); |

### Struktur switch … Case ...

Cara lain untuk membuat percabangan adalah dengan menggunakan kata kunci switch. Dengan menggunakan switch kita bisa melakukan percabangan dengan persyaratan yang beragam.

**Syntax**

|  |
| --- |
| switch (nama\_variabel) {    case 'nilai\_1':      // Kode program yang dijalankan jika nama\_variabel == nilai\_1      break;    case 'nilai\_2':      // Kode program yang dijalankan jika nama\_variabel == nilai\_2      break;    case 'nilai\_3':      // Kode program yang dijalankan jika nama\_variabel == nilai\_3      break;    ...    ...    default:       // Kode program yang dijalankan jika tidak ada kondisi yang terpenuhi  } |

**Contoh program**

|  |
| --- |
| char nilai = 'B';  switch (nilai) {  case 'A':  System.out.println("Pertahankan!");  break;  case 'B':  System.out.println("Harus lebih baik lagi");  break;  case 'C':  System.out.println("Perbanyak belajar");  break;  case 'D':  System.out.println("Jangan keseringan main");  break;  case 'E':  System.out.println("Kebanyakan bolos...");  break;  default:  System.out.println("Maaf, format nilai tidak sesuai");  } |

## Looping For, While, DO-While

**Looping** pada bahasa pemrograman **Java** adalah fitur yang digunakan untuk memfasilitasi eksekusi program dari sekelompok instruksi atau fungsi secara berulang ketika beberapa kondisi tertentu yang menjadi syarat adalah bernilai benar

### Looping with FOR

* **for**

Sturktur pengulangan for yaitu melakukan Pengulangan eksekusi code beberapa kali.

|  |
| --- |
| for (InitExpression; Loopcondition; StepExpression )  {  statements;  } |

**Keterangan**

* InitExpression – Inisialisasi dari variabel loop.
* LoopCondition – membandingkan variabel lop pada nilai batas.
* StepExpression – melakukan update pada variabel loop

**Contoh** **program**

|  |
| --- |
| 1. **For**   for(int i=1;i<=10;i++){ 2. System.out.println(i); 3. }   **Looping for dalam for**  //loop of i  for(int i=1;i<=3;i++){  //loop of j  for(int j=1;j<=3;j++){  System.out.println(i+" "+j);  }//end of i  }//end of j |

* **Java for-each Loop**

for Each Loop ini berfungsi untuk mencetak isi elemen array:

**Syntax**

|  |
| --- |
| 1. for(data\_type variable : array\_name){ 2. //code to be executed 3. } |

**Contoh program**

|  |
| --- |
| 1. //Declaring an array 2. int arr[]={12,23,44,56,78}; 3. //Printing array using for-each loop 4. for(int i:arr){ 5. System.out.println(i); 6. } |

**Keterbatasan dari for-each loop**

* Penggunaan *for-each loop* menjadi tidak sesuai ketika ingin melakukan modifikasi nilai *array*
* For-each loop tidak dapat memproses dua statement pembuat keputusan secara bersamaan

### Looping with While

While Loop adalah statement atau blok statement yang diulang-ulang mencapai kondisi yang cocok.  
 - do while : jalanin dulu baru chek   
- while : chek kondisi lalu jalanin

**Syntax**

|  |
| --- |
| 1. **while** (condition){ 2. //code to be executed 3. } |

**Contoh**

|  |
| --- |
| 1. **int** i=1; 2. **while**(i<=10){ 3. System.out.println(i); 4. i++; 5. } |

### Looping do while

Do-while loop mirip dengan while-loop. Statemen di dalam do-while loop akan dieksekusi beberapa kali selama kondisi bernilai true. Perbedaan antara while dan do-while loop adalah statemen di dalam do-while loop dieksekusi sedikit satu kali.

**Syntax**

|  |
| --- |
| 1. do { 2. // code block to be executed 3. } 4. while (condition); |

**Contoh Program**

|  |
| --- |
| 1. int i = 0+1 =1; 2. do { 3. System.out.println(i); 4. i++; 1 + 1 = 2 5. } 6. while (2 < 5); |

### Perintah Jump

Java mendukung tiga statement Jump/Melompat yaitu:

* break
* continue
* return
* **Break**

Menggunakan break dapat memaksa program keluar dari loop, dengan cara melewati kondisi ekspresi dari kode program yang tersisa pada looping.

**Contoh program**

|  |
| --- |
| 1. //using for loop 2. for(int i=1;i<=10;i++){ 3. if(i==6){   //breaking the loop  break;   1. } 2. System.out.println(i); 3. } |

**Output  
1**

**2**

**3**

**4**

**5**

* **Continue**

Statement continue sangat berguna untuk memaksa iterasi lebih awal dari proses loop, dimana programmer dapat melanjutkan program pada looping tetapi menghentikan proses sisa dari kode program di dalam body statement untuk iterasi tertentu

**Contoh program**

|  |
| --- |
| //for loop  for(int i=1;i<=10;i++){  if(i==5){  //using continue statement  continue;//it will skip the rest statement  }  System.out.println(i);  } |

**1**

**2**

**3**

**4**

**-continue**

**6**

**7**

* **Return**

Statement return digunakan untuk eksplisit return dari method. Sehingga, return menyebabkan kendali program untuk mengirim kembali menuju pemanggil (caller) program dari method.

**Contoh program**

|  |
| --- |
| public static void returnJump(){  for(int i=1;i<=10;i++){  if(i==5){  return ; if return method value 5  }  System.out.println(i);  }  } |

Latihankelas

A diagram of a flowchart

Description automatically generated

## Recursive

Recursive method adalah kemampuan method memanggil method dirinya sendiri. Supaya lebih jelas pertama kita buat koding tanpa menggunakan recursive method, kita buat kelas bernama RecursiveMethod untuk menghitung faktorial 6. Perhatikan sintaks berikut ini:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | public class RecursiveMethod {      public static void main(String[] args) {            System.out.println(faktorial(6)); //memanggil method faktorial, diisikan 6 untuk cari faktorial 6      }        //deklarasi method bernama faktorial      static int faktorial(int angkaFaktorial){          var hasil = 1;          for(var angka = 1; angka <= angkaFaktorial; angka++){ //perulangan dari 1 sampai dengan angkaFaktorial - 1              hasil \*= angka; //artinya hasil = hasil \* angka          }          return hasil; //nilai dari hasil diberikan ke method faktorial      }  } |

Jika kita run maka hasilnya adalah:

|  |  |
| --- | --- |
| 1 | 720 //ini adalah factorial 6 |

Sekarang kita coba menggunakan recursive method, perhatikan sintaks di bawah ini:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | public class RecursiveMethod {      public static void main(String[] args) {            System.out.println(faktorial(6)); //memanggil method faktorial, diisikan 6 untuk cari faktorial 6      }        //deklarasi method bernama faktorial      static int faktorial(int angkaFaktorial){         if(angkaFaktorial == 1){ //blok ini dijalankan jika angkaFaktorial bernilai 1.             return 1;         }else{ //blok ini dijalankan jika angkaFaktorial tidak bernilai 1             //misal kita beri 5 pada angkaFaktorial, maka disini akan dilakukan operasi 5 \* 4, begitu seterusnya             return angkaFaktorial \* faktorial(angkaFaktorial - 1);         }      }  } |

Jika kita run maka hasilnya adalah:

|  |  |
| --- | --- |
| 1 | 720 //merupakan factorial dari 6 |

## Latihan – Aplikasi Kalkulator

Soal yang mencakup fitur terbaru untuk melihat total hasil semua operasi dalam riwayat kalkulator:

### Pertanyaan Dasar:

Buatlah sebuah program Java untuk kalkulator sederhana yang dapat melakukan penambahan, pengurangan, perkalian, dan pembagian. Program ini harus memiliki metode terpisah untuk setiap operasi dan meminta pengguna untuk memasukkan dua angka serta operator matematika.

### Input dan Output:

Buat program kalkulator yang meminta pengguna untuk memasukkan dua angka dan operator (+, -, \*, /). Program ini kemudian menampilkan hasil kalkulasi dan memungkinkan pengguna untuk mengulangi perhitungan atau keluar dari program.

### Tipe Data Referensi:

Modifikasi program Anda agar dapat menghitung rata-rata dari sejumlah angka yang dimasukkan pengguna. Gunakan array untuk menyimpan angka-angka ini dan gunakan loop untuk menghitung rata-ratanya.

### Operasi Matematis

Buat fungsi tambah dan kurang dalam program Anda yang dapat mengambil dua angka sebagai argumen dan mengembalikan hasil penambahan dan pengurangan tersebut.

### Penggunaan If-Else. Switch case:

Tambahkan kondisi pada program Anda untuk memeriksa apakah hasil kalkulasi adalah bilangan positif, negatif, atau nol. Tampilkan pesan yang sesuai kepada pengguna berdasarkan hasil ini.

### Ternary Operator:

Modifikasi program Anda untuk menampilkan pesan kepada pengguna berdasarkan apakah hasil kalkulasi adalah bilangan ganjil atau genap menggunakan operator ternary.

### Looping: while, for , do while

Berikan opsi kepada pengguna untuk melakukan lebih dari satu perhitungan tanpa harus menjalankan program ulang. Gunakan loop untuk mengimplementasikan fitur ini.

### Validasi Input:

Pastikan program Anda mampu memvalidasi masukan pengguna sehingga tidak akan terjadi kesalahan jika pengguna memasukkan karakter yang tidak valid.

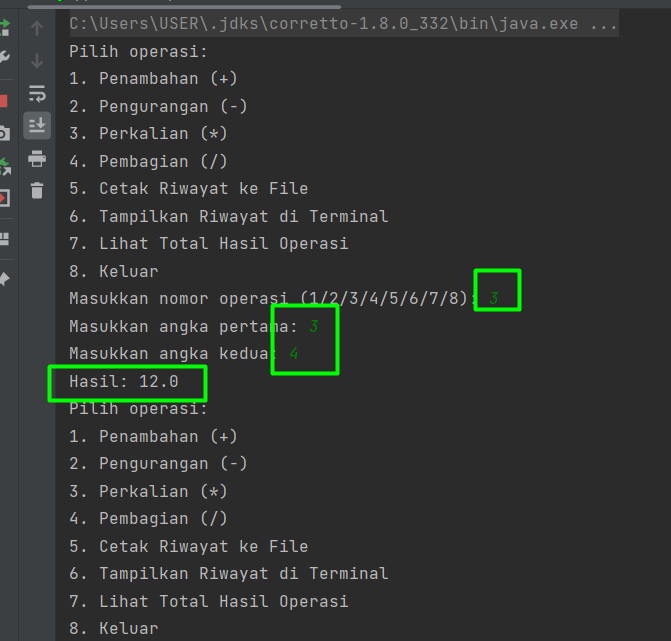
### Cetak history transaksi pada file txt

### Melihat Total Hasil Operasi:

Modifikasi program Anda untuk memungkinkan pengguna untuk melihat total hasil dari semua operasi dalam riwayat kalkulator.

### Contoh Output

#### Kalkulator operasi



#### Menampilkan riwayat transaksi

A screenshot of a computer program

Description automatically generated

#### Melihat total hasil transaksi

A screenshot of a computer program

Description automatically generated

#### Melihat output transaksi di file txt

A screenshot of a computer

Description automatically generated

### Branch/gitlab

[KLIK](https://gitlab.com/rikialdi/java-fundamental-final/-/tree/sesi3-looping-if-else-kalkulator) or <https://gitlab.com/rikialdi/java-fundamental-final/-/tree/sesi3-looping-if-else-kalkulator>

# CH1 TOP 5 - Programming Algorithm

## Materi

* + Konsep Algorithm
  + Cara Penggunaan Flowchart
  + Konsep Pseudocode
  + Identifikasi jenis algorithm melalui latihan

# CH1 TOP 6 - Java Standard Class

## Materi

|  |
| --- |
| * **String** * **Date and Calendar class, Timezone, localDate** * **Math Class** * **Regular expression** * **Write and read .txt dan .csv file** |

## Perbedaan StringBuilder dengan StringBuffer

<https://ichwansholihin.medium.com/perbedaan-antara-stringbuffer-dan-stringbuilder-di-java-5465a8d01005>

Kesimpulan

Pemilihan antara StringBuffer dan StringBuilder tergantung pada kebutuhan aplikasi Anda. Jika Anda perlu melakukan manipulasi string dalam lingkungan multi-threaded atau mengutamakan thread-safety, maka StringBuffer adalah pilihan yang baik. Namun, jika Anda bekerja dalam lingkungan single-threaded atau tidak memerlukan thread-safety, maka StringBuilder dapat memberikan kinerja yang lebih baik. Pilihan yang tepat akan memastikan bahwa aplikasi Anda berfungsi dengan efisien dan aman.

## StringBuffer

### Method

| **Methods** | **Action Performed** |
| --- | --- |
| append() | Used to add text at the end of the existing text. |
| length() | The length of a StringBuffer can be found by the length( ) method |
| capacity() | the total allocated capacity can be found by the capacity( ) method |
| charAt() | This method returns the char value in this sequence at the specified index. |
| delete() | Deletes a sequence of characters from the invoking object |
| deleteCharAt() | Deletes the character at the index specified by the *loc* |
| ensureCapacity() | Ensures capacity is at least equal to the given minimum. |
| insert() | Inserts text at the specified index position |
| length() | Returns the length of the string |
| reverse() | Reverse the characters within a StringBuffer object |
| replace() | Replace one set of characters with another set inside a StringBuffer object |

Refer : https://www.geeksforgeeks.org/stringbuffer-class-in-java/

contoh **StringBuffer**

|  |
| --- |
| public static void main(String[] args) {  StringBuffer stringBuffer = new StringBuffer("Hello");   // Mengubah isi string  stringBuffer.append(" World");   // Menyisipkan teks pada posisi tertentu  stringBuffer.insert(5, " Java");   // Menghapus sebagian string  stringBuffer.delete(6, 11);   // Menggantikan teks dengan teks lain  stringBuffer.replace(0, 5, "Hi");   // Mengonversi StringBuffer menjadi String  String result = stringBuffer.toString();   System.*out*.println(result); // Output: Hi World Java } |

## StringBuilder

### Contoh

|  |
| --- |
| public static void main(String[] args) {  StringBuilder stringBuilder = new StringBuilder("Hello");   // Mengubah isi string  stringBuilder.append(" World");   // Menyisipkan teks pada posisi tertentu  stringBuilder.insert(5, " Java");   // Menghapus sebagian string  stringBuilder.delete(6, 11);   // Menggantikan teks dengan teks lain  stringBuilder.replace(0, 5, "Hi");   // Mengonversi StringBuilder menjadi String  String result = stringBuilder.toString();   System.*out*.println(result); // Output: Hi World Java } |

## String Joiner

|  |
| --- |
| StringJoiner adalah kelas utilitas dalam Java yang digunakan untuk menggabungkan beberapa string dengan delimiter tertentu. Ini memungkinkan kita untuk mengonstruksi string secara fleksibel dengan menambahkan elemen satu per satu dan menentukan pemisah di antara setiap elemen. StringJoiner memungkinkan kita untuk membangun string dengan lebih mudah daripada menggunakan pendekatan konkatenasi string biasa, terutama ketika bekerja dengan kumpulan data. |

### Contoh penerapan

|  |
| --- |
| public static void main(String[] args) {  // Membuat StringJoiner dengan delimiter ","  StringJoiner joiner1 = new StringJoiner(",");  joiner1.add("Apple");  joiner1.add("Banana");  joiner1.add("Orange");   // Mengonversi StringJoiner menjadi string  String fruits1 = joiner1.toString();  System.*out*.println(fruits1); // Output: Apple,Banana,Orange   // Membuat StringJoiner dengan delimiter ", " dan prefix "[", suffix "]"  StringJoiner joiner2 = new StringJoiner(", ", "[", "]");  joiner2.add("Red");  joiner2.add("Green");  joiner2.add("Blue");   // Mengonversi StringJoiner menjadi string  String colors = joiner2.toString();  System.*out*.println(colors); // Output: [Red, Green, Blue] } |

### Implementasi dunia kerja

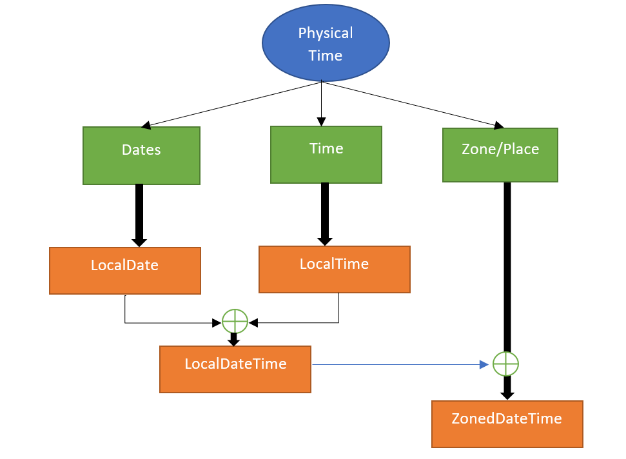
StringJoiner sering digunakan dalam pembuatan laporan atau pesan yang memerlukan penggabungan data dari beberapa sumber dengan format tertentu. Berikut adalah contoh penerapannya dalam skenario pengiriman email.

|  |
| --- |
| public static void main(String[] args) {  // Data penerima email  String recipientName = "Name";  String recipientEmail = "name@example.com";   // Informasi pesan email  String subject = "Pemberitahuan Penting";  String messageBody = "Halo, " + recipientName + "! Ini adalah pemberitahuan penting.";   // Daftar lampiran  String[] attachments = {"document1.pdf", "document2.pdf"};   // StringJoiner untuk membuat daftar lampiran dalam format yang diinginkan  StringJoiner attachmentList = new StringJoiner(", ");  for (String attachment : attachments) {  attachmentList.add(attachment);  }   // Menggabungkan semua informasi untuk membentuk pesan email lengkap  String emailContent = "To: " + recipientEmail + "\n" +  "Subject: " + subject + "\n\n" +  messageBody + "\n\n" +  "Lampiran: " + attachmentList.toString();   // Proses pengiriman email  *sendEmail*(recipientEmail, emailContent); }  private static void sendEmail(String recipientEmail, String emailContent) {  // Logika pengiriman email  System.*out*.println("Email sent to " + recipientEmail + " with content:\n" + emailContent); } |

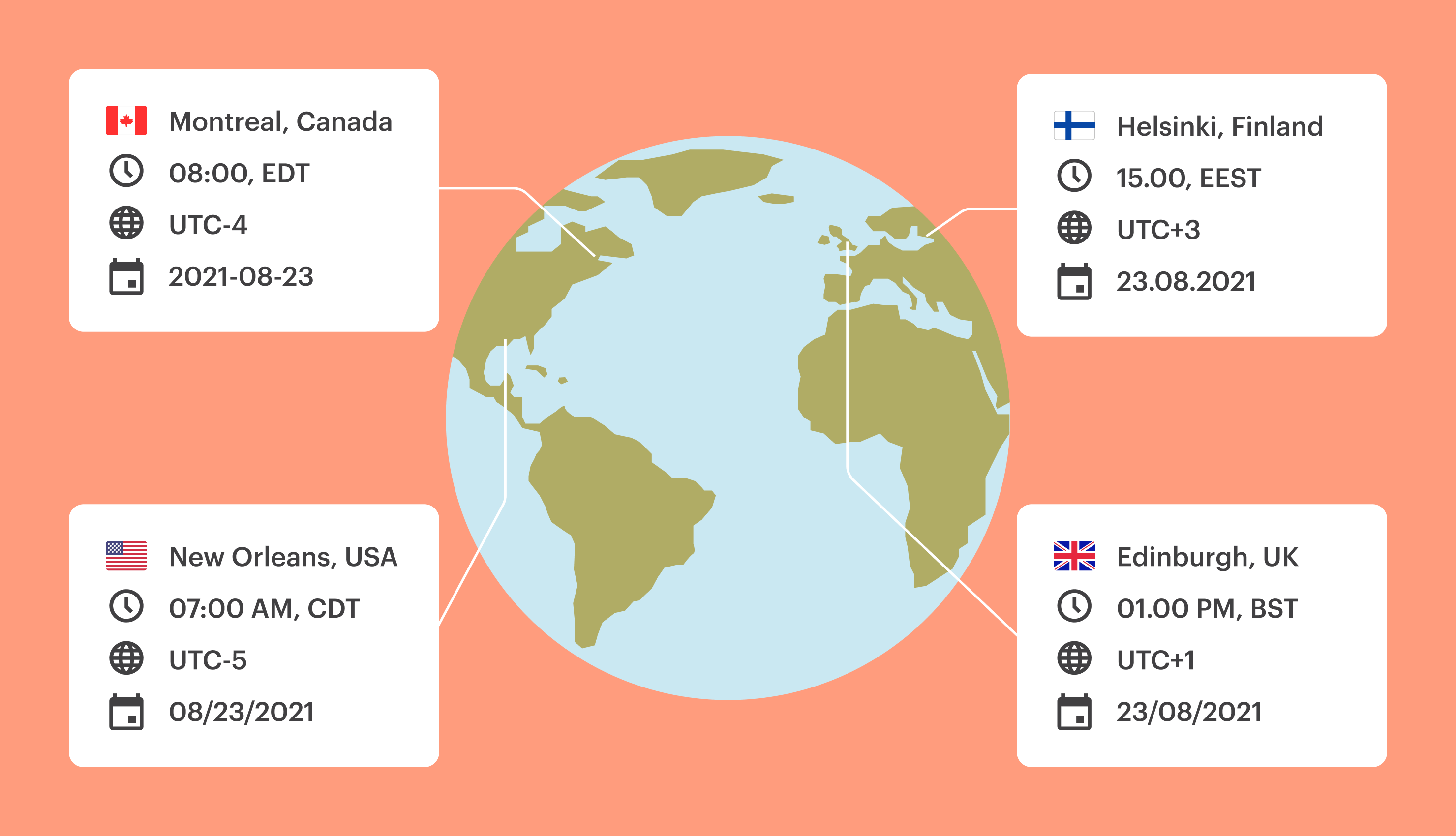
## Penerapan Stringbuilder dan StringBuffer dunia kerja

* Pembuatan Log
* Pemrosesan Data Tabular: Ketika aplikasi perlu menghasilkan laporan atau output dalam bentuk tabel, StringBuilder atau StringBuffer dapat digunakan untuk membangun baris-baris tabel secara dinamis.
* Pembuatan Pesan Email: Dalam aplikasi yang memungkinkan pengguna untuk mengirim email, StringBuilder atau StringBuffer dapat digunakan untuk membangun isi pesan email secara dinamis

## Konsep Date, Calender, TimeZone, LocalDate



Ref : https://pritampanhale-tech.medium.com/date-time-apis-in-java-8-d1d3b66b08d

Ex :beda wilayah ?  


Ref : https://lokalise.com/blog/java-localdate-localization/

## Apa perbedaan Date and Calendar class, Timezone, localDate

Perbedaan utama antara **Date** class, **Calendar** class, **Timezone**, dan **LocalDate** adalah dalam cara mereka mengelola dan merepresentasikan waktu dan tanggal. Berikut adalah perbedaan masing-masing:

1. Date class (java.util.Date):

* **Date** class adalah bagian dari Java sejak lama (sejak Java 1.0), tetapi saat ini dianggap sudah usang.
* Ini merepresentasikan waktu dan tanggal dalam bentuk timestamp (detik sejak Epoch) dan tidak memiliki informasi zona waktu.
* Tidak aman dalam penggunaan bersamaan (thread-unsafe).
* Tidak dapat mengatasi zona waktu dengan baik.

1. Calendar class (java.util.Calendar):

* Calendar class adalah perbaikan dari Date class yang lebih kuat dan dapat mengatasi berbagai perhitungan tanggal dan waktu.
* Ini mendukung zona waktu dan operasi terhadap tanggal seperti penambahan, pengurangan, dan pemformatan.
* Meskipun lebih kuat daripada Date, tetapi seringkali dianggap rumit dan sulit digunakan.

1. Timezone:

* Timezone adalah konsep yang digunakan untuk menentukan perbedaan waktu antara suatu wilayah geografis dengan zona waktu UTC (Coordinated Universal Time).
* Ini memungkinkan aplikasi untuk mengelola perbedaan waktu dan menyelaraskan data waktu dari berbagai zona waktu.
* Java menyediakan kelas TimeZone untuk mengelola informasi zona waktu.

1. LocalDate:

* LocalDate adalah bagian dari Java 8 dan kemudian (Java 8+), yang merupakan bagian dari API Date and Time yang lebih modern (java.time package).
* Ini merepresentasikan tanggal (tanpa jam atau zona waktu) dalam format yang lebih sederhana dan mudah digunakan.
* Sangat berguna untuk operasi yang tidak memerlukan informasi waktu atau zona waktu tertentu.

## Contoh : Date, Calender, TimeZone, LocalDate

|  |
| --- |
| public static void main(String[] args) {  // Contoh penggunaan kelas Date  Date currentDate = new Date();  System.*out*.println("Current Date: " + currentDate);   // Contoh penggunaan kelas Calendar  Calendar calendar = Calendar.getInstance();  int year = calendar.get(Calendar.YEAR);  int month = calendar.get(Calendar.MONTH) + 1; // Bulan dimulai dari 0  int day = calendar.get(Calendar.DAY\_OF\_MONTH);  System.*out*.println("Current Date (using Calendar): " + day + "/" + month + "/" + year);   // Contoh penggunaan kelas TimeZone  TimeZone timeZone = TimeZone.getDefault();  System.*out*.println("Default Timezone: " + timeZone.getID());   // Contoh penggunaan kelas LocalDate  LocalDate localDate = LocalDate.now();  System.*out*.println("Current Date (using LocalDate): " + localDate); } |

### Penjelasan

|  |
| --- |
| * Kelas **Date** digunakan untuk mendapatkan tanggal dan waktu saat ini. Namun, kelas ini telah usang dan tidak direkomendasikan untuk penggunaan baru. * Kelas **Calendar** digunakan untuk mendapatkan komponen-komponen tanggal dan waktu seperti tahun, bulan, dan hari. Namun, juga memiliki kekurangan dan telah digantikan oleh LocalDate di API baru. * Kelas **TimeZone** digunakan untuk mendapatkan zona waktu default sistem. * Kelas **LocalDate** dari paket java.time digunakan untuk merepresentasikan tanggal tanpa informasi waktu. Ini merupakan bagian dari API Date and Time yang diperkenalkan dalam Java 8 dan lebih disarankan untuk digunakan dalam pengembangan modern. |

## Latihan Txt buku telpon

|  |
| --- |
| package application.chapter1.sesi6; import java.io.\*; import java.util.ArrayList; import java.util.Scanner;  public class TextFileBukuTelpon {  private static final String *PHONE\_BOOK\_FILE* = "phonebook.txt";   public static void main(String[] args) {  Scanner scanner = new Scanner(System.*in*);  ArrayList<Contact> phoneBook = new ArrayList<>();   // Memeriksa apakah file phonebook.txt sudah ada, jika belum, maka buat file tersebut  if (!*fileExists*(*PHONE\_BOOK\_FILE*)) {  *createFile*(*PHONE\_BOOK\_FILE*);  } else {  phoneBook = *loadPhoneBook*();  }   while (true) {  System.*out*.println("=== Aplikasi Buku Telepon ===");  System.*out*.println("1. Tambah Kontak");  System.*out*.println("2. Cetak Buku Telepon");  System.*out*.println("3. Tampilkan Riwayat Buku Telepon");  System.*out*.println("4. Keluar");  System.*out*.print("Pilihan: ");   int choice = scanner.nextInt();   switch (choice) {  case 1:  *addContact*(scanner, phoneBook);  break;  case 2:  *printPhoneBook*(phoneBook);  break;  case 3:  *displayHistory*(phoneBook);  break;  case 4:  *savePhoneBook*(phoneBook);  System.*out*.println("Terima kasih! Aplikasi selesai.");  return;  default:  System.*out*.println("Pilihan tidak valid. Silakan coba lagi.");  break;  }  }  }   private static boolean fileExists(String fileName) {  File file = new File(fileName);  return file.exists();  }   private static void createFile(String fileName) {  try {  File file = new File(fileName);  if (file.createNewFile()) {  System.*out*.println("File " + fileName + " berhasil dibuat.");  } else {  System.*out*.println("File " + fileName + " sudah ada.");  }  } catch (IOException e) {  System.*out*.println("Terjadi kesalahan saat menciptakan file: " + e.getMessage());  }  }   private static ArrayList<Contact> loadPhoneBook() {  ArrayList<Contact> phoneBook = new ArrayList<>();  try (BufferedReader reader = new BufferedReader(new FileReader(*PHONE\_BOOK\_FILE*))) {  String line;  while ((line = reader.readLine()) != null) {  String[] parts = line.split(",");  if (parts.length == 2) {  String name = parts[0];  String phoneNumber = parts[1];  phoneBook.add(new Contact(name, phoneNumber));  }  }  } catch (IOException e) {  System.*out*.println("Gagal membaca buku telepon: " + e.getMessage());  }  return phoneBook;  }   private static void addContact(Scanner scanner, ArrayList<Contact> phoneBook) {  System.*out*.print("Masukkan Nama: ");  String name = scanner.next();  System.*out*.print("Masukkan Nomor Telepon: ");  String phoneNumber = scanner.next();  phoneBook.add(new Contact(name, phoneNumber));  System.*out*.println("Kontak berhasil ditambahkan.");  }   private static void printPhoneBook(ArrayList<Contact> phoneBook) {  System.*out*.println("=== Buku Telepon ===");  for (Contact contact : phoneBook) {  System.*out*.println(contact);  }  System.*out*.println("===================");  }   private static void displayHistory(ArrayList<Contact> phoneBook) {  System.*out*.println("=== Riwayat Buku Telepon ===");  for (Contact contact : phoneBook) {  System.*out*.println(contact.getName() + ": " + contact.getPhoneNumber());  }  System.*out*.println("============================");  }   private static void savePhoneBook(ArrayList<Contact> phoneBook) {  try (BufferedWriter writer = new BufferedWriter(new FileWriter(*PHONE\_BOOK\_FILE*))) {  for (Contact contact : phoneBook) {  //writer.write(contact.getName() + "," + contact.getPhoneNumber());  StringJoiner joiner = new StringJoiner(",");  joiner.add(contact.getName()).add(contact.getPhoneNumber());  writer.write(joiner.toString());  writer.newLine();  }  System.*out*.println("Buku telepon berhasil disimpan.");  } catch (IOException e) {  System.*out*.println("Gagal menyimpan buku telepon: " + e.getMessage());  }  } }  class Contact {  private String name;  private String phoneNumber;   public Contact(String name, String phoneNumber) {  this.name = name;  this.phoneNumber = phoneNumber;  }   public String getName() {  return name;  }   public String getPhoneNumber() {  return phoneNumber;  }   @Override  public String toString() {  return name + ": " + phoneNumber;  } } |

## Latihan txt dan csv buku telepon

|  |
| --- |
| import java.io.\*; import java.util.ArrayList; import java.util.Scanner;  public class TextFileBukuTelponAndCetakCSV {  private static final String *PHONE\_BOOK\_FILE* = "phonebook.txt";  private static final String *CSV\_FILE* = "phonebook.csv";   public static void main(String[] args) {  Scanner scanner = new Scanner(System.*in*);  ArrayList<Contact> phoneBook = new ArrayList<>();   // Memeriksa apakah file phonebook.txt sudah ada, jika belum, maka buat file tersebut  if (!*fileExists*(*PHONE\_BOOK\_FILE*)) {  *createFile*(*PHONE\_BOOK\_FILE*);  } else {  phoneBook = *loadPhoneBook*();  }   while (true) {  System.*out*.println("=== Aplikasi Buku Telepon ===");  System.*out*.println("1. Tambah Kontak");  System.*out*.println("2. Cetak Buku Telepon");  System.*out*.println("3. Tampilkan Riwayat Buku Telepon");  System.*out*.println("4. Cetak ke CSV");  System.*out*.println("5. Keluar");  System.*out*.print("Pilihan: ");   int choice = scanner.nextInt();   switch (choice) {  case 1:  *addContact*(scanner, phoneBook);  break;  case 2:  *printPhoneBook*(phoneBook);  break;  case 3:  *displayHistory*(phoneBook);  break;  case 4:  *printToCSV*(phoneBook);  break;  case 5:  *savePhoneBook*(phoneBook);  System.*out*.println("Terima kasih! Aplikasi selesai.");  return;  default:  System.*out*.println("Pilihan tidak valid. Silakan coba lagi.");  break;  }  }  }   private static boolean fileExists(String fileName) {  File file = new File(fileName);  return file.exists();  }   private static void createFile(String fileName) {  try {  File file = new File(fileName);  if (file.createNewFile()) {  System.*out*.println("File " + fileName + " berhasil dibuat.");  } else {  System.*out*.println("File " + fileName + " sudah ada.");  }  } catch (IOException e) {  System.*out*.println("Terjadi kesalahan saat menciptakan file: " + e.getMessage());  }  }   private static ArrayList<Contact> loadPhoneBook() {  ArrayList<Contact> phoneBook = new ArrayList<>();  try (BufferedReader reader = new BufferedReader(new FileReader(*PHONE\_BOOK\_FILE*))) {  String line;  while ((line = reader.readLine()) != null) {  String[] parts = line.split(",");  if (parts.length == 2) {  String name = parts[0];  String phoneNumber = parts[1];  phoneBook.add(new Contact(name, phoneNumber));  }  }  } catch (IOException e) {  System.*out*.println("Gagal membaca buku telepon: " + e.getMessage());  }  return phoneBook;  }   private static void addContact(Scanner scanner, ArrayList<Contact> phoneBook) {  System.*out*.print("Masukkan Nama: ");  String name = scanner.next();  System.*out*.print("Masukkan Nomor Telepon: ");  String phoneNumber = scanner.next();  phoneBook.add(new Contact(name, phoneNumber));  System.*out*.println("Kontak berhasil ditambahkan.");  }   private static void printPhoneBook(ArrayList<Contact> phoneBook) {  System.*out*.println("=== Buku Telepon ===");  for (Contact contact : phoneBook) {  System.*out*.println(contact);  }  System.*out*.println("===================");  }   private static void displayHistory(ArrayList<Contact> phoneBook) {  System.*out*.println("=== Riwayat Buku Telepon ===");  for (Contact contact : phoneBook) {  System.*out*.println(contact.getName() + ": " + contact.getPhoneNumber());  }  System.*out*.println("============================");  }   private static void savePhoneBook(ArrayList<Contact> phoneBook) {  try (BufferedWriter writer = new BufferedWriter(new FileWriter(*PHONE\_BOOK\_FILE*))) {  for (Contact contact : phoneBook) {  writer.write(contact.getName() + "," + contact.getPhoneNumber());  writer.newLine();  }  System.*out*.println("Buku telepon berhasil disimpan.");  } catch (IOException e) {  System.*out*.println("Gagal menyimpan buku telepon: " + e.getMessage());  }  }   private static void printToCSV(ArrayList<Contact> phoneBook) {  try (BufferedWriter writer = new BufferedWriter(new FileWriter(*CSV\_FILE*))) {  writer.write("Nama,Nomor Telepon");  writer.newLine();  for (Contact contact : phoneBook) {  writer.write(contact.getName() + "," + contact.getPhoneNumber());  writer.newLine();  }  System.*out*.println("Buku telepon berhasil dicetak ke dalam file CSV.");  } catch (IOException e) {  System.*out*.println("Gagal mencetak ke file CSV: " + e.getMessage());  }  } }  class Contact {  private String name;  private String phoneNumber;   public Contact(String name, String phoneNumber) {  this.name = name;  this.phoneNumber = phoneNumber;  }   public String getName() {  return name;  }   public String getPhoneNumber() {  return phoneNumber;  }   @Override  public String toString() {  return name + ": " + phoneNumber;  } } |

## Branch

|  |
| --- |
| sesi6-csvhttps://gitlab.com/rikialdi/java-fundamental-final.git  or https://gitlab.com/rikialdi/batch7ex |

## Soal latihan

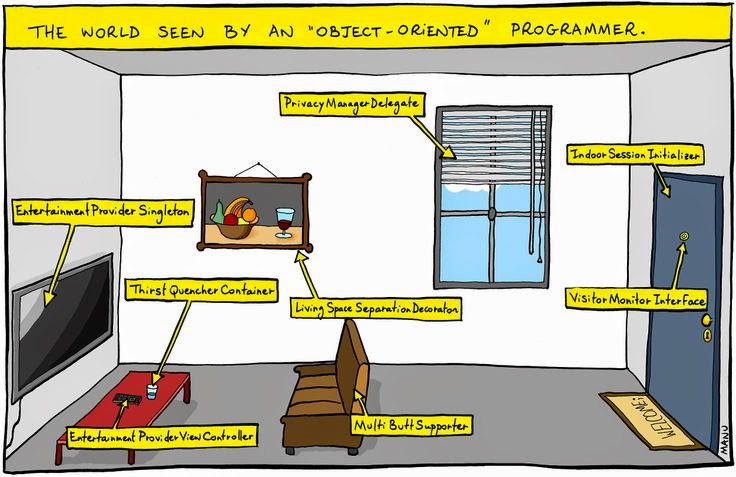
|  |
| --- |
| Anda adalah seorang pengembang perangkat lunak yang bertugas membuat aplikasi manajemen buku telepon sederhana menggunakan Java. Aplikasi ini memungkinkan pengguna untuk menambahkan, mencetak, dan melihat riwayat kontak dalam buku telepon mereka.   * Menambahkan Kontak: Pengguna dapat menambahkan kontak baru ke dalam buku telepon dengan memasukkan nama dan nomor telepon yang diinginkan. * Mencetak Buku Telepon: Pengguna dapat mencetak daftar kontak yang ada dalam buku telepon, yang mencakup nama dan nomor telepon setiap kontak. * Melihat Riwayat Buku Telepon: Pengguna dapat melihat riwayat kontak yang telah ditambahkan ke dalam buku telepon, termasuk nama dan nomor telepon setiap kontak. * Simpan Buku Telepon: Aplikasi ini akan menyimpan buku telepon ke dalam sebuah file teks setelah pengguna memilih untuk keluar dari aplikasi. * Pengaturan Waktu: Aplikasi ini akan menyesuaikan tanggal dan waktu dengan zona waktu setempat pengguna, untuk memastikan konsistensi penggunaan tanggal dan waktu. * Cetak history dengan format CSV, serta gunakan StringJoiner untuk menampung value. |

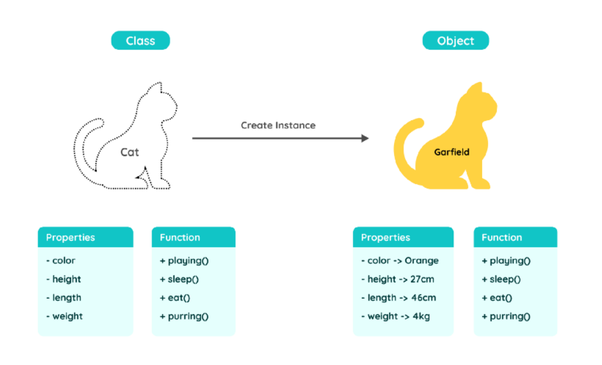
# CH2 TOP 1 - Java OOP Part 1

## Point Pembahasan

|  |
| --- |
|  |

## Pengantar Objek Oriented Programing(OOP) : Intro before Class





Ref : [Klik Link](https://www.google.com/url?sa=i&url=https%3A%2F%2Fid.quora.com%2FApa-perbedaan-Class-dan-Sub-Class-pada-bahasa-pemrograman-berorientasi-objek-OOP&psig=AOvVaw0P2HW8Qn8q1hAyjSGyowVO&ust=1710854897307000&source=images&cd=vfe&opi=89978449&ved=0CBUQjhxqFwoTCKDi4aT1_YQDFQAAAAAdAAAAABAE)

## Apa itu OOP ?

OOP (Object Oriented Programming) adalah suatu metode pemrograman yang berorientasi kepada objek.

Ini digunakan untuk menyusun program *software* (perangkat lunak) menjadi potongan-potongan sederhana dari cetak biru kode (biasanya disebut *class*) yang digunakan untuk membuat *instance* objek secara individu.

Karena OOP merupakan paradigma pemrograman, ada banyak bahasa pemrograman yang berorientasi objek termasuk seperti C++, Java, Python dan Ruby serta lainnya.

## Apa tujuan OOP ?

Tujuan dari OOP diciptakan adalah untuk mempermudah pengembangan program dengan cara mengikuti model yang telah ada di kehidupan sehari-hari. Tujuan OOP juga untuk mempermudah pengembangan program.

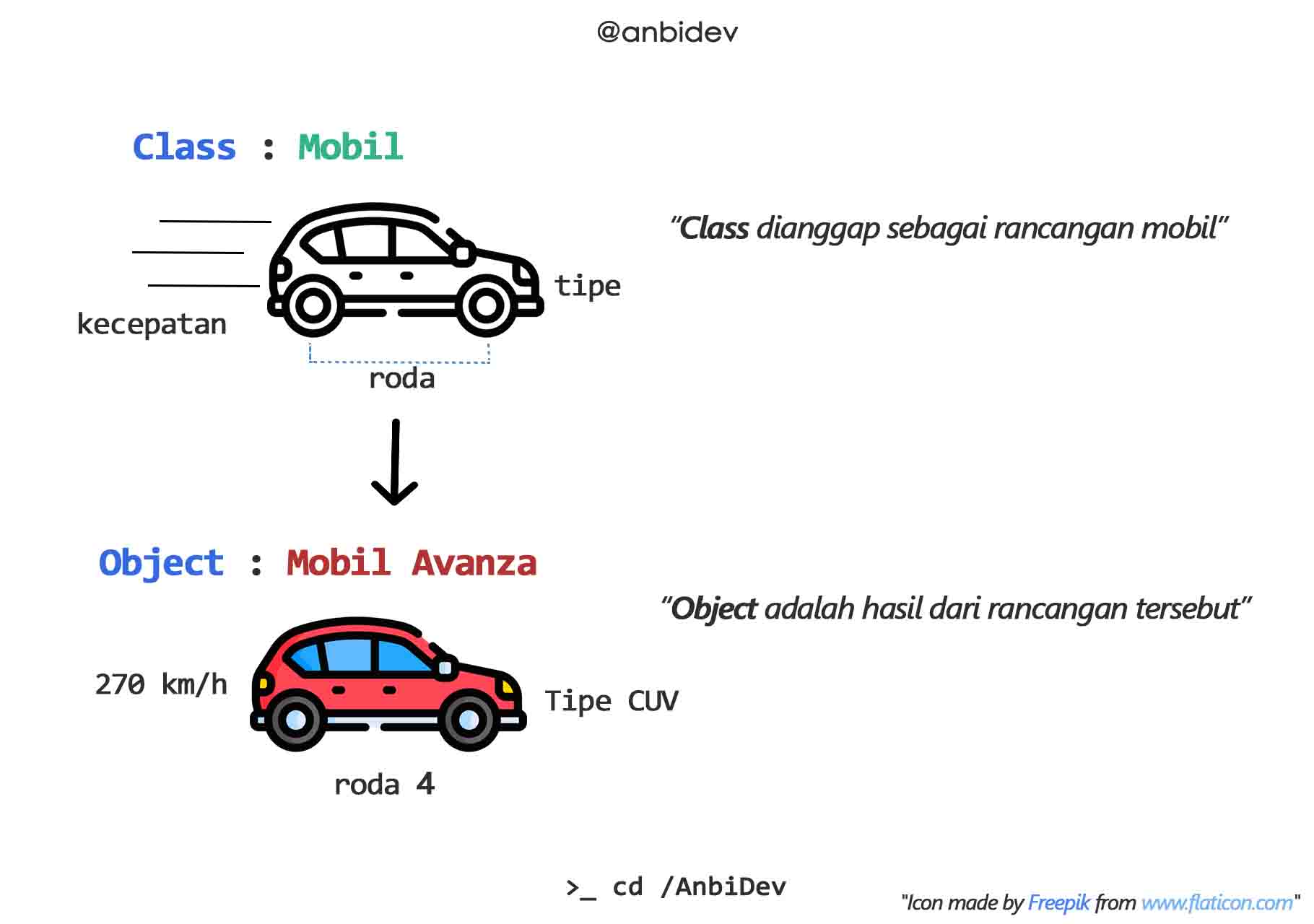
Seorang *programmer* merancang program perangkat lunak dengan mengatur informasi dan perilaku yang terkait bersama-sama ke dalam templat yang disebut kelas.

Kemudian objek individual dibuatkan atau dibuat dari *template*  kelas, benda-benda ini biasanya mewakili benda dunia nyata.

Seluruh program perangkat lunak dijalankan dengan menyuruh banyak objek berinteraksi dengan objek untuk membuat program yang lebih besar.

*Object-Oriented Programming* (OOP) adalah model bahasa pemrograman di mana program disusun di sekitar data, atau objek, bukan fungsi dan logika.

Objek dapat didefinisikan sebagai bidang data yang memiliki atribut dan perilaku yang *unique* (unik).



Contoh Analogi OOP

Contohnya seperti ini, anggap kita punya Class **mobil**. Ia memiliki attribute

* **roda**,
* **kecepatan** dan
* **tipe**.

Serta didalam mobil dia juga memiliki method :

* **melaju**(),
* **klakson**() dan
* **berbelok**().

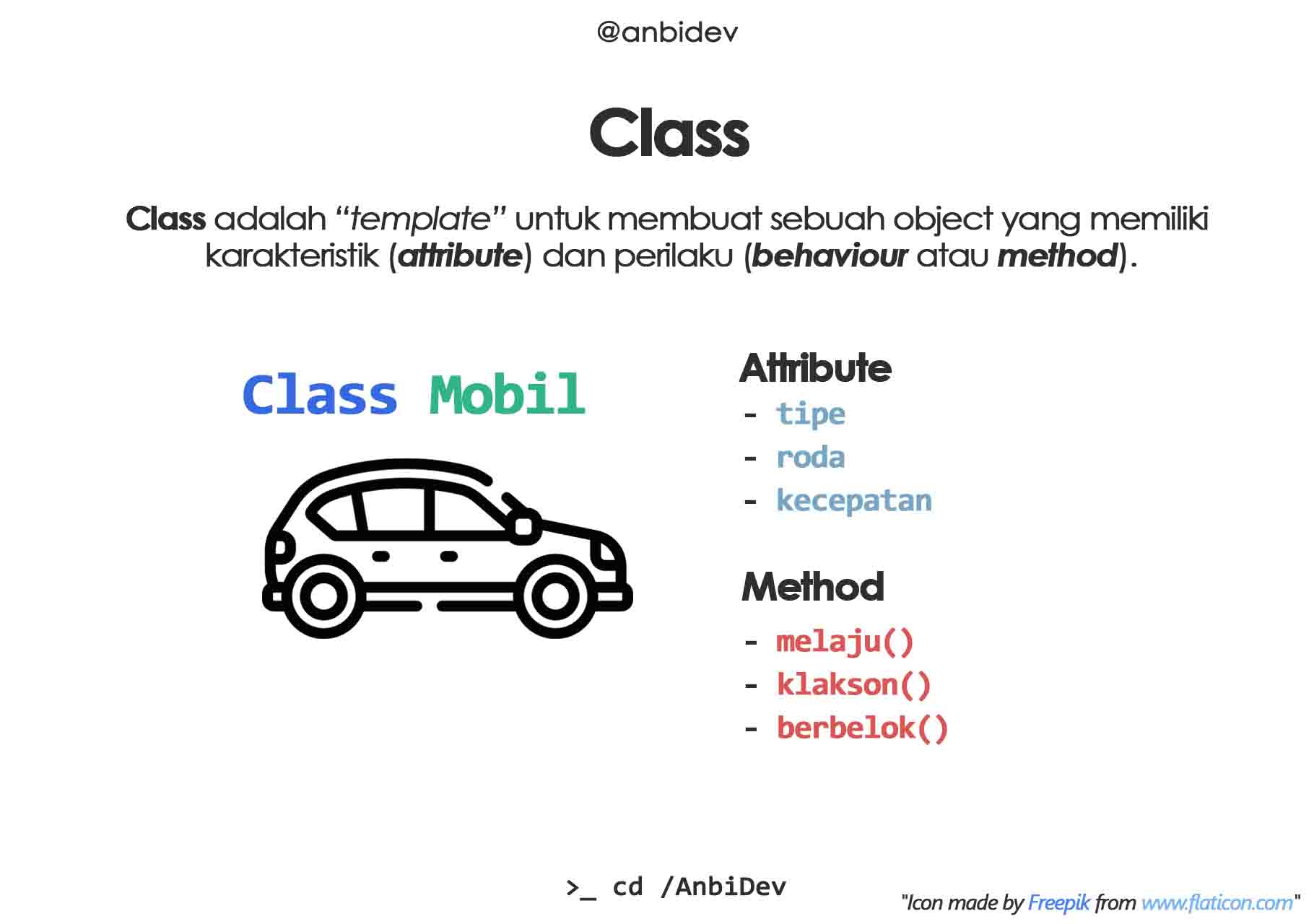
Jika Class dalam perumpamaannya adalah blueprint dari suatu kategori yang umum, maka object diumpamakan sebagai hal yang lebih spesifik. Jadi, class **mobil** tadi bisa membuat banyak objek yang lebih spesifik, misal mobilAvanza atau mobilFerari.

OOP karena merupakan objek, maka objek ini memiliki **class, atribut, method**. Mari kita simak apa class, atribut, method seperti dibawah ini.

## Apa itu Class?

**Class** adalah ‘cetakan’, ‘template’ atau ‘blueprint’ untuk membuat sebuah **object**. Class hanya sebagai kerangka dasar dari object. Sehingga nanti cetakan (class) tersebut dapat membuat berbagai macam objek untuk kebutuhan yang berbeda - beda.

Class diibaratkan sebagai kategori yang umum, seperti yang dijelaskan diatas. Anggaplah kita membuat Class bernama "mobil", didalam class tersebut terdapat karakteristik (attribute) dan perilaku (method) yang hanya dimiliki oleh **mobil**.



Gambar Analogi OOP class

Kita tahu bahwa mobil memiliki roda, tipe atau kecepatan mobil. Hal ini bisa kita sebut dengan attribute. Lalu kita juga tahu bahwa mobil juga memiliki fungsi untuk jalan melaju, klakson dan berbelok.

Dalam ilustrasi diatas, kita bisa menyimpulkan seperti apa bentuk class **mobil**. Dan selanjutnya, bagaimana bentuk class tersebut, jika ditranslasikan kedalam bentuk code java.

|  |
| --- |
| public class Mobil {    String type;  int roda;  Double kecepatan;    public Mobil(String type, int roda, Double kecepatan) {  this.type = type;  this.roda = roda;  this.kecepatan = kecepatan;  }    public void doMelaju(String kecepatan){  System.*out*.println("Melaju dengan kecepatan : "+kecepatan);  }    public void doKlakson(){  System.*out*.println("klakson");  }    public void doBelok(String arah){  System.*out*.println("Belok ke arah "+arah);  }  } |

## Apa itu Attribut ?

**Attribute** atau **properties** merupakan karakteristik dari sebuah

class. Attribute ini berupa suatu variabel didalam sebuah class. **Variabel** yang didefinisikan sebagai sebuah attribute juga disebut dengan variabel Global.

**Bagaimana Cara Deklarasi Attribut ?**

Cara mendeklarasikan sebuah attribute sama dengan cara mendeklarasikan sebuah variabel. Dalam java, variabel yang digunakan sebagai attribute biasanya didefinisikan dalam fungsi constructur (type data\_). Variabel **attribute** pada class disebut juga dengan instance (turunan) variable.

**Kenapa harus pakai constructor?**

karena instance variabel hanya bisa didefinisikan hanya dari sebuah method.

Tadi kita sudah sedikit menyinggung tentang *attribute* yang dimiliki oleh mobil (tipe, roda dan kecepatan). Dalam kode yang dibuat diawal.

|  |
| --- |
| String type;  int roda;  Double kecepatan; |

Pada kode diatas kita mendeklarasikan *attribute* dengan nama variabel “**type**,**roda**” bertype String dan variabel “**kecepatan**” bertype Double.

## Apa itu Construktor ?

Constructor merupakan suatu method yang akan memberikan nilai awal pada saat suatu objek dibuat. Pada saat program dijalankan, constructor akan langsung memberikan nilai awal pada saat perintah new, membuat suatu objek. Pada saat kita bekerja dengan constructor, hal mendasar yang perlu diperhatikan, **yaitu** :

* Nama Constructor sama dengan nama Class.
* Tidak ada return type yang diberikan kedalam Constructor Signature.
* Tidak ada return stetement, didalam tubuh constructor.

Jika kita lihat pada kode class **mobil** diatas, maka kodingan yang menunjukkan construktor adalah sebagai berikut :

|  |
| --- |
| public class Mobil {    String type;  int roda;  Double kecepatan;    public Mobil(String type, int roda, Double kecepatan) {  this.type = type;  this.roda = roda;  this.kecepatan = kecepatan;  }  …...  } |

Dapat kita lihat, nama kelas dan nama construktor nya sama , yaitu mobil. Dan tidak memiliki return nilai.

## Apa Itu Method/Fungsi ?

Method adalah fungsi yang didefinisikan dalam suatu **class**. Biasanya method memiliki hubungan dalam behaviour atau perilaku kelas tersebut. Dalam kasus class **mobil**, dapat mendefinisikan 3 method yaitu :

* **melaju,**
* **klakson**dan
* **berbelok.**

Contoh penulisan method dalam java seperti dibawah ini :

|  |
| --- |
| public class Mobil {  ….  public void doMelaju(String kecepatan){  System.*out*.println("Melaju dengan kecepatan : "+kecepatan);  }    public void doKlakson(){  System.*out*.println("klakson");  }    public void doBelok(String arah){  System.*out*.println("Belok ke arah "+arah);  }  ….  } |

Method diatas adalah jenis method **void,** artinya method yang tidak mengembalikan nilai. Adapun **jenis** metho pada ada java aad 2, yaitu :

* **void Method**
* **return Method**

**void** **method** adalah method yang tidak mengembalikan suatu nilai. Dilihat sekilas, void method dapat diidentifikasi dengan adanya kata kunci “void” di depan nama method. Contoh nya terlihat pada tabel kode program diatas yaitu **doMelaju, doKlakson, doBelok.**

**Terus, apa itu return method ?**

return method adalah method yang mengembalikan nilai. Method ini bisa diidentifikasi dengan adanya data type di depan nama method-nya dan kata kunci return di dalam method-nya. Contoh:

|  |
| --- |
| //Return Method  public double hitungKecepatan(Double kecepatan, int waktuDetik){  return kecepatan/waktuDetik;  } |

Method hitungKecepatan merupakan return method dengan type data double, ditandai dengan adanya return di body methodnya.

## Apa Itu Setter Getter ?

Method setter dan getter adalah dua method yang tugasnya untuk mengambil dan mengisi data ke dalam objek.

Dalam OOP kita sering mendengar istilah *encapsulation* (pembungkusan), di mana data dibungkus dengan modifier **private** agar tidak bisa diakses secara langsung dari luar class. Nanti akan kita bahas apa itu *encapsulation*. Contoh method setter dan getter jika mengacu pada class mobil pada variabel nya adalah sebagai berikut :

|  |
| --- |
| //Setter and Getter  public String getType() {  return type;  }    public void setType(String type) {  this.type = type;  }    public int getRoda() {  return roda;  }    public void setRoda(int roda) {  this.roda = roda;  }      public Double getKecepatan() {  return kecepatan;  }    public void setKecepatan(Double kecepatan) {  this.kecepatan = kecepatan;  } |

Pada potongan kode diatas, setiap method ditandai dengan adanya keyword **set** dan **get**. Misalnya setRoda dan getRoda. Artinya setRoda(“roda 1”), variabel roda akan berisi nilai “roda 1”. Jika String ambilRoda= getRoda(); , artinya varibale ambilRoda akan berisi nilai dari method getRoda();

**Contoh**

|  |
| --- |
| public class MainMobil {  public static void main(String[] args){  Mobil mobil= new Mobil();  mobil.setRoda(4);  System.*out*.println("isi method roda :"+mobil.getRoda());  }  } |

Maka outputnya adalah :

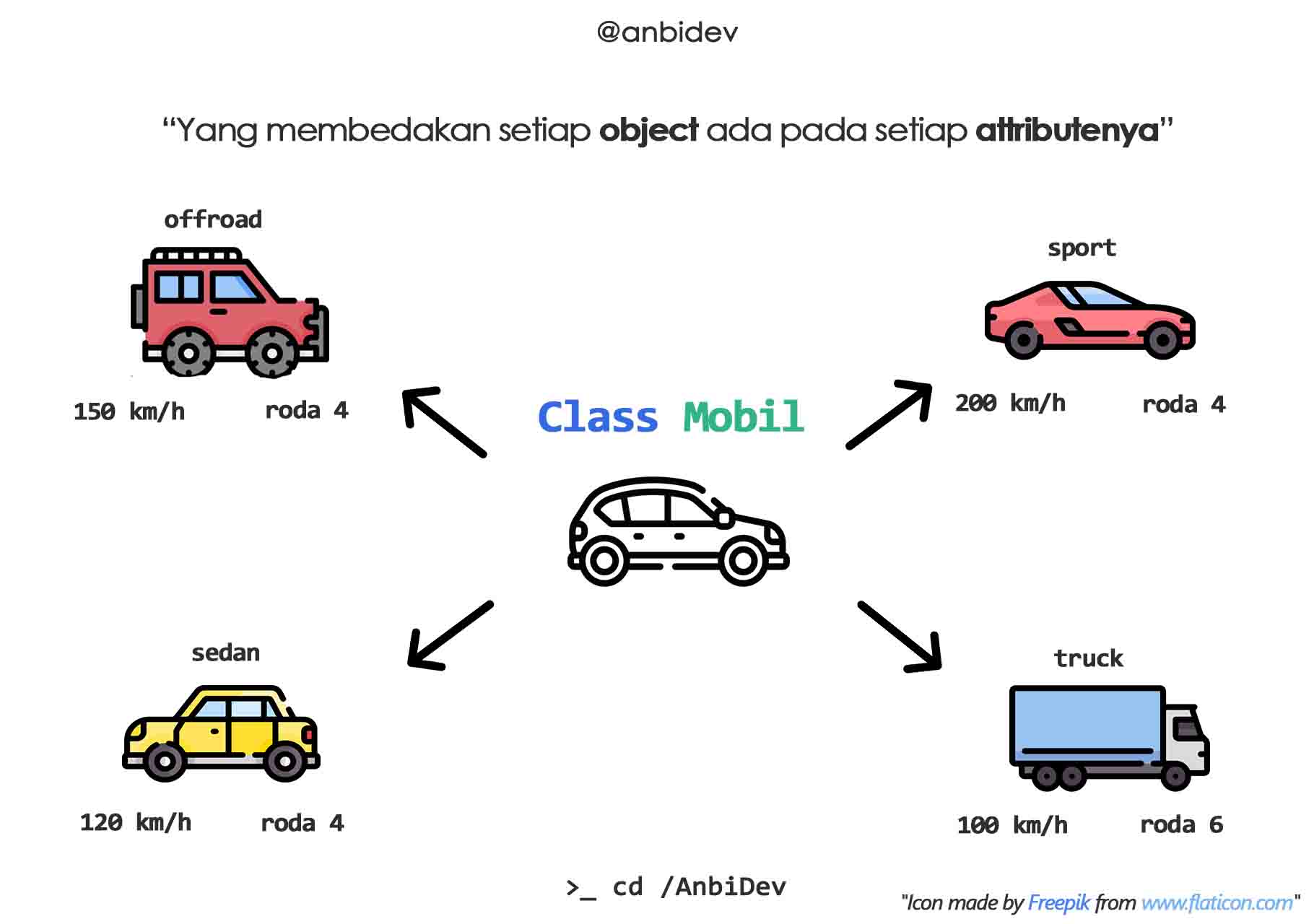
A screen shot of a computer

Description automatically generated

Gambar Output Method Roda

## Apa itu Object?

Object adalah instance atau representasi dari sebuah class. Jika class adalah sebuah cetakan, maka object adalah hasil dari cetakan tersebut. Kita dapat membuat berbagai object dengan class mobil yang tadi kita buat.

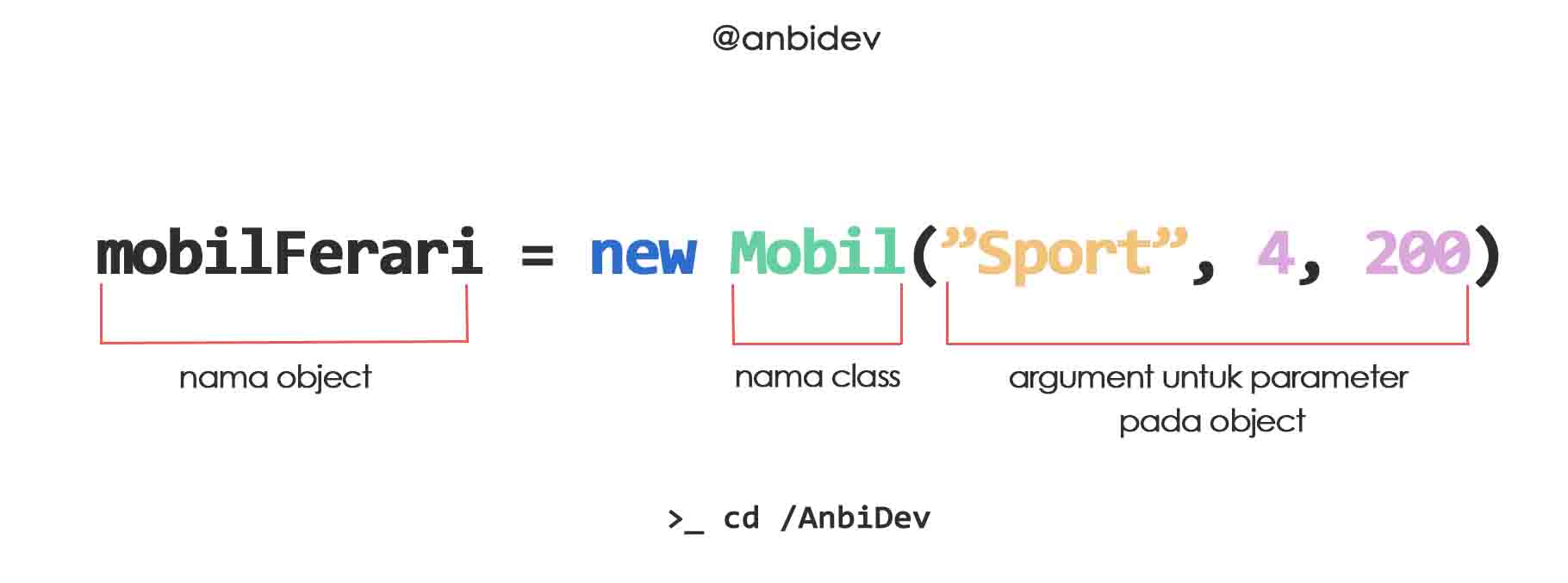


Gambar Objek Class Mobil

Cara mendeklarasikan object dari sebuah class pada java adalah dengan memanggil nama class beserta dengan parameter yang diberikan pada fungsi *constructor.*

**Cara memanggil Objek kelas ?**

|  |
| --- |
| Mobil mobilFerari = new Mobil("Sport", 4, 200.0);  Mobil mobilJeep = new Mobil("Offroad", 6, 150.0); |



Gambar Struktur Deklarasi Objek

Pada kode diatas membuat **2 object** pertama adalah object **mobilFerari** dan **mobilOffroad**. Kedua object tersebut dibuat dari class yang sama (Mobil) dan memiliki perilaku (*behaviour*) yang sama tapi dengan karakteristik (*attribute*) yang berbeda.

Jika di run, maka outputnya adalah :

A screen shot of a computer

Description automatically generated

Gambar Output Objek Class Mobil

## Kode Program Lengkap

Class Mobil

|  |
| --- |
| package com.example.demo.model;  //Class  public class Mobil {    //Variabel  public String type;  public int roda;  public Double kecepatan;    //Construktor  public Mobil(){};      public Mobil(String type, int roda, Double kecepatan) {  this.type = type;  this.roda = roda;  this.kecepatan = kecepatan;  }    //Method VOID  public void doMelaju(String kecepatan){  System.*out*.println("Melaju dengan kecepatan : "+kecepatan);  }    public void doKlakson(){  System.*out*.println("klakson");  }    public void doBelok(String arah){  System.*out*.println("Belok ke arah "+arah);  }    //Return Method  public double hitungKecepatan(Double kecepatan, int waktuDetik){  return kecepatan/waktuDetik;  }    //Setter and Getter  public String getType() {  return type;  }    public void setType(String type) {  this.type = type;  }    public int getRoda() {  return roda;  }    public void setRoda(int roda) {  this.roda = roda;  }    public Double getKecepatan() {  return kecepatan;  }    public void setKecepatan(Double kecepatan) {  this.kecepatan = kecepatan;  }  } |

Class MainMobil

|  |
| --- |
| public class MainMobil {  public static Mobil *objMobil* = new Mobil();  public static void main(String[] args){  Mobil mobilFerari = new Mobil("Sport", 4, 200.0);  Mobil mobilJeep = new Mobil("Offroad", 6, 150.0);  System.*out*.println("mobilFerari");  System.*out*.println("roda :"+mobilFerari.getRoda());  System.*out*.println("type :"+mobilFerari.getType());  System.*out*.println("kecepatan :"+mobilFerari.getKecepatan());  System.*out*.println("mobilJeep");  System.*out*.println("roda :"+mobilJeep.getRoda());  System.*out*.println("type :"+mobilJeep.getType());  System.*out*.println("kecepatan :"+mobilJeep.getKecepatan());    }    } |

## Cara Kerja OOP (Object-Oriented Programming)

**Langkah pertama**

dalam OOP yaitu mengidentifikasi semua objek yang ingin dimanipulasi oleh *programmer* dan bagaimana mereka saling berhubungan.

Praktiknya sering dikenal sebagai [*data modelling* atau pemodelan data](https://rifqimulyawan.com/blog/pengertian-data-modelling/).

**Langkah kedua**

Setelah suatu objek diketahui, itu kemudian digeneralisasikan sebagai *object class* (kelas objek) yang mendefinisikan jenis data yang dikandungnya dan urutan logika apa pun yang dapat memanipulasinya.

**Langkah ketiga**

Setiap urutan logika yang berbeda dikenal sebagai *method* dan objek dapat berkomunikasi dengan antarmuka yang didefinisikan dengan baik yang disebut *message*.

Sederhananya, OOP berfokus pada objek yang ingin dimanipulasi *developers* atau pengembang daripada logika yang diperlukan untuk memanipulasi mereka.

Pendekatan pemrograman ini sangat cocok untuk program yang besar, kompleks, dan diperbarui atau dipelihara secara aktif.

Karena pengorganisasian program berorientasi objek, metode ini juga kondusif untuk pengembangan kolaboratif di mana proyek dapat dibagi menjadi kelompok-kelompok.

## Manfaat OOP

Pemrograman berorientasi objek, seperti namanya, *Object-Oriented Programming* atau OOP mengacu pada bahasa yang menggunakan objek dalam pemrograman.

Pemrograman berorientasi objek bertujuan untuk mengimplementasikan entitas dunia nyata seperti pewarisan, persembunyian, polimorfisme dan lainnya dalam pemrograman.

Tujuan utama OOP adalah untuk menyatukan data dan fungsi yang beroperasi pada mereka sehingga tidak ada bagian lain dari kode yang dapat mengakses data ini kecuali fungsi itu.

Salah satu manfaat besar dan utama OOP (*Object-Oriented Programming*) adalah skalabilitas, dengan objek dan definisi tidak memiliki batasan yang terbatas.

Selain itu, pemisahan data dari metode mencegah masalah umum yang ditemukan dalam bahasa perangkat lunak linier yang lebih lama.

Jika bug ([baca pengertian bug disini](https://rifqimulyawan.com/blog/pengertian-bug/)) muncul dalam kode linier, bug itu dapat diterjemahkan melalui sistem dan membuat banyak kesalahan yang sulit dilacak.

Sebaliknya, program OOP, dengan pemisahan metode dan data, tidak rentan terhadap kesalahan yang dapat berkembang biak.

Selain itu, adapun beberapa keuntungan dan manfaat dari *Object-Oriented Programming* (OOP) yang harus kalian ketahui adalah sebagai berikut:

* OOP memodelkan hal-hal kompleks sebagai struktur sederhana yang dapat direproduksi.
* Kode OOP dapat digunakan kembali.
* OOP mencegah duplication atau duplikasi kode.
* OOP membuat memperbaiki bug lebih mudah.
* Memperbaiki kesalahan yang terkandung dalam kelas yang terstruktur dengan baik lebih mudah daripada menemukan kesalahan di banyak tempat dalam kode.
* OOP melindungi informasi melalui encapsulation (enkapsulasi). Data objek hanya dapat diakses melalui properti dan metode publik.
* OOP menggunakan abstraksi untuk membuat kompleksitas dapat dicerna.
* OOP mudah diajak bekerja sama dalam tim pengembang, class dapat diberikan kepada pengembang yang berbeda, yang memungkinkan banyak orang untuk membuat kode proyek secara bersamaan.
* Prinsip-Prinsip Dasar OOP

Dalam membahas apa itu OOP atau Object-Oriented Programming, adalah merupakan hal yang penting untuk juga menjelaskan tentang prinsip-prinsip dasarnya.

Ya, perlu teman-teman ketahui bahwa Object-Oriented Programming (OOP) atau Pemrograman Berorientasi Objek (PBO) ini didasarkan pada prinsip-prinsip berikut.

* Encapsulation
* Abstraction
* Inheritance
* Polymorphism

## Encapsulation

### Apa itu encapsulasi ?

*Encapsulation* atau pengkapsulan adalah konsep tentang pengikatan data atau metode berbeda yang disatukan atau “dikapsulkan” menjadi satu unit data.

Encapsulation di Java adalah proses untuk Menyembunyikan

dan melindungi secara menyeluruh (hak akses) dari *fungsionalitas* dan detail implementasinya. Yang dilindungi adalah class, variable, dan method dengan cara pemberian hak akses (modifier) yang terdiri dari empat tipe yaitu private, protected, public, dan kosong/tanpa modifier.

### Access Modifier



Gambar Modifier

**Keterangan** :

* Class = Class itu sendiri
* Package = Package Class yang sama
* Sub Class = Class anaknya
* Global = selain di class itu sendiri, package yang sama, dan sub classnya

**Berikut** **penjelasan**

* **Modifier Private** adalah modifier yang hanya bisa di akses oleh class itu sendiri.
* **Modifier Protected** adalah modifier yang hanya bisa di akses oleh class itu sendiri dan anak2nya (inheritance) serta class yang satu package dengannya.
* **Modifier Public** adalah modifier yang bisa di akses dari class manapun dan package manapun.
* **Tanpa modifier** adalah hak akses yang mirip sama protected tapi untuk class anak2nya (inheritance) harus satu package baru dapat akses.

### Penerapan enkapsulasi pada kode program ?

* + setter getter di java
  + model
  + entity

### Bagaimana Contoh ?

Contoh Pada Variabel memberikan akses **public**

|  |
| --- |
| public String type;  public int roda;  public Double kecepatan; |

Contoh pada Method memberikan akses **public**

|  |
| --- |
| public void doMelaju(String kecepatan){  System.*out*.println("Melaju dengan kecepatan : "+kecepatan);  }    public void doKlakson(){  System.*out*.println("klakson");  }    public void doBelok(String arah){  System.*out*.println("Belok ke arah "+arah);  } |

Contoh pada class memberikan **public**

|  |
| --- |
| public Mobil objMobil = new Mobil(); |

Contoh Setter dan Getter pada **Atribut**

|  |
| --- |
| public String getType() {  return type;  }    public void setType(String type) {  this.type = type;  }    public int getRoda() {  return roda;  }    public void setRoda(int roda) {  this.roda = roda;  }    public Double getKecepatan() {  return kecepatan;  }    public void setKecepatan(Double kecepatan) {  this.kecepatan = kecepatan;  } |

Jika di run,cara memanggil code seperti kode dibawah ini:

|  |
| --- |
| public static Mobil *objMobil* = new Mobil();  public static void main(String[] args) {  *objMobil*.setRoda(4);  System.*out*.println("isi method roda :" + *objMobil*.getRoda());  } |

Dapata dilihat pada kode *objMobil*.setRoda(4), penerapan encapsulasi. Objek tersebut dibungkus dalam method **setRoda**.

Maka outputnya adalah:

A black background with white text

Description automatically generated

Gambar Output Encapsulasi

### Kode Program Lengkap

Class Mobil

|  |
| --- |
| package com.example.demo.model;  //Class  public class Mobil {    //Variabel  public String type;  public int roda;  public Double kecepatan;    //Construktor  public Mobil(){};      public Mobil(String type, int roda, Double kecepatan) {  this.type = type;  this.roda = roda;  this.kecepatan = kecepatan;  }    //Method VOID  public void doMelaju(String kecepatan){  System.*out*.println("Melaju dengan kecepatan : "+kecepatan);  }    public void doKlakson(){  System.*out*.println("klakson");  }    public void doBelok(String arah){  System.*out*.println("Belok ke arah "+arah);  }    //Return Method  public double hitungKecepatan(Double kecepatan, int waktuDetik){  return kecepatan/waktuDetik;  }    //Setter and Getter  public String getType() {  return type;  }    public void setType(String type) {  this.type = type;  }    public int getRoda() {  return roda;  }    public void setRoda(int roda) {  this.roda = roda;  }    public Double getKecepatan() {  return kecepatan;  }    public void setKecepatan(Double kecepatan) {  this.kecepatan = kecepatan;  }  } |

Class MainMobil

|  |
| --- |
| public class MainMobil {  public static Mobil *objMobil* = new Mobil();  public static void main(String[] args){    *objMobil*.setRoda(4);  System.*out*.println("isi method roda :" + *objMobil*.getRoda());  }    } |

## Polymorphism

### Apa itu Polymorphism

Poly artinya banyak, morfisme artinya bentuk.

Polimorfisme (bahasa inggris polymorphism) adalah sebuah prinsip dalam biologi di mana oraganisme atau spesias dapat memiliki banyak bentuk atau tahapan (stages).

Polimorfisme dalam OOP adalah sebuah prinsip di mana class dapat memiliki banyak “bentuk” method yang berbeda-beda meskipun namanya sama.

“**Bentuk**” di sini dapat kita artikan: isinya berbeda, parameternya berbeda, dan tipe datanya berbeda.

### Jenis Polymorphism

Polimorfisme pada Java ada dua macam:

* Static Polymorphism (Polimorfisme statis);
* Dynamic Polymorphism (Polimorfisme dinamis).

Polimorfisme statis menggunakan method overloading

Sedangkan polimorfisme dinamis menggunakan method overriding. Method everriding telah dibahas di inheritance, silahkan refer kesana ya.

### Overloading

Overloading merupakan fungsionalitas yang memungkinkan kita untuk mendefinisikan beberapa method yang mempunyai nama sama tetapi memiliki body method yang berbeda

Pada method overloading perbedaan parameter mencakup :

* **Jumlah parameter**
* **Tipe data dari parameter**
* **Urutan dari tipe data parameter**

**Contoh**

|  |
| --- |
| public class ClassOVerloading {    String laptop(){  return "1";  }  String laptop(String a){  return a;  }    int laptop(int a){  return 1;  }  int laptop(int a,String b){  return 1;  }  }  Class ClassOVerloading memiliki dua method yang namanya sama, yakni laptop().  Tapi parameter dan tipe datanya berbeda..  ..dan juga isi atau rumus di dalamnya berbeda.  Inilah yang disebut polimorfisme statis. |

### Plimorfisme dinamis

biasanya terjadi saat kita menggunakan [pewarisan *(inheritance)*](https://www.petanikode.com/java-oop-inheritance/) dan implementasi ***interface***.

## Studi Kasus OOP – Enkapsulasi dan Polymorphism

### Studi Kasus: Sistem Pemesanan Makanan di Restoran

Sebuah restoran ingin meningkatkan efisiensi dalam pengelolaan pemesanan makanan mereka. Restoran ini menerima pesanan dari pelanggan yang datang langsung ke tempat atau melalui layanan pesan antar. Mereka ingin memiliki sistem yang dapat mengelola pesanan dengan baik, menghitung total harga pesanan, dan mencetak struk pembelian untuk pelanggan.

### Deskripsi Studi Kasus:

1. Restoran memiliki berbagai macam makanan yang tersedia di menu mereka, seperti Nasi Goreng, Mie Goreng, dan Ayam Goreng, dengan harga yang bervariasi.
2. Setiap pelanggan yang datang akan diminta untuk memberikan nama mereka.
3. Pelanggan dapat memesan satu atau lebih jenis makanan dari menu.
4. Setelah pelanggan selesai memesan, sistem akan menghitung total harga pesanan.
5. Pelanggan akan diminta untuk melakukan pembayaran dengan jumlah yang sesuai dengan total harga pesanan.
6. Sistem akan menghitung uang kembalian jika pelanggan membayar dengan jumlah yang lebih besar dari total harga pesanan.
7. Setelah pembayaran selesai, sistem akan mencetak struk pembelian yang berisi detail pesanan dan jumlah uang pembayaran.
8. Struk pembelian juga akan disimpan dalam file untuk referensi dan pencatatan.

### Tujuan:

1. Mengurangi kesalahan dalam mencatat pesanan pelanggan.
2. Meningkatkan efisiensi dalam proses pemesanan dan pembayaran.
3. Memberikan layanan yang lebih baik kepada pelanggan dengan menyediakan struk pembelian yang jelas dan terperinci.

### Implementasi

1. Menggunakan sistem OOP untuk memisahkan fungsi-fungsi terkait pemesanan makanan ke dalam kelas-kelas yang terdefinisi dengan baik.
2. Menerapkan konsep enkapsulasi untuk menyembunyikan detail implementasi dari kelas-kelas yang ada.
3. Menggunakan polimorfisme untuk memungkinkan penggunaan kelas Order dengan berbagai jenis item makanan.
4. Dengan implementasi yang tepat dari studi kasus ini, restoran dapat meningkatkan efisiensi dalam pengelolaan pesanan makanan mereka dan memberikan layanan yang lebih baik kepada pelanggan.

### Sample Output

A screenshot of a computer program

Description automatically generated

### Implementasi dari kode program sebelumnya

https://gitlab.com/rikialdi/batch7ex/-/blob/master/CH2-Sesi%201-OOP1/src/ch1/studicase/RestoranV5.java?ref\_type=heads

### Branch

https://gitlab.com/rikialdi/batch7ex/-/tree/master/CH2-Sesi%201-OOP1/src/ch2/pertemuan1OOP/studyCase?ref\_type=heads

# CH2 TOP 1 - Java OOP Part 2

## Topik

A screenshot of a computer

Description automatically generated

## Inheritance

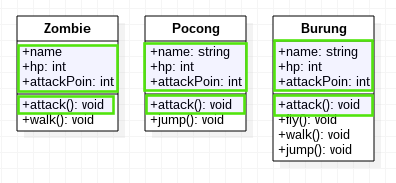
### Apa itu inheritance ?

Pewarisan/Penurunan adalah konsep pemrograman dimana sebuah class dapat 'menurunkan' property dan method yang dimilikinya kepada class lain

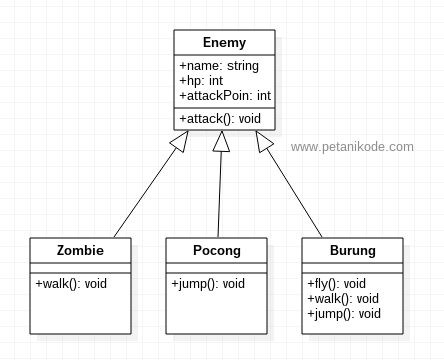
Konsep inheritance digunakan untuk memanfaatkan fitur ‘code reuse’ untuk menghindari duplikasi kode program.

### Konsep

Before : 3 method sama, 3 variabel sama



After: with inheritance



Ref : https://www.petanikode.com/java-oop-inheritance/

### Istilah Inheritance

* **Extends**  
  Keyword ini harus kita tambahkan pada definisi class yang menjadi subclass.
* **Superclass** Superclass digunakan untuk menunjukkan hirarki class yang berarti class dasar dari subclass/class anak.
* **Subclass** Subclass adalah class anak atau turunan secara hirarki dari superclass.
* **Super** Keyword ini digunakan untuk memanggil konstruktor dari superclass atau menjadi variabel yang mengacu pada superclass.
* **Methode Overriding** Pendefinisian ulang method yang sama pada subclass

### Method Inheritance

**Terdiri dari :**

* **Method overriding**

Method overriding adalah mendefinisikan kembali method yang sama, baik nama method maupun signature atau parameter yang diperlukan dalam subclass.

**Overriden** adalah method yang terdapat di super class

**Overriding** adalah method yang terdapat di sub class

* **Method** **overloading**

Method overloading adalah mendefinisikan method yang memiliki nama yang sama, tetapi dengan signature yang berbeda dalam definisi class yang sama.

### Contoh Penggunaan Extends

|  |
| --- |
| public class MobilAvanza extends Mobil{  public String deskripsi;  public String type;  } |

Class ***MobilAvanza*** adalah class anak , sedangkan class **Mobil** class induk. Itu ditandai dengan adanya keyword **extends** di class ***MobilAvanza.***

### Contoh penggunaan Method Overloading

|  |
| --- |
| public static String typeMobil(String typeMobil){  return typeMobil;  }    public static int typeMobil(int jumlahRoda){  return jumlahRoda;  }  public static void main(String[] args){  System.*out*.println("type mobil 1: "+*typeMobil*("Avanza"));  System.*out*.println("type mobil 2: "+*typeMobil*(4));  } |

Terlihat pada kode program diatas, typeMobil memiliki nama yang sama, namun prameter type datanya berbeda. Sehingga method ini dikatakan sebagai method overloading. Jika di run, outputnya seperti dibawah ini :

A black screen with white text

Description automatically generated

Gambar Output Method Overloading

### Contoh penggunaan Method Overriding

**Tujuan** overriding yaitu sub class ingin mengimplementasikan atau menerapkannya sendiri.

Class Mobil

|  |
| --- |
| public class Mobil {  ...  public void doMelaju(Double kecepatan){  System.*out*.println("Kecepatan Induk : "+kecepatan);  }  ...  } |

**Class MobilAvanza**

|  |
| --- |
| public class MobilAvanza extends Mobil {  ...  @Override  public void doMelaju(Double kecepatan) {  System.*out*.println("Kecepatan anak :"+kecepatan);  }  ...  } |

**Class MainMobil: cetak ke console**

|  |
| --- |
| public class MainMobil {  public static MobilAvanza *objMobil* = new MobilAvanza();    public static void main(String[] args) {  *objMobil*.doMelaju(100.0);  }    } |

**Jika di run, output nya adalah:**

A black screen with white text

Description automatically generated

**Gambar Output Method Overriding**

Seperti yang teman-teman lihat pada kode program diatas, kedua class(**Mobil** dan **MobilAvanza**) tersebut memiliki method void **doMelaju**(…). Class **MobilAvanza** memberikan implementasinya sendiri pada method **doMelaju** (…) tersebut.

Dengan kata lain meng-override method **doMelaju** (…) dari super class.

Sampai di sini tujuan overriding ini menjadi jelas, yaitu sub class ingin mengimplementasikan atau menerapkannya sendiri.

**Kode Lengkap**

**Class Mobil**

|  |
| --- |
| package com.example.demo.model.inheritance;  //Class  public class Mobil {    //Variabel  public String type;  public int roda;  public Double kecepatan;    //Construktor  public Mobil(){};      public Mobil(String type, int roda, Double kecepatan) {  this.type = type;  this.roda = roda;  this.kecepatan = kecepatan;  }    //Method VOID  public void doMelaju(Double kecepatan){  System.*out*.println("Kecepatan Induk : "+kecepatan);  }    public void doKlakson(){  System.*out*.println("klakson");  }    public void doBelok(String arah){  System.*out*.println("Belok ke arah "+arah);  }    //Return Method  public double hitungKecepatan(Double kecepatan, int waktuDetik){  return kecepatan/waktuDetik;  }    //Setter and Getter  public String getType() {  return type;  }    public void setType(String type) {  this.type = type;  }    public int getRoda() {  return roda;  }    public void setRoda(int roda) {  this.roda = roda;  }    public Double getKecepatan() {  return kecepatan;  }    public void setKecepatan(Double kecepatan) {  this.kecepatan = kecepatan;  }  } |

**Class Mobil Avanza**

|  |
| --- |
| package com.example.demo.model.inheritance;    public class MobilAvanza extends Mobil {  public String deskripsi;  public String type;    public String getDeskripsi() {  return deskripsi;  }    public void setDeskripsi(String deskripsi) {  this.deskripsi = deskripsi;  }    public String getType() {  return type;  }    public void setType(String type) {  this.type = type;  }    public static String typeMobil(String typeMobil) {  return typeMobil;  }    public static int typeMobil(int jumlahRoda) {  return jumlahRoda;  }    @Override  public void doMelaju(Double kecepatan) {  System.*out*.println("Kecepatan anak :"+kecepatan);  }      // public static void main(String[] args){  // System.out.println("type mobil 1: "+typeMobil("Avanza"));  // System.out.println("type mobil 2: "+typeMobil(4));  // }    } |

**Class MainMobil**

|  |
| --- |
| package com.example.demo.model.inheritance;    public class MainMobil {  public static MobilAvanza *objMobil* = new MobilAvanza();    public static void main(String[] args) {  *objMobil*.doMelaju(100.0);  }    } |

### Keuntungan Ketika Overriding Method

**Terdapat keuntungan** ketika meng-override method yaitu super class selaku pemilik overriden method, kodenya tidak perlu mengalami perubahan sama sekali, sementara itu di sisi lainnya sub class dapat mengimplementasikan kode tersebut sesuai dengan kebutuhan.

Bisa dibayangkan jika suatu superclass memiliki banyak subclass. Ketika sub class – sub class tersebut perlu untuk menggunakan method dari super class, mereka dapat menggunakannya karena sub class dapat menggunakan method super class pada konsep inheritance.

Dan jika mereka perlu mengimplementasikannya dengan berbeda mereka tinggal meng-override method dari super class tanpa sama sekali menyentuh kode method dari super class.

### Aturan-aturan Pada Overriding Method

Berikut ini adalah aturan-aturan pada overriding method yang perlu untuk anda ketahui.

* List dari argumen pada overriden dan overriding method harus benar-benar sama, termasuk tipe data dan urutan pada argumen tersebut.
* Acces modifier pada overriding method tidak bisa lebih ketat daripada access modifier di overriden method.
* Jika suatu class inherit dari class abstract atau mengimplementasikan interface, maka class tersebut harus meng-override semua method abstract kecuali class itu sendiri adalah class abstract.
* Method static, private dan final tidak bisa di-override. Namun method static bisa dideklarasikan ulang pada subclass.
* Overriding method harus memiliki tipe return (sub type) yang sama
* Jika method tidak bisa diwariskan, maka tidak bisa di-override
* Sub class yang berada pada package yang berbeda hanya dapat meng-override method non-final yang dideklarasikan sebagai public atau protected
* Suatu constructor tidak bisa di-override
* Overriding method tidak boleh melontarkan checked exception yang baru atau yang lebih luas.
* Untuk memanggil overriden method melalui sub class dapat menggunakan keyword super.

### Contoh penggunaan Keyword Super

Keyword super akan **mengacu pada superclass**.

Nah, fungsi dari keyword super itu sendiri adalah:

* Memanggil constructor dari superclass
* Memanggil method dari superclass
* Mengakses variabel instance super class

### Memanggil Conctructor Dari Superclass

Constructor digunakan untuk membentuk instance dari suatu class. Namun, tidak seperti properti dan method, constructor ini tidak diwariskan pada subclass.

Sehingga constructor dari subclass ini hanya dapat dipanggil melalui constructor dari sub class dengan menggunakan keyword super.

**Syntax nya adalah:**

* **super**()  
  Akan memanggil constructor tanpa argumen dari superclass
* **super**(**parameter**)  
  Memanggil constructor dari superclass yang memiliki argumen-argumen yang sesuai.

Pernyataan super() ataupun super(parameter) harus ditempatkan pertama kali di dalam constructor dari sub class.

Dan ini adalah satu-satunya cara untuk memanggil constructor super class melalui subclass.

**Contoh:**

|  |
| --- |
| public class MobilAvanza extends Mobil {  public String deskripsi;  public String type;    public MobilAvanza(String deskripsi, String type) {  super("type dari induk", 4, 100.0);  this.deskripsi = deskripsi;  this.type = type;  }  ...    public static void main(String[] args){  MobilAvanza obj = new MobilAvanza("Deksripsi Mobil Avanza", "Type Avanza");  System.*out*.println("Deksripsi :"+obj.getDeskripsi());  System.*out*.println("Type :"+obj.getType());  System.*out*.println("Kecepatan :"+obj.getKecepatan());  System.*out*.println("Rode :"+obj.getRoda());  }    } |

Terlihat dari kode program diatas, penggunaan kelas super pada construktor kelas induk. super("type dari induk", 4, 100.0); .

Jika di run, outputnya adalah :

A screen shot of a computer

Description automatically generated

Gambar output Keyword Super Construktor

### Memanggil Method Milik Superclass

Syntax

|  |
| --- |
| *super.method(parameters);* |

**Contoh**

**Class MobilAvanza**

|  |
| --- |
| ...  @Override  public void doMelaju(Double kecepatan) {  super.doMelaju(kecepatan);  System.*out*.println("Kecepatan anak :"+kecepatan);  }  ... |

**Class MainMobil**

|  |
| --- |
| public class MainMobil {  public static MobilAvanza *objMobil* = new MobilAvanza();    public static void main(String[] args) {  *objMobil*.doMelaju(100.0);  }    } |

**Class Mobil**

|  |
| --- |
| ...  public void doMelaju(Double kecepatan){  System.*out*.println("Kecepatan Induk : "+kecepatan);  }  ... |

Perhatikan pada class **MobilAvanza,** ada penambahan keywordsuper.doMelaju(kecepatan);**.** Artinya itu adalah memanggil method induknya.

**Maka Output**

A black screen with white text

Description automatically generated

Gambar Output Super Method

**Kode Lengkap**

**Mobil**

|  |
| --- |
| package com.example.demo.model.inheritance;  //Class  public class Mobil {    //Variabel  public String type;  public int roda;  public Double kecepatan;    //Construktor  public Mobil(){};      public Mobil(String type, int roda, Double kecepatan) {  this.type = type;  this.roda = roda;  this.kecepatan = kecepatan;  }    //Method VOID  public void doMelaju(Double kecepatan){  System.*out*.println("Kecepatan Induk : "+kecepatan);  }    public void doKlakson(){  System.*out*.println("klakson");  }    public void doBelok(String arah){  System.*out*.println("Belok ke arah "+arah);  }    //Return Method  public double hitungKecepatan(Double kecepatan, int waktuDetik){  return kecepatan/waktuDetik;  }    //Setter and Getter  public String getType() {  return type;  }    public void setType(String type) {  this.type = type;  }    public int getRoda() {  return roda;  }    public void setRoda(int roda) {  this.roda = roda;  }    public Double getKecepatan() {  return kecepatan;  }    public void setKecepatan(Double kecepatan) {  this.kecepatan = kecepatan;  }  } |

**MobilAvanza**

|  |
| --- |
| package com.example.demo.model.inheritance;    public class MobilAvanza extends Mobil {  public String deskripsi;  public String type;    public MobilAvanza(){};    public MobilAvanza(String deskripsi, String type) {  super("type dari induk", 4, 100.0);  this.deskripsi = deskripsi;  this.type = type;  }    public String getDeskripsi() {  return deskripsi;  }    public void setDeskripsi(String deskripsi) {  this.deskripsi = deskripsi;  }    public String getType() {  return type;  }    public void setType(String type) {  this.type = type;  }    public static String typeMobil(String typeMobil) {  return typeMobil;  }    public static int typeMobil(int jumlahRoda) {  return jumlahRoda;  }    @Override  public void doMelaju(Double kecepatan) {  super.doMelaju(kecepatan);  System.*out*.println("Kecepatan anak :"+kecepatan);  }      public static void main(String[] args){  MobilAvanza obj = new MobilAvanza("Deksripsi Mobil Avanza", "Type Avanza");  System.*out*.println("Deksripsi :"+obj.getDeskripsi());  System.*out*.println("Type :"+obj.getType());  System.*out*.println("Kecepatan :"+obj.getKecepatan());  System.*out*.println("Rode :"+obj.getRoda());  }    } |

**MainMobil**

|  |
| --- |
| package com.example.demo.model.inheritance;    public class MainMobil {  public static MobilAvanza *objMobil* = new MobilAvanza();    public static void main(String[] args) {  *objMobil*.doMelaju(100.0);  }    } |

### Digunakan Untuk Mengacu Secara Langsung Pada Variabel Instance Milik Super Class

Anda dapat menggunakan keyword super untuk langsung mengakses data member atau field dari super class. Dan ini digunakan jika super class dan sub class memiliki field-field yang sama.

**Contoh**

**Class MobilAvanza**

|  |
| --- |
| ...  public void superVariabel(String type) {  System.*out*.println("Type Mobil dari Anak " + type);  System.*out*.println("Type Mobil dari induk " + super.type);  }  ... |

**Class** **MainMobil**

|  |
| --- |
| public class MainMobil {  public static MobilAvanza *objMobil* = new MobilAvanza();    public static void main(String[] args) {  // objMobil.doMelaju(100.0);  *objMobil*.superVariabel("Avanza");  }    } |

**Output**:

A black background with white text

Description automatically generated

Gambar Output Super Variabel Instance

Dapat dilihat ouput dari indunya adalah null, karena dengan menggunakan keyword **super**, akan memanggil variabel dari class induknya.

**Kode Program lengkap**

**Class Mobil**

|  |
| --- |
| package com.example.demo.model.inheritance;  //Class  public class Mobil {    //Variabel  public String type;  public int roda;  public Double kecepatan;    //Construktor  public Mobil(){};      public Mobil(String type, int roda, Double kecepatan) {  this.type = type;  this.roda = roda;  this.kecepatan = kecepatan;  }    //Method VOID  public void doMelaju(Double kecepatan){  System.*out*.println("Kecepatan Induk : "+kecepatan);  }    public void doKlakson(){  System.*out*.println("klakson");  }    public void doBelok(String arah){  System.*out*.println("Belok ke arah "+arah);  }    //Return Method  public double hitungKecepatan(Double kecepatan, int waktuDetik){  return kecepatan/waktuDetik;  }    //Setter and Getter  public String getType() {  return type;  }    public void setType(String type) {  this.type = type;  }    public int getRoda() {  return roda;  }    public void setRoda(int roda) {  this.roda = roda;  }    public Double getKecepatan() {  return kecepatan;  }    public void setKecepatan(Double kecepatan) {  this.kecepatan = kecepatan;  }  } |

**Class MobilAvanza**

|  |
| --- |
| package com.example.demo.model.inheritance;    public class MobilAvanza extends Mobil {  public String deskripsi;  public String type;    public MobilAvanza(){};    public MobilAvanza(String deskripsi, String type) {  super("type dari induk", 4, 100.0);  this.deskripsi = deskripsi;  this.type = type;  }    public String getDeskripsi() {  return deskripsi;  }    public void setDeskripsi(String deskripsi) {  this.deskripsi = deskripsi;  }    public String getType() {  return type;  }    public void setType(String type) {  this.type = type;  }    public static String typeMobil(String typeMobil) {  return typeMobil;  }    public static int typeMobil(int jumlahRoda) {  return jumlahRoda;  }    @Override  public void doMelaju(Double kecepatan) {  super.doMelaju(kecepatan);  System.*out*.println("Kecepatan anak :"+kecepatan);  }    public void superVariabel(String type) {  System.*out*.println("Type Mobil dari Anak " + type);  System.*out*.println("Type Mobil dari induk " + super.type);  }        public static void main(String[] args){  MobilAvanza obj = new MobilAvanza("Deksripsi Mobil Avanza", "Type Avanza");  System.*out*.println("Deksripsi :"+obj.getDeskripsi());  System.*out*.println("Type :"+obj.getType());  System.*out*.println("Kecepatan :"+obj.getKecepatan());  System.*out*.println("Rode :"+obj.getRoda());      }    } |

**Class MainMobil**

|  |
| --- |
| package com.example.demo.model.inheritance;    public class MainMobil {  public static MobilAvanza *objMobil* = new MobilAvanza();    public static void main(String[] args) {  // objMobil.doMelaju(100.0);  *objMobil*.superVariabel("Avanza");  }    } |

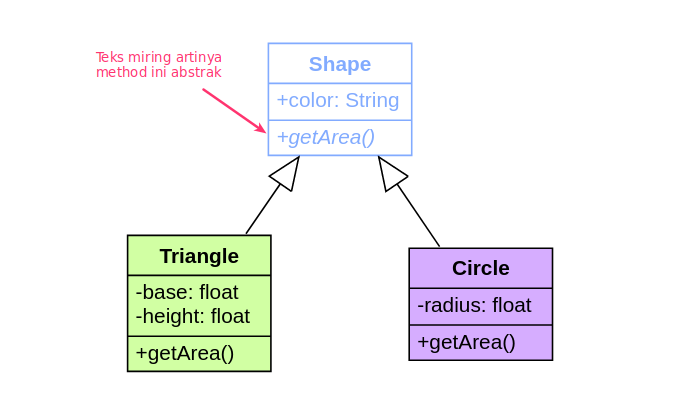
## Abstraction

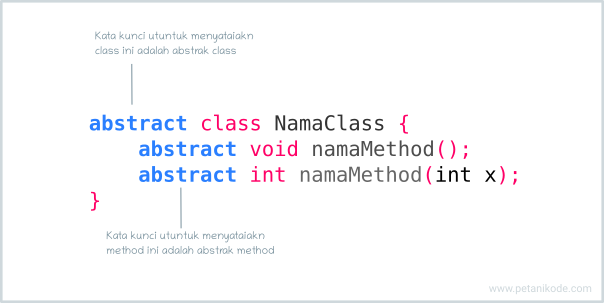
Class abstrak adalah class yang masih dalam bentuk abstrak. Karena bentuknya masih abstrak, dia tidak bisa dibuat langsung menjadi objek.

Sebuah class agar dapat disebut class abstrak setidaknya **memiliki** satu atau lebih **method abstrak**.

Method abstrak itu adalah method yang tidak memiliki isi. Cuma ada nama saja

### Konsep





Ref : https://www.petanikode.com/java-oop-abstract/

### Contoh 1

#### absstack

|  |
| --- |
| // Abstract class untuk merepresentasikan kendaraan abstract class Vehicle {  // Atribut  String brand;  int year;   // Constructor  public Vehicle(String brand, int year) {  this.brand = brand;  this.year = year;  }   // Abstract method untuk menghitung biaya servis kendaraan  abstract double calculateServiceCost(); } |

#### Subclass

|  |
| --- |
| // Subclass yang mewarisi abstract class Vehicle class Car extends Vehicle {  // Atribut tambahan untuk mobil  int mileage; // Jarak tempuh dalam kilometer   // Constructor  public Car(String brand, int year, int mileage) {  super(brand, year);  this.mileage = mileage;  }   // Implementasi dari abstract method calculateServiceCost()  @Override  double calculateServiceCost() {  // Contoh perhitungan biaya servis berdasarkan jarak tempuh  double costPerKm = 0.1; // Rupiah per kilometer  return costPerKm \* mileage;  } } |

#### Subclass 2

|  |
| --- |
| // Subclass lain yang mewarisi abstract class Vehicle class Motorcycle extends Vehicle {  // Atribut tambahan untuk sepeda motor  boolean hasHelmet; // Apakah memiliki helm atau tidak   // Constructor  public Motorcycle(String brand, int year, boolean hasHelmet) {  super(brand, year);  this.hasHelmet = hasHelmet;  }   // Implementasi dari abstract method calculateServiceCost()  @Override  double calculateServiceCost() {  // Biaya servis motor tetap  double fixedCost = 50000; // Rupiah  return fixedCost;  } } |

#### Main class

|  |
| --- |
| public class Main {  public static void main(String[] args) {  // Membuat objek mobil  Car myCar = new Car("Toyota", 2020, 20000);   // Menghitung biaya servis mobil  double carServiceCost = myCar.calculateServiceCost();  System.*out*.println("Biaya servis mobil: " + carServiceCost);   // Membuat objek sepeda motor  Motorcycle myMotorcycle = new Motorcycle("Honda", 2019, true);   // Menghitung biaya servis sepeda motor  double motorcycleServiceCost = myMotorcycle.calculateServiceCost();  System.*out*.println("Biaya servis sepeda motor: " + motorcycleServiceCost);  } } |

### Contoh 2

|  |
| --- |
| // ini abstrak method  void sayHello();  // ini bukan abstrak method karena  // punya implementasi di body method  void greeting(){  System.out.println("Hello Java");  } |

### Mengapa Harus Pakai Class Abstrak?

Class abstrak memang belum bisa digunakan secara langsung.

Karena itu, agar class abstrak dapat digunakan, maka ia harus dibuat bentuk konkritnya.

Cara membuat class abstrak menjadi konkrit adalah dengan membuat implementasi dari method-method yang masih abstrak.

Ini bisa kita lakukan dengan [pewarisan (inheritance)](https://www.petanikode.com/java-oop-inheritance/).

Class abstrak biasanya digunakan sebagai class induk dari class-class yang lain. Class anak akan membuat versi konkrit dari class abstrak.

**Contoh**

|  |
| --- |
| abstract class abstraction {  public abstract void mData(String name, String nim);    public void mBiodata(){  System.*out*.println("biodata saya");  }  } |

Berikut ini adalah contoh class yang menggunakan Abstraction . terdapat 2 buat method, yaitu mData dan mBiodata. Pada mData terdapat kata abstract itu menunjukan bahwa method itu merupakan method abstract dan harus di override pada class turunanya.

|  |
| --- |
| public class Implementasi extends abstraction {  @Override  public void mData(String name, String nim) {  System.*out*.println("nama="+name);  System.*out*.println("nim="+nim);  }  } |

ini dia kelas turunan dari class abstrac diatas. Dikarenakan class ini merupakan extends dari class abstrac Biodata makan method abstract pun di implemetasikan di class ini.

|  |
| --- |
| public class ClassMain {  public static void main(String[] args) {  Implementasi imp = new Implementasi();  String nama ="nama saya";  String nim ="1234";    imp.mBiodata();  imp.mData(nama,nim);  }  }  output  A screen shot of a computer  Description automatically generated |

Pada  imp.mData(nama,nim**)**, berikut adalah untuk memanggil method mData pada class imp atau Implemetasi beserta parameter sesuia kebutuhannya.

Secara singkatnya, ketika kita menggunakan abstraction pada sebuah class dan method nya, dan ketika kita turunkan . Maka class turunanya **wajib menyertakan method abstrac melalui override.**

## Keyword static dan final

### Best practice

Penggunaan kata kunci static dan final dalam pemrograman berorientasi objek memiliki beberapa best practice yang dapat membantu dalam merancang kode yang bersih, efisien, dan mudah dipelihara. Berikut adalah beberapa best practice yang umum:

**Keyword static:**

* **Menggunakan variabel static untuk data bersama:** Variabel static digunakan untuk menyimpan data yang bersama-sama digunakan oleh semua instance dari kelas. Ini termasuk variabel konstan, penghitung, dan data yang dibagi di seluruh aplikasi.
* **Pemanggilan metode static dengan cara yang konsisten**: Metode static sebaiknya dipanggil menggunakan nama kelas, bukan instance objek. Ini membuatnya jelas bahwa metode tersebut adalah metode static.
* **Pentingnya pengujian:** Metode static yang kompleks sering kali sulit untuk diuji. Membungkus logika tersebut dalam kelas utilitas terpisah atau membuat metode menjadi non-static dan mengirim instance sebagai parameter dapat memudahkan pengujian.

**Keyword final:**

* **Menggunakan variabel final untuk nilai yang tidak dapat diubah:** Variabel final digunakan untuk menyatakan bahwa nilai variabel tersebut tidak akan berubah setelah diinisialisasi. Ini membantu dalam menjaga keamanan kode dan meminimalkan bug yang disebabkan oleh perubahan nilai yang tidak disengaja.
* **Penggunaan konstanta:** Variabel final sering digunakan untuk mendefinisikan konstanta dalam kode, seperti konstanta matematika atau konstanta yang digunakan di beberapa bagian kode.
* **Pertimbangkan penggunaan final untuk metode dan kelas:** Selain variabel, kata kunci final juga dapat diterapkan pada metode dan kelas. Menandakan metode atau kelas sebagai final berarti mereka tidak dapat diubah atau diwarisi oleh kelas lain. Ini sering digunakan untuk alasan keamanan atau kinerja.
* **Perluasan Immutability**: Menggunakan final pada kelas atau objek yang tidak dapat diubah (immutability) dapat membantu dalam menjaga keamanan dan konsistensi data. Hal ini penting terutama dalam lingkungan multi-threading.

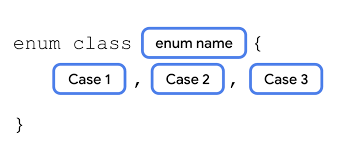
Perlindungan terhadap Override: Menandai metode atau kelas sebagai final dapat melindungi mereka dari penggunaan yang tidak disengaja atau override yang tidak diinginkan, terutama saat bekerja dalam tim besar atau dalam konteks pemrograman kerangka kerja.

### Contoh :

|  |
| --- |
| public class Mobil {  private static int *jumlahMobilDibuat* = 0; // contoh penggunaan static   private String roda;  private String type;  private Double kecepatan;  public static final String *DEFAULT\_RODA* = "4"; // contoh penggunaan final   public Mobil(String roda, String type, Double kecepatan) {  this.roda = roda;  this.type = type;  this.kecepatan = kecepatan;  *jumlahMobilDibuat*++; // setiap kali sebuah mobil dibuat, jumlahMobilDibuat akan bertambah  }   public static int getJumlahMobilDibuat() {  return *jumlahMobilDibuat*; // metode static untuk mengakses variabel static  }   public String getRoda() {  return roda;  }   public void setRoda(String roda) {  this.roda = roda;  }   public String getType() {  return type;  }   public void setType(String type) {  this.type = type;  }   public Double getKecepatan() {  return kecepatan;  }   public void setKecepatan(Double kecepatan) {  this.kecepatan = kecepatan;  } } |

## ENUM

### Konsep



### Penerapan Enum

* **Level Jabatan Karyawan:** Sebuah perusahaan menggunakan enum untuk mengelompokkan level jabatan karyawan, seperti MANAGER, SUPERVISOR, STAFF, dan INTERN. Ini memungkinkan perusahaan untuk dengan mudah mengelola hak akses, gaji, dan kebijakan lainnya berdasarkan level jabatan.
* **Status Pemesanan**: Sebuah aplikasi e-commerce menggunakan enum untuk menyimpan status pemesanan, seperti DIPROSES, DIKIRIM, DITERIMA, dan DIBATALKAN. Ini membantu dalam melacak dan memantau status pemesanan pelanggan secara efisien.
* **Hari Kerja:** Sebuah sistem manajemen kehadiran menggunakan enum untuk mewakili hari kerja dalam seminggu, seperti SENIN, SELASA, RABU, dan seterusnya. Ini memudahkan dalam pengaturan jadwal kerja karyawan dan pengelolaan cuti.
* **Tingkat Prioritas Tugas:** Sebuah aplikasi manajemen proyek menggunakan enum untuk menetapkan tingkat prioritas tugas, seperti TINGGI, SEDANG, dan RENDAH. Ini membantu tim dalam menetapkan urutan penyelesaian tugas.
* **Jenis Produk:** Sebuah toko online menggunakan enum untuk mengelompokkan jenis produk yang dijual, seperti ELEKTRONIK, FASHION, BUKU, dan PERALATAN RUMAH TANGGA. Ini memudahkan pelanggan untuk menavigasi dan menemukan produk yang mereka cari.
* **Status Pekerjaan:** Sebuah platform perekrutan menggunakan enum untuk melacak status pekerjaan yang sedang diiklankan, seperti TERBUKA, DITUTUP, DIISI, dan DITUNDA. Ini membantu dalam menyediakan informasi yang jelas kepada pencari kerja.
* **Kategori Pelanggan:** Sebuah perusahaan menggunakan enum untuk mengelompokkan kategori pelanggan, seperti INDIVIDU, BISNIS KECIL, BISNIS MENENGAH, dan KORPORAT. Ini membantu dalam menyediakan layanan yang disesuaikan dengan kebutuhan dan preferensi setiap kategori pelanggan.

### Contoh

Mari kita tambahkan contoh penggunaan **enum** dalam kelas **Mobil**. Kita akan membuat **enum** untuk tipe roda dan jenis mobil:

|  |
| --- |
| // Enum untuk tipe roda enum Roda {  *DUA*, *EMPAT*; // Daftar nilai yang mungkin }  // Enum untuk jenis mobil enum Type {  *SEDAN*, *SUV*, *HATCHBACK*; // Daftar nilai yang mungkin }  public class Mobil {  private Roda roda; // Menggunakan enum Roda  private Type type; // Menggunakan enum Type  private Double kecepatan;   public Mobil(Roda roda, Type type, Double kecepatan) {  this.roda = roda;  this.type = type;  this.kecepatan = kecepatan;  }   // Getter dan setter dihilangkan untuk kejelasan   @Override  public String toString() {  return "Mobil{" +  "roda=" + roda +  ", type=" + type +  ", kecepatan=" + kecepatan +  '}';  }   public static void main(String[] args) {  Mobil mobil1 = new Mobil(Roda.*EMPAT*, Type.*SEDAN*, 100.0);  Mobil mobil2 = new Mobil(Roda.*DUA*, Type.*SUV*, 120.0);   System.*out*.println("Mobil 1: " + mobil1);  System.*out*.println("Mobil 2: " + mobil2);  } } |

### Best practice ENUM

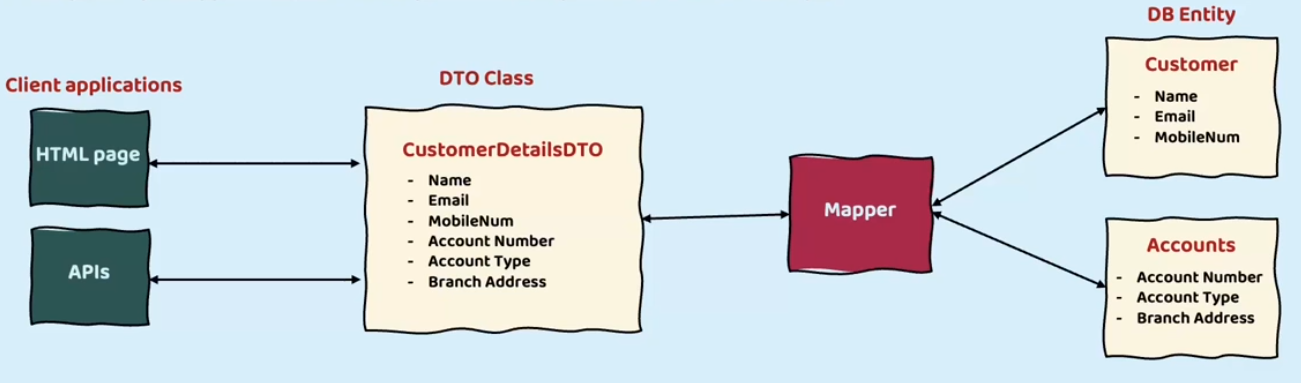
* **Gunakan enum untuk menyimpan konstanta:** Enum merupakan cara yang baik untuk menyimpan konstanta dalam aplikasi Anda, terutama jika konstanta-konstanta tersebut berkaitan satu sama lain atau memiliki hubungan logis.
* **Hindari penggunaan enum untuk tipe data yang sering berubah:** Enum cocok untuk tipe data yang stabil dan tidak berubah-ubah. Jika Anda memiliki tipe data yang mungkin berubah di masa depan, pertimbangkan untuk menggunakan struktur data lain yang lebih fleksibel.
* **Gunakan enum untuk menyederhanakan kode:** Enum memungkinkan Anda untuk menyederhanakan kode dengan mengelompokkan konstanta-konstanta terkait dalam satu tempat. Hal ini membuat kode lebih mudah dibaca dan dipelihara.
* **Gunakan enum dengan bijak untuk pengkodean yang aman**: Enum dapat membantu dalam membuat kode Anda lebih aman dengan membatasi nilai yang mungkin untuk suatu variabel. Ini dapat membantu dalam menghindari kesalahan pada saat runtime.
* **Jangan gunakan enum untuk data yang besar atau tidak terbatas:** Enum cocok untuk kumpulan data yang relatif kecil dan terbatas. Jika Anda memiliki data yang besar atau tidak terbatas, pertimbangkan penggunaan struktur data lain seperti array atau koleksi.
* **Gunakan metode dan atribut dalam enum**: Anda dapat menambahkan metode dan atribut ke dalam enum untuk memberikan lebih banyak fungsionalitas. Ini berguna jika Anda perlu melampirkan perilaku tertentu ke nilai-nilai enum Anda.
* **Pertimbangkan penggunaan nilai tambahan dalam enum:** Kadang-kadang, Anda mungkin ingin melampirkan nilai tambahan ke setiap nilai enum. Anda dapat menggunakan konstruktor enum untuk mencapai ini.
* **Gunakan enum sebagai parameter**: Enum dapat digunakan sebagai parameter dalam metode atau konstruktor untuk membuat kode Anda lebih jelas dan lebih mudah dipahami.
* **Berikan nama yang deskriptif untuk nilai enum:** Berikan nama yang deskriptif untuk setiap nilai enum untuk membuat kode lebih mudah dibaca dan dipahami oleh orang lain.

## DTO

### Apa itu DTO?

DTO (Data Transfer Object) adalah pola desain yang umum digunakan dalam pemrograman berorientasi objek untuk mentransfer data antar komponen aplikasi. Berikut adalah contoh penerapan DTO dalam dunia kerja:

### Konsep



Referensi : https://medium.com/@m.agungmahardika12/data-transfer-object-dto-patterns-in-spring-boot-application-156fe7460141

### Best Practices

* **Perancangan DTO yang Jelas:** Desain DTO dengan atribut yang jelas dan spesifik untuk setiap tujuan penggunaannya. Pastikan DTO hanya menyertakan data yang diperlukan untuk tujuan tertentu dan hindari memasukkan informasi yang tidak relevan.
* **Immutability:** Pertimbangkan untuk membuat DTO immutable (tidak dapat diubah setelah pembuatan). Ini membantu mencegah perubahan tak terduga dalam nilai DTO yang dapat menyebabkan efek samping yang tidak diinginkan.
* **Serialisasi:** Pastikan DTO mendukung serialisasi dengan baik. Ini penting jika Anda perlu mentransfer DTO melalui jaringan atau menyimpannya dalam penyimpanan persisten seperti database.
* **Penggunaan Atribut yang Cocok:** Gunakan tipe data yang sesuai untuk atribut DTO. Pastikan tipe data yang digunakan memenuhi kebutuhan bisnis dan mempertimbangkan faktor seperti ukuran dan performa data.

## SOLID

### Apa itu solid ?

# Gerbace Collector

Garbage Collection adalah mekanisme Java untuk menghapus suatu objek dari memori tanpa perlu dideklarasikan secara eksplisit dalam program. Berfungsi untuk meningkatkan Management Memory yaitu kita dapat menghemat penggunaan memori, java akan menghapus sebuah objek yang tidak diperlukan atau tidak di referensikan lagi, ruang memori pada objek tersebut akan dimanfaatkan untuk keperluan lain sehingga tidak ada pemborosan memori. Garbage Collection merupakan salah satu mekanisme dari fitur JVM (Java Virtual Machine). Ini adalah salah satu kelebihan dari bahasa pemrograman Java.

Pada materi berikut ini kita akan belajar mengenai fungsi Garbage Collection pada Java, dengan contoh program sederhana yang menggambarkan bagaimana Garbage Collection tersebut berfungsi.

Sebagai seorang programmer yang handal, aplikasi atau program yang kita buat harus memiliki penggunaan memori yang kecil atau seminimal mungkin, karena dengan begitu, program yang kita buatpun menjadi lebih handal juga.  
  
Salah satu kelebihan dari pemrograman Java adalah kita dapat menigkatkan efisiensi pada memori, dengan membuang objek yang tidak terpakai atau tidak direferensikan, ini akan secara otomatis dikontrol oleh JVM (Java Virtual Machine).  
  
Untuk melihat bagaimana cara kerja Garbage Collection pada java, kita dapat menambahkan codingan, seperti method finalize() dan juga System.gc(). Contoh Sederhananya seperti berikut ini:

|  |
| --- |
| public class GarbaceCollectorClass {  //Method ini akan ditampilkan jika Garbage Collection bekerja  public void finalize(){  System.*out*.println("Objek Yang Tidak Terpakai Sudah Dibersihkan:");  }    public static void main(String[] args){  Runtime RT = Runtime.*getRuntime*();  System.*out*.println("Jumlah Memori Awal: "+RT.totalMemory());    GarbaceCollectorClass objek1 = new GarbaceCollectorClass(); //Block Satu  GarbaceCollectorClass objek2 = new GarbaceCollectorClass(); //Block Dua  GarbaceCollectorClass objek3 = objek1; //Objek3 Mengembalikan Nilai Objek1  objek1 = null;  objek2 = null;  System.*out*.println("Jumlah Memori Yeng Tersedia Sebelum di GC: "+RT.freeMemory());  System.*gc*();  System.*out*.println("===========================================");  System.*out*.println("Jumlah Memori Yeng Tersedia Setelah di GC: "+RT.freeMemory());  }  } |

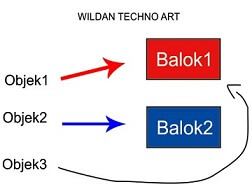
**Output:**

A screenshot of a computer code

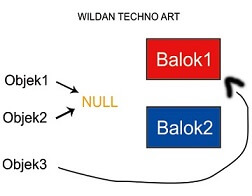
Description automatically generated

Gambar Output Gerbage Collector

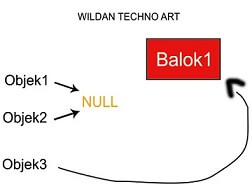
**Penjelasan**:  
Pada program tersebut, disana kita membuat 3 buah objek atau variable dari Class GarbaceCollectorClass, objek1 dan objek2 diinisialisasikan kedalam objek1 dan objek2, dan objek3 mempunyai nilai kembali dari objek GarbaceCollectorClass1.



Pada baris selanjutnya, objek1 dan objek2 kita ubah nilainnya menjadi null yang berarti kosong, sedangkan objek3 masih mengembalikan nilainya objek1.



Karena pada GarbaceCollectorClass2 tidak ada yang direferensikan nilainya, maka secara otomatis, Balok2 tersebut akan di bawa oleh System Garbage Collection pada JVM.



System.gc(), berfungsi untuk menjalankan Garbage Collection pada java. lalu Method finalize(), digunakan untuk memastikan bahwa objek telah bersih dan Garbage Collection telah bekerja, Di dalam metode finalize(), kita dapat menentukan tindakan yang harus dilakukan sebelum suatu objek hancur.  
  
Dan terakhir, kita menambahkan sebuah variable Runtime, yang digunakan untuk mengecek Total Memori serta Jumlah Memori yang tersedia sebelum dan sesudah di Garbage Collection.

# Referensi

<https://www.dicoding.com/blog/apa-itu-database/>

<https://dosenit.com/sql/perbedaan-sql-dan-nosql>

<https://aws.amazon.com/id/nosql/>

<https://www.sekawanmedia.co.id/pengertian-sql/>

<https://www.contohapps.com/2020/08/mengenal-database-nosql.html>

<https://idcloudhost.com/panduan-sql-fungsi-cara-kerja-serta-perintah-dasarnya/>

<https://www.tutorialspoint.com/sql/sql-data-types.htm>

<https://arfianhidayat.com/tipe-constraint-pada-sql-dan-fungsi-constraint>

<https://arfianhidayat.com/view-pada-sql>

<https://bundet.com/d/917-apa-itu-tuning-basis-data>

<https://www.w3schools.com/sql/sql_alter.asp>

<https://www.duniailkom.com/tutorial-belajar-java-part-3-pengertian-jre-dan-jdk/>

<https://sis.binus.ac.id/2018/02/02/pengenalan-java-console/>

<https://www.duniailkom.com/tutorial-belajar-java-part-3-pengertian-jre-dan-jdk/>

<https://www.dicoding.com/blog/berkenalan-dengan-intellij-idea/>

<https://id.wikipedia.org/wiki/PostgreSQL>

<https://medium.com/@novancimol12/postman-4f181d625fe1>

<https://www.petanikode.com/git-untuk-pemula/>

<https://dedykuncoro.com/2013/03/macam-macam-tipe-data-pada-java.html>

https://www.duniailkom.com/tutorial-belajar-java-pengertian-variabel-dan-cara-penulisan-variabel-bahasa-java/

<https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html>

https://www.petanikode.com/java-prosedur-dan-fungsi/

<https://www.petanikode.com/java-input-output/>

<https://www.niagahoster.co.id/blog/java-adalah/?amp>

<https://www.kodingindonesia.com/cara-install-jdk/>

<https://www.dicoding.com/blog/berkenalan-dengan-intellij-idea/>

<https://www.elfanmauludi.id/2019/03/for-each-loop-pada-java.html>

<https://www.w3schools.com/java/java_while_loop.asp>

https://www.petanikode.com/java-operator/  
UBL/Pemprograman Java-M.Anif

<https://www.duniailkom.com/tutorial-belajar-java-percabangan-kondisi-if-else-if-bahasa-java/>

<https://www.anbidev.com/prinsip-solid/>

<https://www.anbidev.com/prinsip-dasar-programmer/>

<https://glints.com/id/lowongan/kiss-principle-adalah/#.YT2HlJ0zbIU>

<https://java-design-patterns.com/principles/>

binar-academy-java-be

https://www.anbidev.com/python-oop-intro/

<https://idcloudhost.com/panduan/mengenal-pengertian-dan-konsep-oop/>

<https://www.anbidev.com/python-oop-intro/>

<https://www.jagoanhosting.com/blog/mengenal-oop-encapsulation-inheritance-polyorphism-dan-abstrak-class/>

<http://imamfarisi.com/encapsulation-di-java/>

<https://andriansyahdoni.wordpress.com/2016/12/23/encapsulation-in-java-programming-using-netbeans/>

<https://sis.binus.ac.id/2017/09/07/inheritance/>

<http://frieyadie.web.id/constructor-pada-java/>

<https://lecturer.ppns.ac.id/khoirul/method-class-object/>

<https://bahasajava.com/method-overriding-java/>

<https://bahasajava.com/keyword-super-program-java/>

<https://www.wildantechnoart.net/2017/11/fungsi-garbage-collection-pada-java.html>

https://www.anbidev.com/python-oop-intro/

<https://idcloudhost.com/panduan/mengenal-pengertian-dan-konsep-oop/>

<https://www.anbidev.com/python-oop-intro/>

<https://www.jagoanhosting.com/blog/mengenal-oop-encapsulation-inheritance-polyorphism-dan-abstrak-class/>

<http://imamfarisi.com/encapsulation-di-java/>

<https://andriansyahdoni.wordpress.com/2016/12/23/encapsulation-in-java-programming-using-netbeans/>

<https://sis.binus.ac.id/2017/09/07/inheritance/>

<http://frieyadie.web.id/constructor-pada-java/>

<https://lecturer.ppns.ac.id/khoirul/method-class-object/>

<https://bahasajava.com/method-overriding-java/>

<https://bahasajava.com/keyword-super-program-java/>

https://www.wildantechnoart.net/2017/11/fungsi-garbage-collection-pada-java.html