



UNIVERSITAS
HASANUDDIN

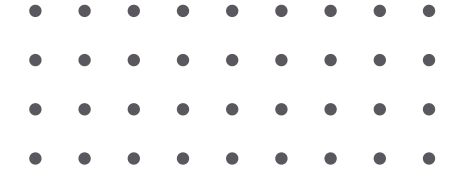
PENGANTAR *UNIT TESTING*

INGIN CODING TANPA KHAWATIR *BUG*?
UNIT TESTING ADALAH KUNCINYA!



KELOMPOK 5





PERKENALAN DULU GA SIH?

KELOMPOK 5

Zaenab Putri Az Zakiyyah

H071231001

Nancy Jiwono

H071231004

Rudy Peter

H071231015

Cholyn Sharon Enos

H071231040

Andi Aisar Saputra Dwi Anna

H071231048

M. Ervin

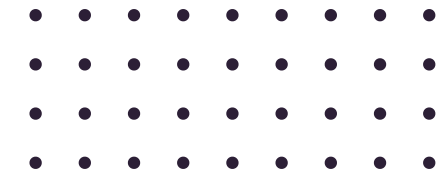
H071231050

Fara Rahmasari Fahirun

H071231055

Harmelia Yuli Rahmatika A.

H071231079



PEMBAHASAN KAMI

01 APA ITU *UNIT TESTING*?

02 KENAPA *UNIT TESTING* ITU PENTING?

03 PENGENALAN *FRAMEWORK* POPULER

04 POLA DASAR *ARRANGE, ACT, ASSERT (AAA)*

05 CARA MEMVERIFIKASI HASIL TES

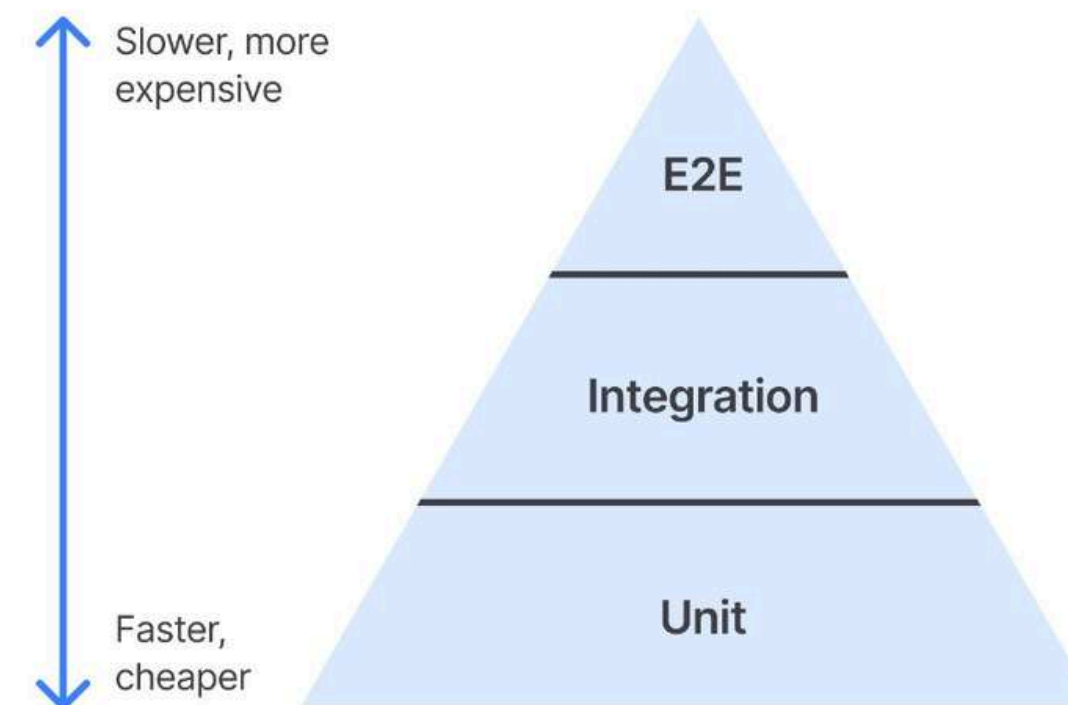
06 CONTOH *LIVE CODING UNIT TESTING* SEDERHANA



APA ITU *UNIT TESTING*?

Unit testing adalah salah satu jenis pengujian perangkat lunak (*software*) yang berfokus pada pengujian unit-unit terkecil dalam sebuah sistem perangkat lunak. Biasanya, unit testing mencakup pengujian *function*, *method*, dan *class*.

Umumnya, *unit testing* adalah pengujian paling awal yang dilakukan oleh developer sebelum melakukan pengujian lain, seperti *integration test*, *functional test*, dan *end-to-end test*.

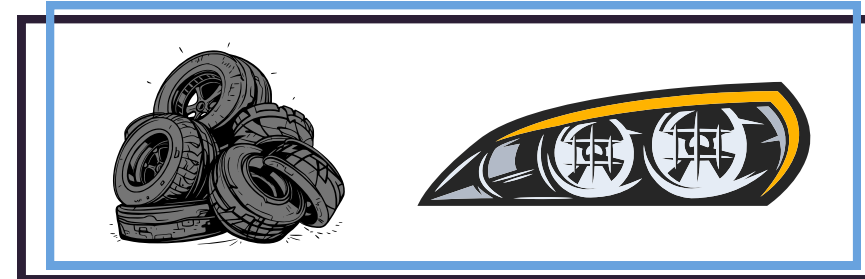


Menguji komponen tanpa bergantung pada sistem lain dan hanya menguji *function by function*.

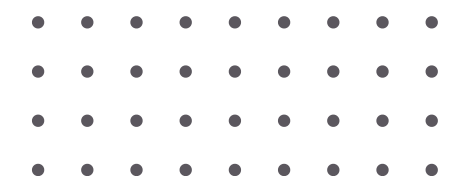


ANALOGI *UNIT TESTING*

Saat mengecek setiap komponen mobil secara terpisah sebelum di rakit menjadi satu mobil utuh.



Jika setiap komponen lulus tes, mobil rakitan pasti berkualitas. Jika ada masalah, kita tahu bukan dari komponen dasar.





KENAPA *UNIT TESTING* ITU PENTING?




Membantu memastikan keandalan, kualitas, dan kemudahan pemeliharaan kode yang kita tulis.

- 1 Mempermudah Perubahan dan *Refactoring*
- 2 Memberikan Dokumentasi Kode yang Hidup
- 3 Mendeteksi *Bug* Lebih Awal
- 4 Meningkatkan Kualitas Kode
- 5 Menghemat Waktu dan Biaya
- 6 Meningkatkan Kepercayaan Diri



Pengenalan *FRAMEWORK* Populer

Tiga *framework* paling populer di ekosistemnya masing-masing:

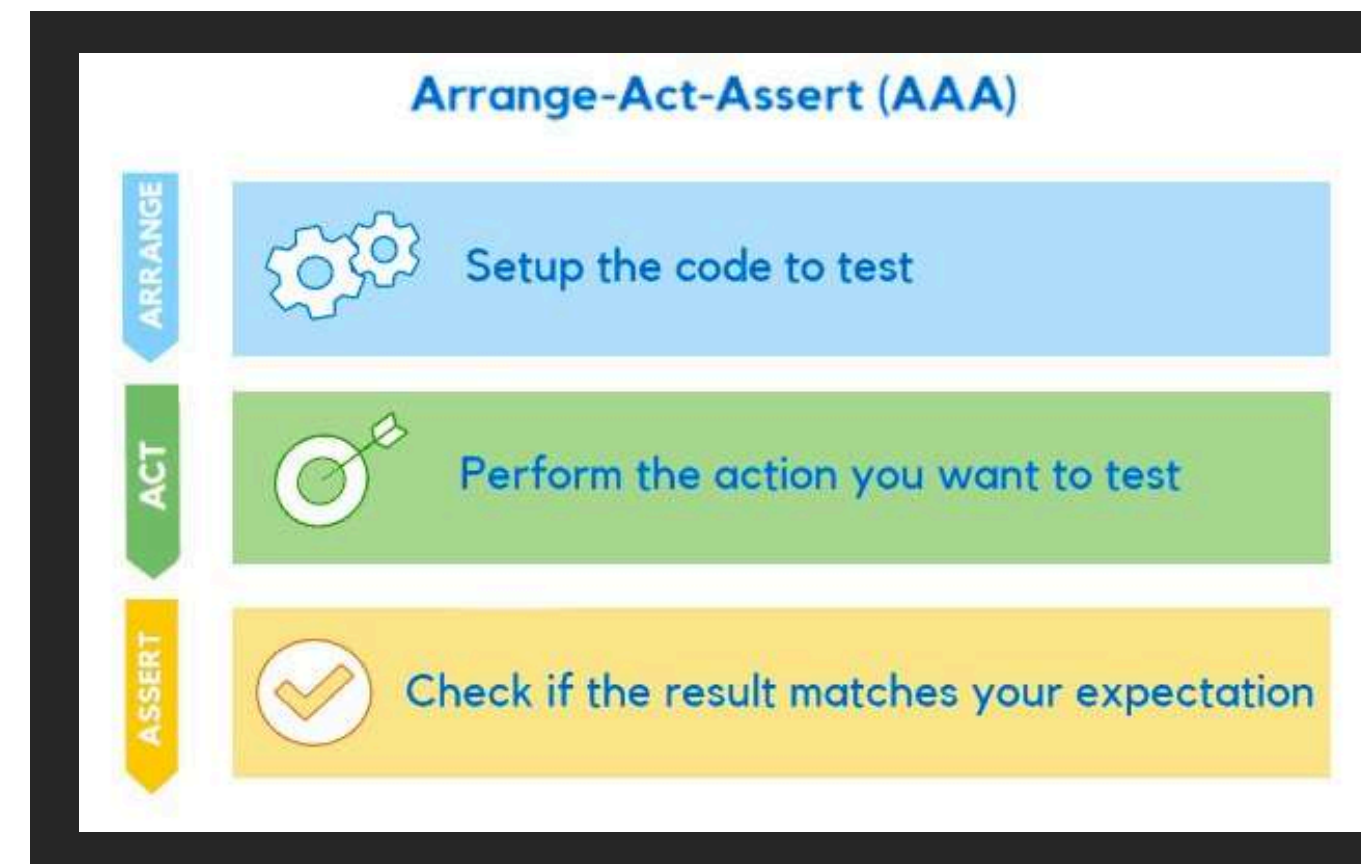
	JUNIT 5 (JAVA) <i>Framework de facto</i> dan pelopor dalam dunia unit testing untuk Java.		JEST (JAVASCRIPT) <i>Framework</i> JavaScript buatan Meta yang efisien dan menyenangkan.		PYTEST (PYTHON) Sederhana, mudah dibaca, dan kuat dengan fitur canggih tersembunyi.
Kapan digunakan? Jika kita bekerja dengan Java atau bahasa berbasis JVM lainnya (seperti Kotlin atau Scala).		Kapan digunakan? jika menggunakan React, Node.js, TypeScript, atau <i>framework frontend</i> modern lainnya.		Kapan digunakan? Proyek Python apapun, mulai skrip sederhana, aplikasi web, hingga proyek <i>data science</i> dan API.	
Keunggulan: <ul style="list-style-type: none">• Integrasi Penuh• Ekosistem yang Matang• Struktur Berbasis Anotasi		Keunggulan: <ul style="list-style-type: none">• Konfigurasi Minimal (<i>Zero-Config</i>)• <i>Batteries-Included</i>• Fitur <i>Snapshot Testing</i>		Keunggulan: <ul style="list-style-type: none">• Sintaks Sederhana & <i>Boilerplate</i> Rendah• <i>Fixtures</i> yang Sangat Kuat• Pelaporan <i>Error</i> yang Detail	



POLA DASAR *ARRANGE, ACT, ASSERT* (AAA)

Pendekatan populer dalam penulisan *unit test* yang membagi setiap tes menjadi tiga bagian utama:

- 1 ARRANGE**
Menyiapkan kondisi awal tes.
- 2 ACT**
Menjalankan fungsi atau metode yang akan diuji.
- 3 ASSERT**
Memverifikasi bahwa hasil dari tindakan yang dilakukan sesuai dengan ekspektasi.





LIVE CODING

UNIT TESTING



HASIL *LIVE CODING* KAMI MENGGUNAKAN PYTEST (PYTHON)

Kode File Shopping Cart

```
from typing import List

class ShoppingCart:
    def __init__(self, max_size: int) -> None:
        self.items: List[str] = []
        self.max_size = max_size

    def add(self, item: str):
        if len(self.items) == self.max_size:
            raise OverflowError("cannot add more items")
        self.items.append(item)

    def get_items(self) -> List[str]:
        return self.items

    def update(self, old_item: str, new_item: str):
        if old_item not in self.items:
            raise ValueError("item not found")
        index = self.items.index(old_item)
        self.items[index] = new_item

    def delete(self, item: str):
        if item not in self.items:
            raise ValueError("item not found")
        self.items.remove(item)
```

Kode File Test Shopping Cart

```
import pytest
from shopping_cart import ShoppingCart

@pytest.fixture
def cart():
    return ShoppingCart(5)

def test_add(cart):
    cart.add("apple")
    assert "apple" in cart.get_items()
    assert len(cart.get_items()) == 1

def test_add_overflow(cart):
    with pytest.raises(OverflowError):
        for _ in range(6):
            cart.add("apple")

def test_get_items(cart):
    cart.add("apple")
    cart.add("banana")
    assert cart.get_items() == ["apple", "banana"]

def test_update_item(cart):
    cart.add("apple")
    cart.update("apple", "orange")
    assert "orange" in cart.get_items()
    assert "apple" not in cart.get_items()

def test_update_item_not_found(cart):
    with pytest.raises(ValueError):
        cart.update("grape", "melon")

def test_delete_item(cart):
    cart.add("apple")
    cart.add("banana")
    cart.delete("apple")
    assert "apple" not in cart.get_items()
    assert cart.get_items() == ["banana"]

def test_delete_item_not_found(cart):
    with pytest.raises(ValueError):
        cart.delete("grape")
```



HASIL *LIVE CODING* KAMI MENGGUNAKAN JUNIT 5 (JAVA)

Kode File BankAccountTest

```
1 package com.example;
2 import com.example.BankAccount;
3
4 import static org.junit.jupiter.api.Assertions.*;
5 import org.junit.jupiter.api.Test;
6
7 class BankAccountTest {
8
9
10     @Test
11     void testDeposit() {
12         BankAccount account = new BankAccount();
13         account.deposit(100);
14
15         // Assertions untuk memverifikasi hasil test
16         assertEquals(100, account.getBalance());
17     }
18
19     @Test
20     void testWithdrawSuccess() {
21         BankAccount account = new BankAccount();
22         account.deposit(200);
23         account.withdraw(50);
24
25         // Assertions
26         assertEquals(150, account.getBalance());
27     }
28
29     @Test
30     void testWithdrawInsufficientFunds() {
31         BankAccount account = new BankAccount();
32         account.deposit(50);
33
34         // Assertions untuk error/exception
35         Exception exception = assertThrows(IllegalArgumentException.class, () -> {
36             account.withdraw(100);
37         });
38
39         // Assertions cek pesan error sesuai
40         assertEquals("Saldo tidak cukup!", exception.getMessage());
41     }
42 }
43
```

Kode File BankAccount

```
1 package com.example;
2
3 public class BankAccount {
4     private int balance = 0;
5
6     public void deposit(int amount) {
7         balance += amount;
8     }
9
10    public void withdraw(int amount) {
11        if (amount > balance) {
12            throw new IllegalArgumentException("Saldo tidak cukup!");
13        }
14        balance -= amount;
15    }
16
17    public int getBalance() {
18        return balance;
19    }
20 }
21
22
```

AYOK BERDISKUSI





**TERIMA
KASIH**



SAMA

SAMA