**Java Logging**

## **Pendahuluan**

**Sebelumnya harus sudah paham materi terkait** :

* Java Dasar
* Java Object Oriented Programming
* Apache Maven
* Java Unit Test

**Agenda**

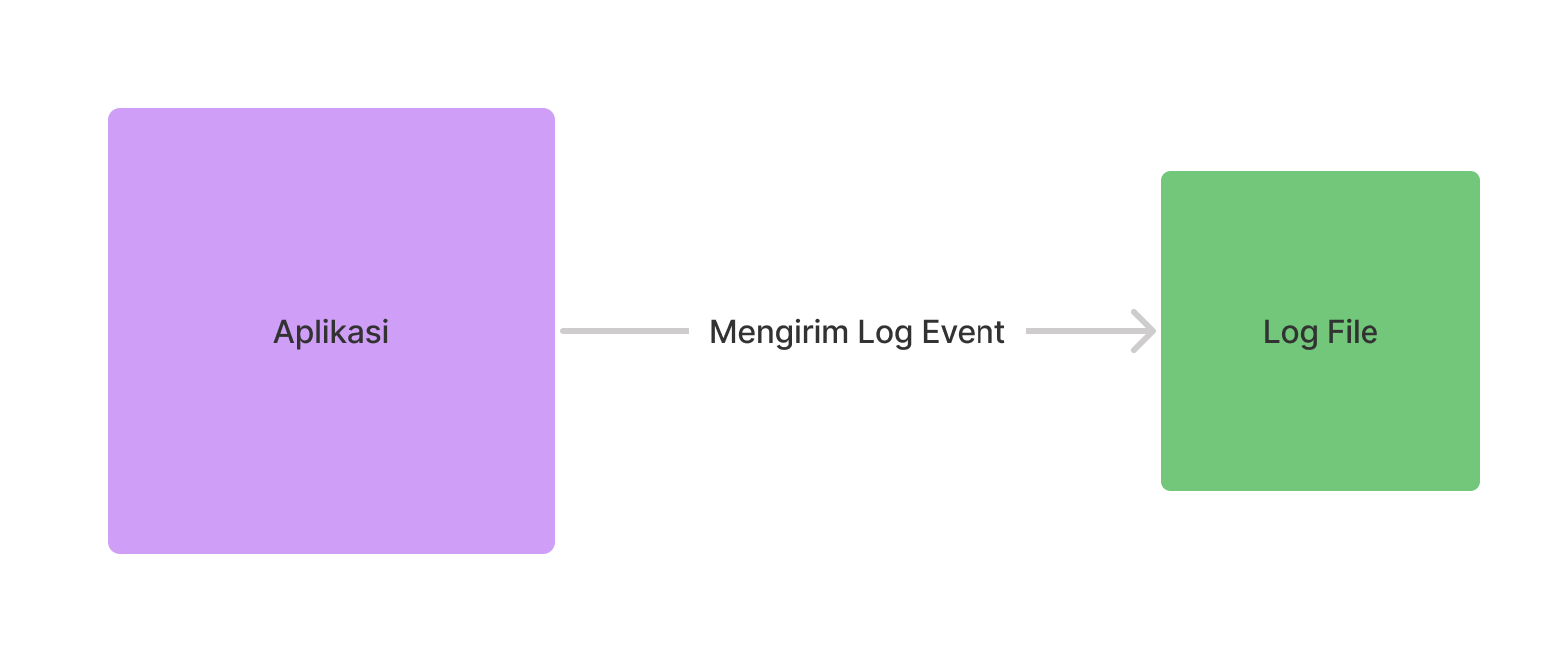
* Pengenalan Logging
* Logger
* Level
* Configuration
* Appender
* Layout
* Dan lain-lain

## **Pengenalan Logging**

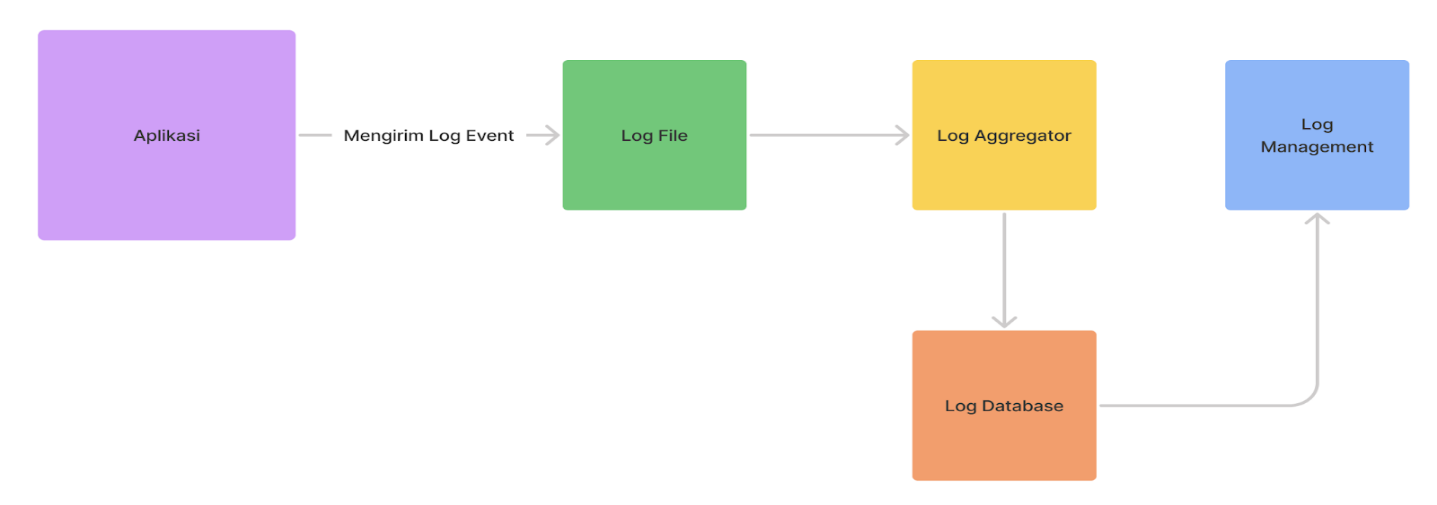
**Bean Validation**

* Log file adalah file yang berisikan informasi kejadian dari sebuah sistem
* Biasanya dalam log file, terdapat informasi waktu kejadian dan pesan kejadian
* Logging adalah aksi menambah informasi log ke log file
* Logging sudah menjadi standard industri untuk menampilkan informasi yang terjadi di aplikasi yang kita buat
* Logging bukan hanya untuk menampilkan informasi, kadang digunakan untuk proses debugging ketika terjadi masalah di aplikasi kita

**Diagram Logging**



**Ekosistem Logging**



## **Logging LibraryLogger**

**Java Logging**

* Java sendiri sebenarnya memilih package yang dikhususkan untuk logging
* Namun saat ini, kebanyakan programmer tidak menggunakannya
* Hal ini dikarenakan penggunaannya yang kurang flexible
* <https://docs.oracle.com/javase/8/docs/api/java/util/logging/package-summary.html>

**Logging Library**

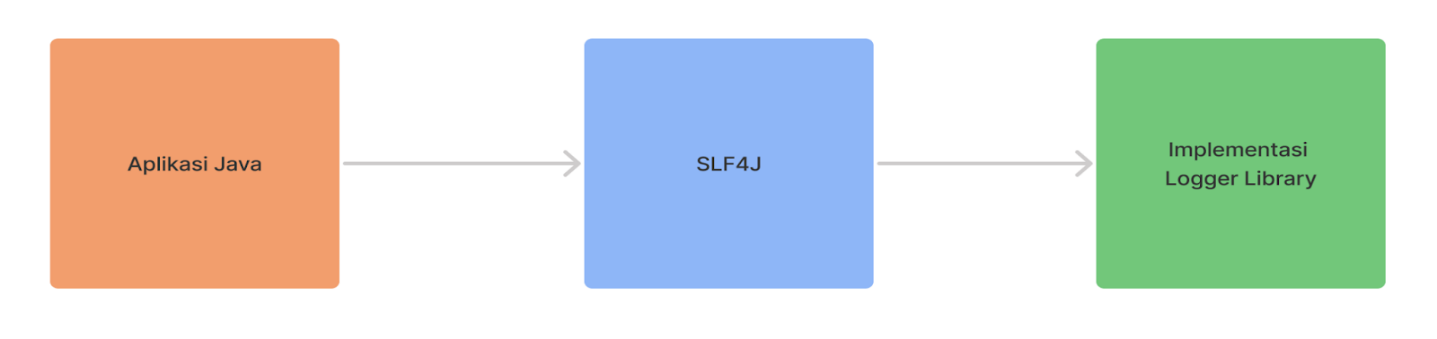
Diluar Java Logging, banyak sekali library yang bisa kita gunakan untuk logging, seperti :

* Apache Common Log
* Apache Log4J
* Logback
* dan lain-lain

**SLF4J**

* Pada kelas ini kita akan menggunakan SLF4J
* SLF4J merupakan framework logging yang digunakan seperti API, dimana kita bisa berganti-ganti implementasi logging framework nya
* SLF4J banyak sekali digunakan oleh programmer karena sangat flexible untuk berganti-ganti library logging, tanpa harus melakukan perubahan di kode program nya
* Kita hanya perlu memilih library yang akan digunakan, dan secara otomatis SLF4J akan menggunakan implementasi library tersebut
* <http://www.slf4j.org/>

**Diagram SLF4J**



**Logback**

* Untuk implementasi library logging nya, kita akan menggunakan Logback
* Logback merupakan salah satu library logging yang populer di Java
* Terutama Logback digunakan secara default di framework Spring Boot, salah satu framework paling populer di Java
* <http://logback.qos.ch/>

**Membuat Project**

* <https://start.spring.io/>

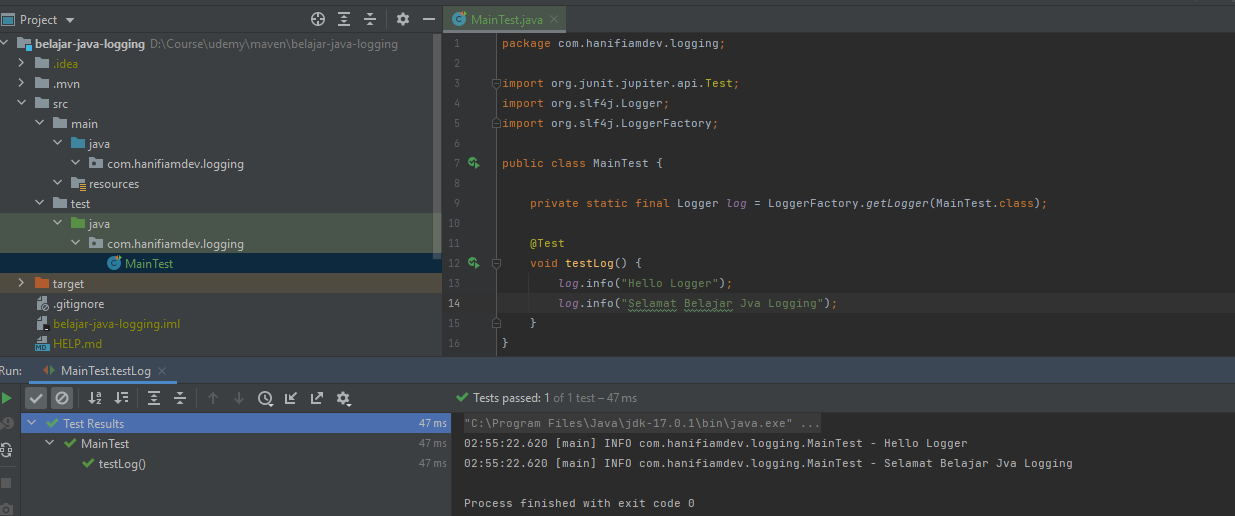
## **LoggerLevel**

* Logger adalah class utama untuk melakukan logging
* Saat kita membuat Logger, biasanya kita akan menyebutkan nama Logger nya
* Biasanya nama logger menggunakan nama class lokasi Logger tersebut
* Hal ini agar mudah ketika melihat hasil log, dari mana hasil log tersebut
* http://www.slf4j.org/apidocs/org/slf4j/Logger.html

**Membuat Logger**

* Untuk membuat Logger, kita tidak perlu membuat objectnya manual menggunakan new
* Kita bisa memanfaatkan factory class LoggerFactory
* <http://www.slf4j.org/apidocs/org/slf4j/LoggerFactory.html>

Kode : Hello Log



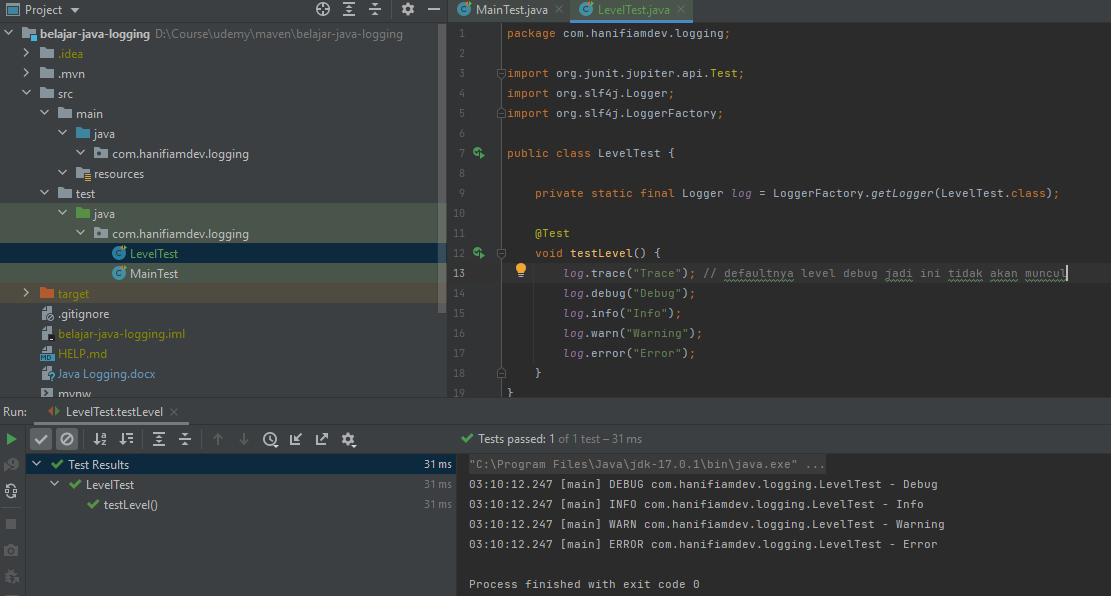
## **LevelLog Format**

* Log memiliki banyak level
* Level disini merupakan level jenis informasi log
* Level itu bertingkat, semakin tinggi artinya informasinya semakin penting
* Level juga biasanya disesuaikan dengan jenis errornya
* Setiap level memiliki method di logger, sehingga kita bisa gunakan level langsung di method nya

**Level (dari terendah ke tertinggi)**

|  |  |
| --- | --- |
| Level | Keterangan |
| trace | Biasanya untuk menambahkan informasi tracing |
| debug | Biasanya untuk menambahkan informasi debug |
| info | Biasanya untuk menambahkan informasi |
| warn | Biasanya untuk menambahkan peringatan |
| error | Biasanya untuk menambahkan error |

Kode : Level Log



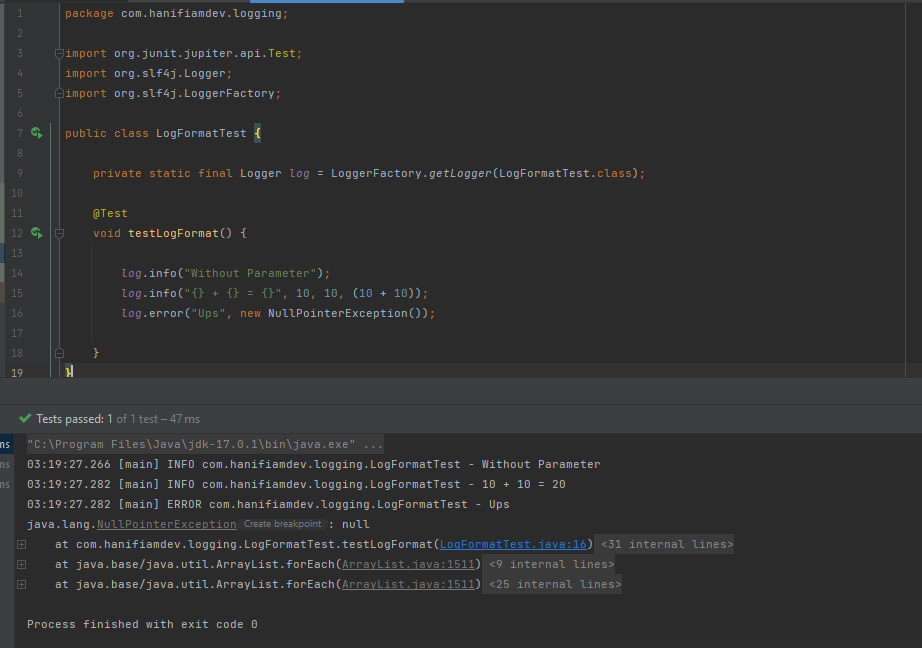
## **Log FormatConfiguration**

* Kadang kita ingin menggunakan parameter saat melakukan logging
* Biasanya kita akan membuat string concat untuk membuat pesan logging nya
* Namun SLF4J sudah menyediakan log format
* Kita bisa menggunakan beberapa method overloading nya

**Log Method**

|  |  |
| --- | --- |
| Method | Keterangan |
| level(String) | Melakukan logging berisi string |
| level(String, Object...) | Melakukan logging dengan parameter, gunakan {} sebagai parameter nya |
| level(String, Throwable) | Melakukan logging dengan menambah stack trace error |

Kode : log Format

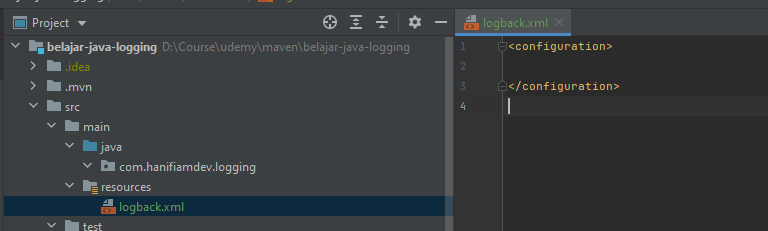


## **Configuration**

* Secara default, saat kita menggunakan logback, kita tidak butuh configuration
* Namun kadang-kadang kita ingin menggunakan configuration
* Misal, ketika di laptop kita ingin menjalankan logging sampai ke level trace, namun ketika production misal kita hanya butuh di level warning agar tidak terlalu banyak log
* Hal tersebut, perlu kita lakukan dengan membuat file konfigurasi

**logback.xml**

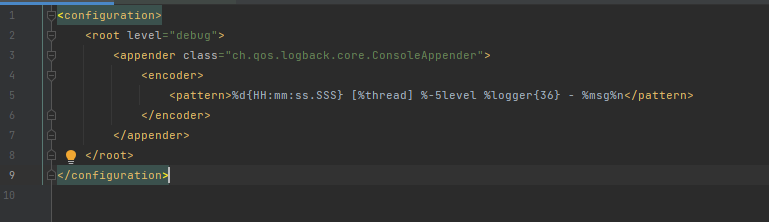
* Logback akan membaca konfigurasi dari file logback.xml yang terdapat di default package
* Artinya kita butuh membuat file logback.xml pada default package



**Level Configuration**

* Secara default, saat kita membuat file konfigurasi, logback akan membaca level yang harus dikeluarkan dari file logback
* Jika tidak ada, maka otomatis tidak akan keluar
* Oleh karena itu, hal pertama yang perlu kita lakukan adalah, menambahkan logger level, untuk memberitahu level mana yang ingin kita keluarkan di log file
* <https://gist.github.com/khannedy/e2ec8fcb20344523f3d9e2a8fcc108d7>

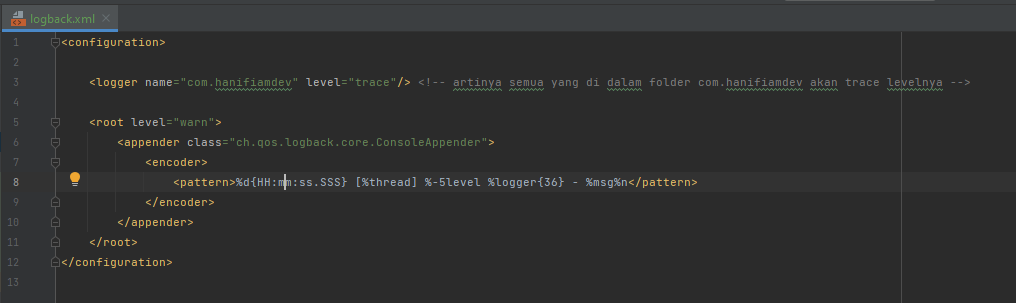
Kode : Root Logger



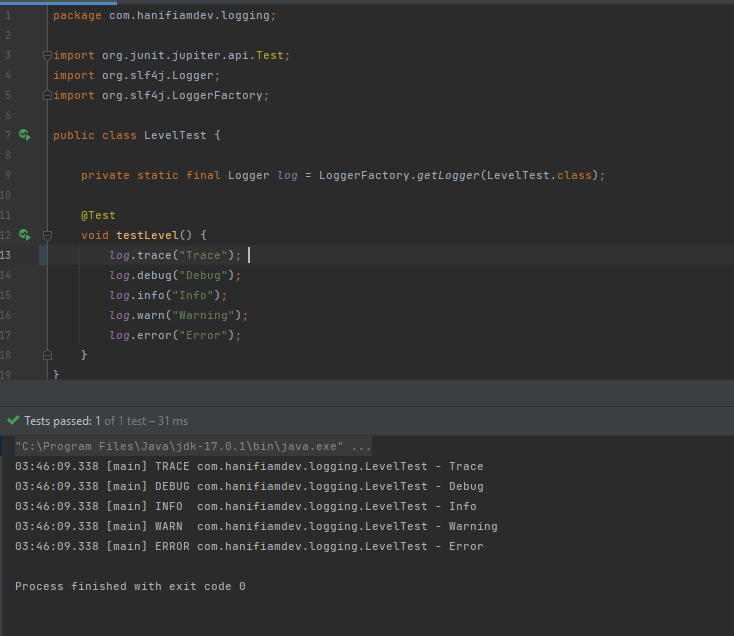
**Logger Package**

* Kadang kita ingin membuat logging level berbeda-beda untuk package
* Misal untuk package framework, kita ingin gunakan warn, tapi untuk package aplikasi kita, kita ingin gunakan info
* Secara default, sebelumnya kita sudah buat root, root adalah default fallback semua package level ketika tidak dibuatkan konfigurasi secara spesifik
* Namun jika kita ingin, kita juga bisa membuat logger level lebih dari satu
* Tinggal gunakan prefix package nya saja, artinya semua package di dalamnya akan ikut logger level tersebut
* <https://gist.github.com/khannedy/63ac69c2be341c02d4b0a28e19490639>

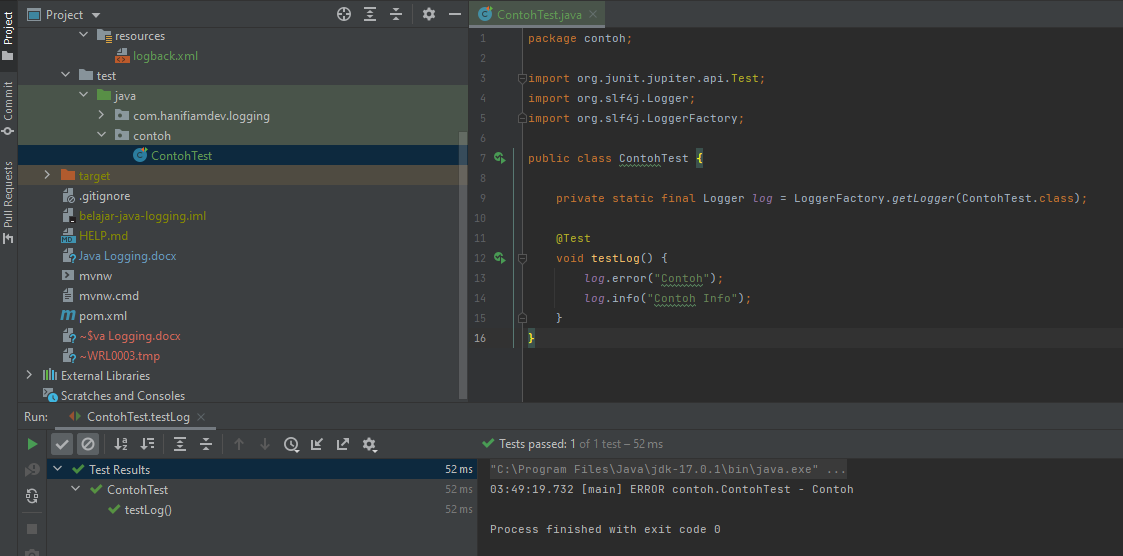
Kode : Package Logger



Test



Jika misalnya kita buat kelas yang tidak didalam package com.hanif



Note : log.info tidak muncul karena root di configuration dibuat warn dan tidak membaca level log package com.hanif karena beda package