Cập nhật tháng 8 năm 2024

# [Bài đọc] Khái niệm về unit test

## 1. Unit Test Là Gì?

- Định nghĩa:
  - **Unit Test** là phương pháp kiểm thử phần mềm, trong đó **một đơn vị nhỏ nhất của chương trình (thường là một hàm, một phương thức hoặc một class)** được kiểm tra một cách độc lập để đảm bảo rằng nó hoạt động đúng như mong đợi
- Mục tiêu:
  - Đảm bảo rằng từng thành phần nhỏ của phần mềm hoạt động chính xác

## 2. Đặc Điểm Của Unit Test

- Kiểm thử ở **cấp độ nhỏ nhất**: phương thức, class
- Độc lập với các thành phần khác: Không phụ thuộc vào database, file, hay hệ thống bên ngoài
- Có thể chạy tự động, nhanh chóng
- Kết quả rõ ràng: Pass (✔) hoặc Fail (❌)

## 3. Vì Sao Cần Unit Test?

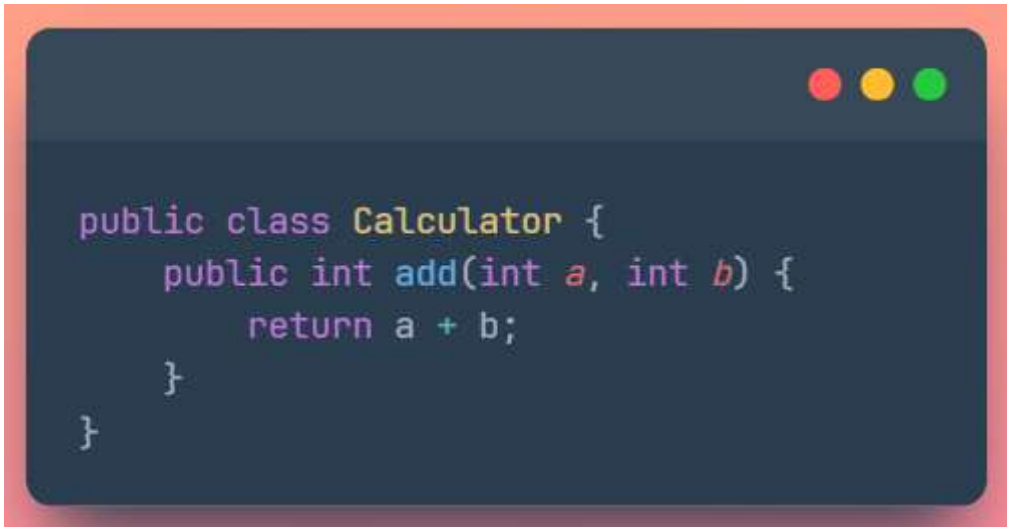
| Lý do | Ý nghĩa |
|---|---|
| Phát hiện lỗi sớm | Ngay tại mức hàm, class |
| Hỗ trợ refactoring | Đảm bảo sửa code mà không phá vỡ chức năng |
| Tăng tốc phát triển | Ít tốn thời gian fix bug về sau |
| Giảm chi phí bảo trì | Code ổn định hơn, ít lỗi |
| Hỗ trợ phát triển nhóm | Đảm bảo mỗi phần riêng lẻ hoạt động đúng |

## 4. Cách Viết Unit Test Trong Java

- Sử dụng thư viện:
  - Phổ biến nhất: **JUnit** (hiện tại là JUnit 5)
  - Hỗ trợ thêm: **Mockito** (dùng để mock các đối tượng phụ thuộc)

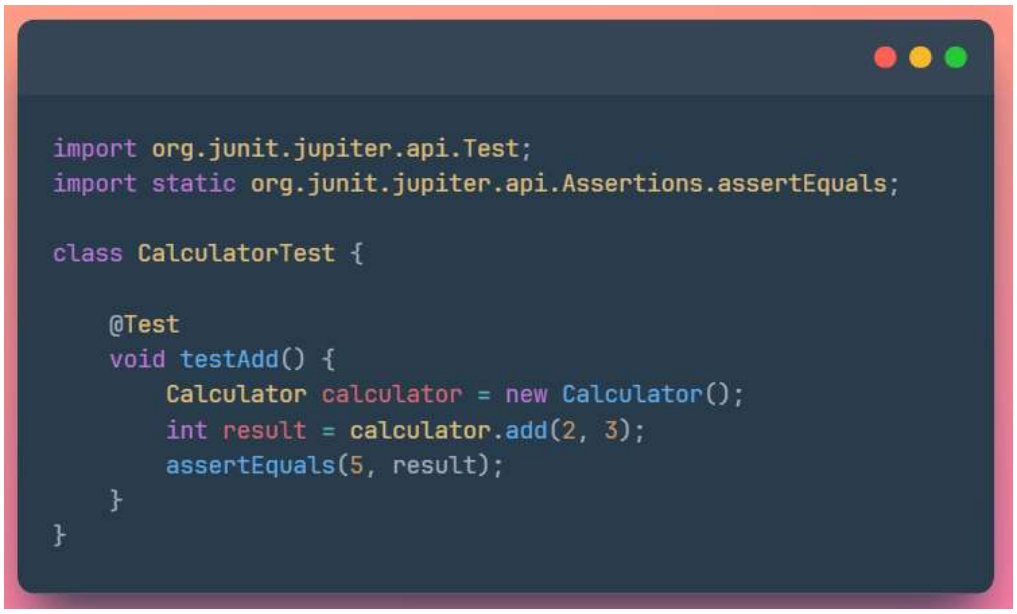
- Ví dụ đơn giản:

- Class cần kiểm tra:



```
public class Calculator {
    public int add(int a, int b) {
        return a + b;
    }
}
```

- Viết unit test với JUnit:



```
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.assertEquals;

class CalculatorTest {

    @Test
    void testAdd() {
        Calculator calculator = new Calculator();
        int result = calculator.add(2, 3);
        assertEquals(5, result);
    }
}
```

- Nếu **add(2, 3)** trả về 5, test pass.
- Nếu trả về khác 5, test fail.

5. **Cấu Trúc Một Unit Test Chuẩn (AAA - Arrange - Act - Assert)**

| **Bước** | **Mô tả** |
|---|---|
| Arrange | Chuẩn bị dữ liệu, đối tượng, môi trường cần thiết |
| Act | Gọi phương thức cần kiểm tra |
| Assert | Kiểm tra kết quả có đúng như mong đợi không |

6. **Các Thành Phần Phổ Biến Trong Unit Test**

- **@Test** – Đánh dấu phương thức là test case
- **@BeforeEach / @AfterEach** – Thiết lập hoặc dọn dẹp trước/sau mỗi test
- **@BeforeAll / @AfterAll** – Thiết lập hoặc dọn dẹp trước/sau toàn bộ test class
- **assertEquals, assertTrue, assertFalse, assertNotNull...** – Các phương thức kiểm tra kết quả

7. **Unit Test Khác Gì Với Các Kiểu Test Khác?**

| Loại test | Mức độ |
|---|---|
| Unit Test | Từng hàm, class |
| Integration Test | Kiểm tra sự kết hợp giữa các module |
| System Test | Kiểm tra toàn bộ hệ thống |
| Acceptance | Kiểm tra theo yêu cầu người dùng |

8. **Nguyên Tắc Viết Unit Test Tốt**
   - Độc lập, không phụ thuộc vào các test khác
   - Chạy nhanh
   - Rõ ràng, dễ đọc
   - Có thể chạy nhiều lần cho kết quả giống nhau
   - Đảm bảo bao phủ các trường hợp đúng, sai, biên

9. **Công Cụ Hỗ Trợ Unit Test Trong Java**
   - **JUnit:** Framework test phổ biến nhất
   - **Mockito:** Giả lập đối tượng phụ thuộc (mock)
   - **AssertJ:** Thư viện assertions nâng cao
   - **JaCoCo:** Đo độ bao phủ test (test coverage)
   - **Maven / Gradle:** Tích hợp chạy test khi build