**JAVA UNIT TEST**

## **Pendahuluan**

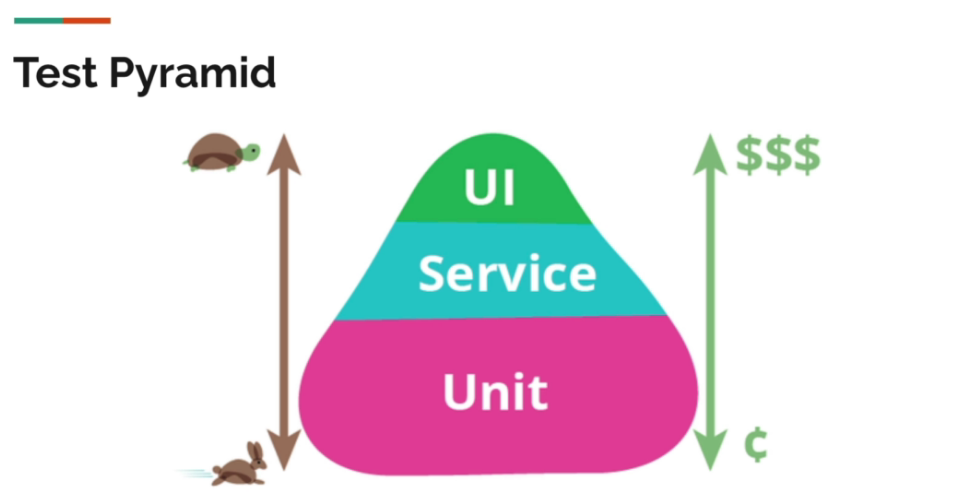
## ## Sebelum Belajar Materi ini ##

## sebaiknya mempelajari Java Dasar, Java Object Oriented Programming, Java Generic, Java Collection, Java Lambda, Apache Maven

## ## Agenda ##

Pengenalan Software Testing, Pengenalan JUnit, Membuat Test, Menggunakan Assertions, Menggunakan Assumptions, Dependency Injection di Test, Mocking Dan lain-lain

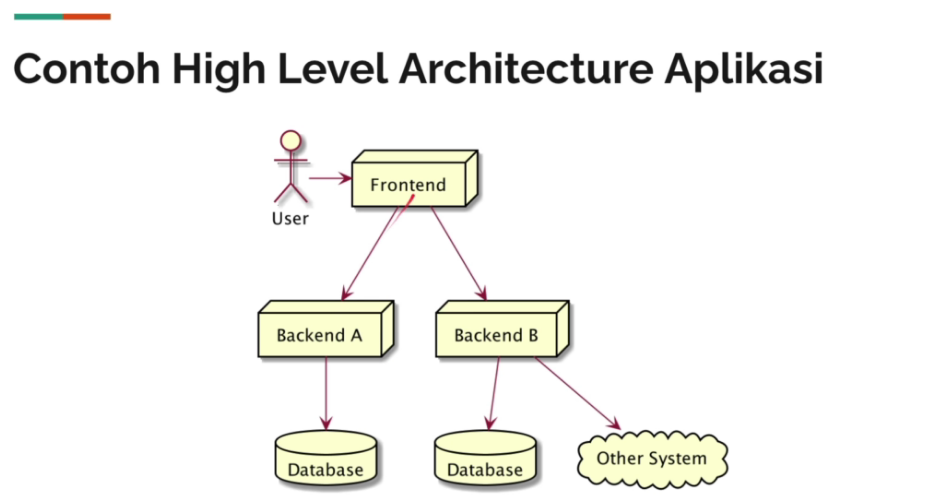
## **Pengenalan Software Testing**

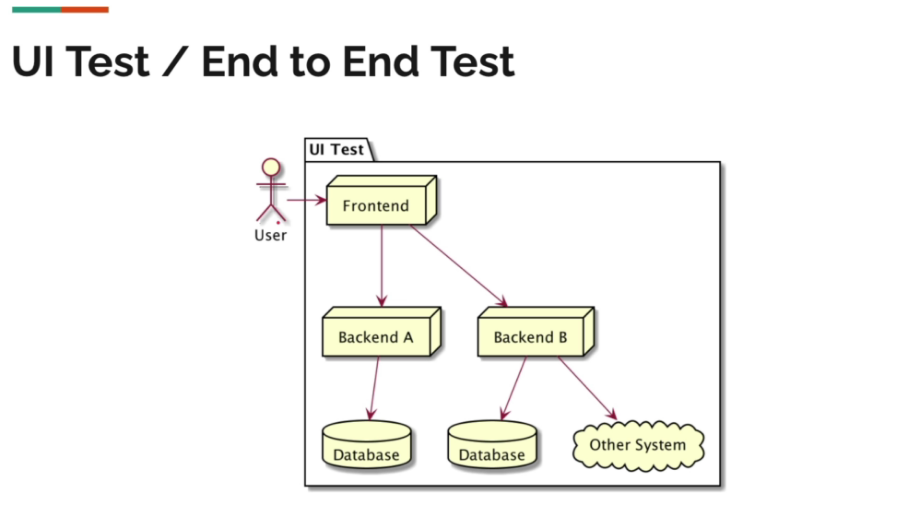


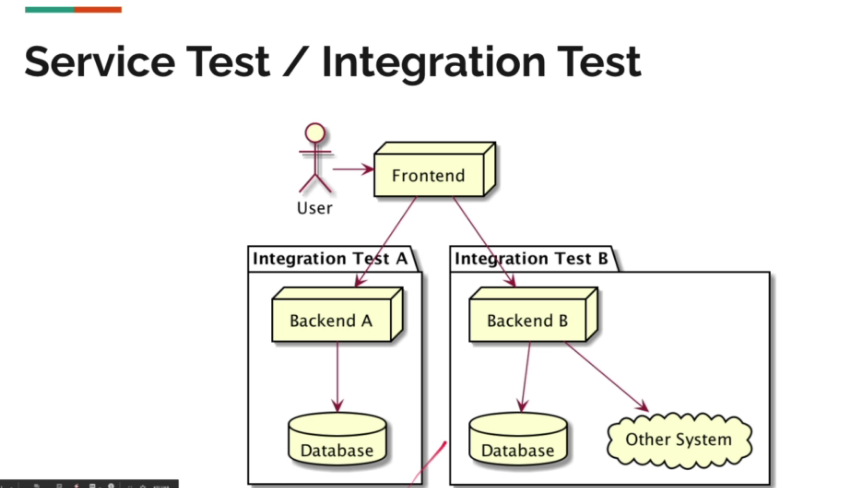
Unit test : testing yang harusnya dibuta sebanyak-banyaknya , yang cepat dan murah

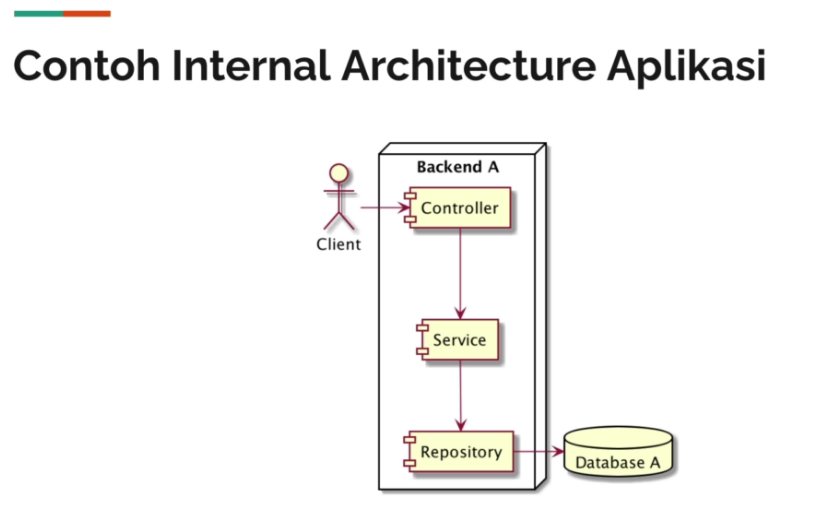
Service : testing pertengahan tidak murah banget, tidak lambat banget juga

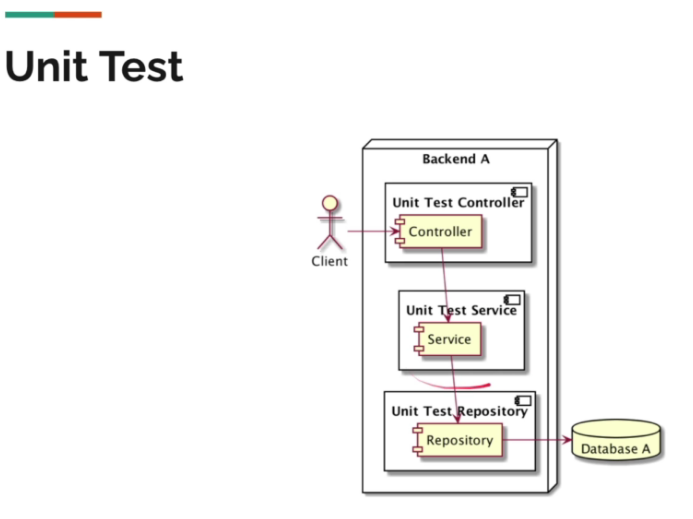
UI : paling lambat pengerjaan, paling mahal











## ## Unit Test ##

## Unit test akan focus menguji bagian kode program terkecil, biasanya menguji sebuah method

* Unit test biasanya dibuat dan cepat, oleh karena itu biasanya kadang kode unit test lebih bnyak dari kode program aslinya, karena semua scenario pengujian akan dicoba di unit test
* Unit test bisa digunakan sebagai cara untuk meningkatkan kualitas kode program kita

## **Pengenalan Junit**

## Junit ##

* Junit adalah test framework yang paling popular di Java
* Saat ini versi terbaru Junit adalah versi 5
* Junit 5 membutuhkan Java minimal versi 8
* https://junit.org/

## Creatre Maven ##

* mvn archetype:generate
* maven-archetype-quickstart
* Pilih *org.maven.archetypes-maven-archetypes-quickstart*
* Pilih version
* Groupid, biasanya nama atau url organisasi atau perusahaan
* artifacId, nama project bisanya hurus kecil semua dan dipisah dengan tanda “-”
* version yaitu terkait versi project
* package yaitu nama package project , biasanya dipihasahkan dengan titik penamaannya

## **Membuat Test**

* Untuk membuat test di Junit itu sederhana, kita cukup membuat class, lalu menambahkan method-method test nya
* Method akan sianggap sebuah test jika ditambahkan annotation @Test
* Kode test disimpan dibagian test folder di maven, bukan di main folder
* Biasanya saat membuat class untuk test, rata- rata orang biasa membuat nama class nya sama dengan nama class yang akan di test, tapi diakhiri dengan kata Test, missal jika nama class nya adalah Calculator, maka nama class test nya adalah Calculator Test

## Menggagalkan Test ##

* Kadang dalam membuat unit test, kita tidak hanya ingin mengetest kasus sukses atau gagal
* Ada kalanya kita ingin mengetes sebuah exception misalnya
* Assertion juga bisa digunakan untuk mengecek apakah sebuah exception terjadi

## **Menggunakan Assertions**

## Assertions ##

Saat membuat test, kita harus memastikan bahwa test tersebut sesuai dengan ekspetasi yang kita inginkan

Jika manual, kitab isa melakukan pengecekan if else, namun itu tidak direkomendasikan

JUnit memiliki fitur untuk melakukan assertions, yaitu memastikan bahwa unit test sesuai dengan kondisi yang kita inginkan

Assertions di JUnit di representasikan dalam class Assertions, dan di dalamnya terdapat banyak sekali function static

https://junit.org/junit5/docs/current/api/org.junit.jupiter.api/org/junit/jupiter/api/Assertions.html

## Mengagalkan Test ##

Kadang dalam membuat unit test, kita tidak hanya ingin mengetest kasus sukses atau gagal

Ada kalanya kita ingin mengetes sebuah exception misalnya

Assertions juga bisa digunakan untuk mengecek apakah sebuah exception terjadi

## **Mengubah Nama Test**

Kadang agak sulit membuat nama function yang mempresentasikan kasus test nya

Jika kita ingin menambahkan deskripsi untuk tiap test, kitab isa menggunakan annotation @DisplayName

Dengan menggunakan annotation@DisplayName, kitab isa menambahkan deskripsi unit testnya

## Menggunakan Display Name Generator ##

JUnit mendukung pembuatan DisplayName secara otomatis menggunakan generator

Yang perlu kita lakukan adalah membuat class turunan dari interface DisplayNameGenerator, lalu menambahkan annotation @DisplayNameGneneration di test class nya

## **Menonaktifkan Test**

* Kadang adakalanya kita ingin menonaktifkan unit test, missal karena terjadi eror di unit test tersebut, dan belum bisa kita perbaiki
* Sebenarnya cara paling mudah untuk menonaktifkan unit test adalah dengan menghapus
* annotation @Test, namun jika kita lakukan itu, kita tidak bisa mendeteksi kalua ada unit testnya yang di disabled
* Untuk menonaktifkan unit test secara benar, kita bisa menggunakan annotation @Disabled

## **Sebelum & Setelah Test**

## Sebelum & Setelah Unit Test ##

* Kadang kita ingin menjalankan kode yang sama sebelum dan setelah ekseskusi unit test
* Hal ini sebenarnya bisa dilakukan secara manual di function @Test nya, namun hal ini akan membuat kode duplikat banyak sekali
* JUnit memiliki annotation @BeforeEach dan @AfterEach
* @BeforeEach digunakan untuk menandai function yang akan dieksekusi sebelum unit test dijalankan
* @AfterEach digunakan untuk menandai function yang akan dieksekusi sebelum unit test dijalankan
* Ingat, bahwa ini akan selalu dieksekusi setiap kali untuk function @Test, bukan sekali untuk class test saja

## Sebelum & Setelah Semua Unit Test ##

* @BeforeEach & @AfterEach akan diekseskusi setiap kali function @Test jalan
* Namun kadang kita ingin melakukan sesuatu sebelum semua unit test berjalan, atau setelah semua unit test berjalan
* Ini bisa dilakukan menggunakan annotation @BeforeAll dan @AfterAll
* Namun hanya static function yang bisa menggunakan @BeforeAll dan @AfterAll

## **Membatalkan Test**

* Kadang kita ingin membatalkan unit test Ketika kondisi tertentu terjadi
* Untuk membatalkan, kitab isa menggunkan exception TestAbortedException
* Jika JUnit mendapatkan exception TestAbortedException, secara otomatis test tersebut akan dibatalkan