

# Software Testing

BDT Gen 9

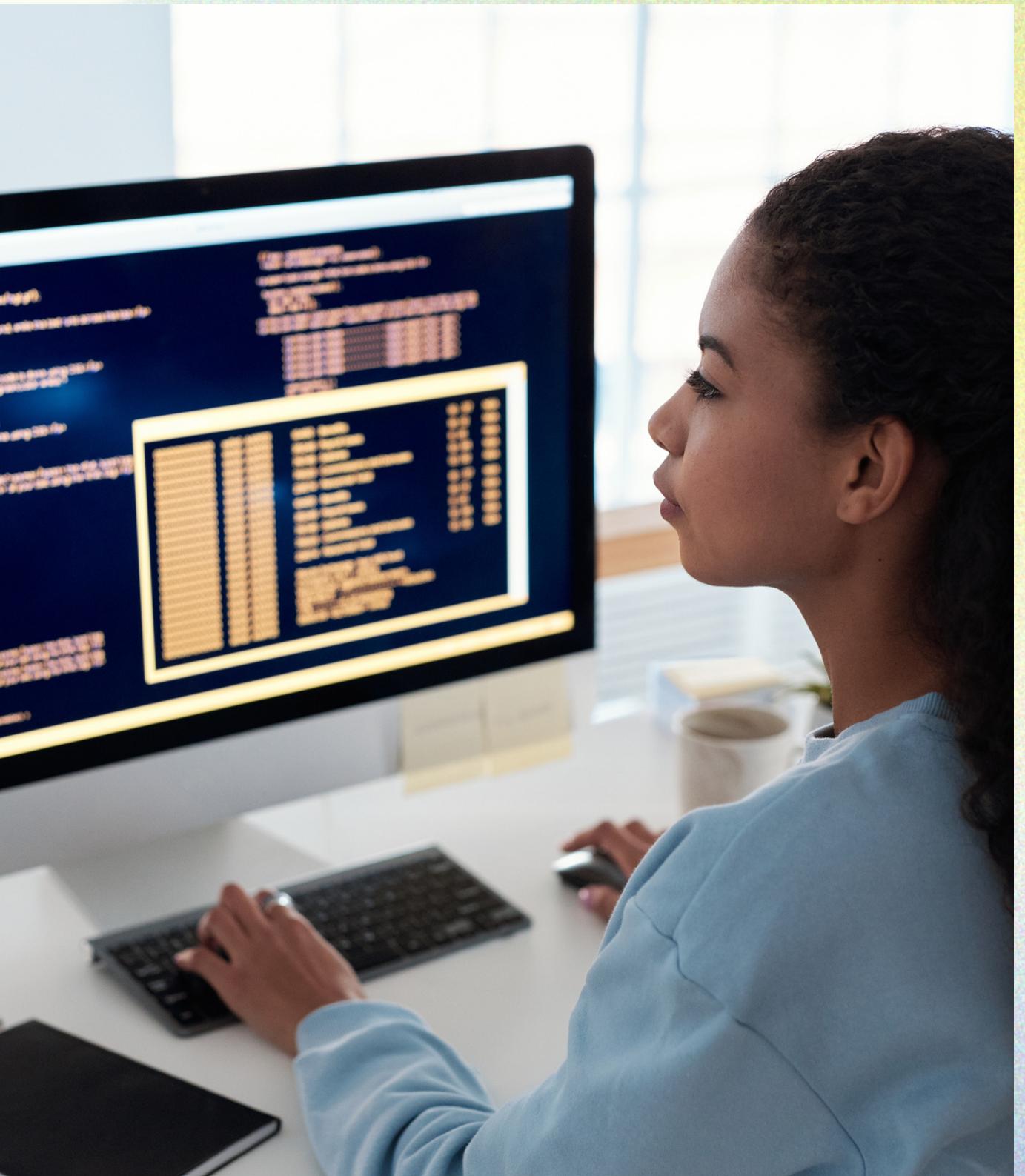
Nguyễn Quốc Hưng  
Software Engineer - One Mount

Software Testing

# Giới thiệu

## Software Testing

- **Software Testing – Kiểm thử phần mềm** là quá trình xác minh và xác thực xem phần mềm, ứng dụng không có lỗi, đáp ứng các yêu cầu kỹ thuật như được hướng dẫn bởi thiết kế, phát triển và đáp ứng các yêu cầu của người dùng một cách hiệu quả bằng cách xử lý tất cả các trường hợp ngoại lệ và biên.



# Định nghĩa

[Quay lại Trang Chương trình](#)



## Automation Testing - Kiểm thử tự động

Đơn giản là ta sẽ viết những kịch bản, script, code để tự động hoá việc kiểm thử.

Ưu điểm:

- Nhanh, hiệu quả
- Đảm bảo kiểm thử hết các kịch bản đề ra, không bỏ sót

Nhược điểm:

- Khi thay đổi code, requirement thì có thể phải viết lại kịch bản kiểm thử
- Thường script tự động sẽ không kiểm tra được tính đúng đắn về giao diện như font chữ, vị trí, ...

## Manual Testing - Kiểm thử thủ công

Ta phải thủ công kiểm thử phần mềm

Ưu điểm:

- Dễ dàng cho việc test giao diện
- Khi có thay đổi nhỏ manual testing không bị mất nhiều thời gian để thay đổi các trường hợp kiểm thử.

Nhược điểm:

- Sai sót yếu tố con người, kết quả thiếu tin cậy
- Không tái sử dụng được
- Khó triển khai performance testing

# Test Driven Development

- Các chức năng, yêu cầu về sản phẩm được chuyển biến về test
- **Test được viết trước khi code được tạo ra**
- Đơn giản, chu trình lặp lại

## Ưu điểm:

- Tăng chất lượng code, đặt việc đảm bảo tính năng luôn đúng lên đầu
- Tăng khả năng bảo trì, giảm thiểu chi phí khi thay đổi code
- Giảm nỗi sợ khi thay đổi code
- ...

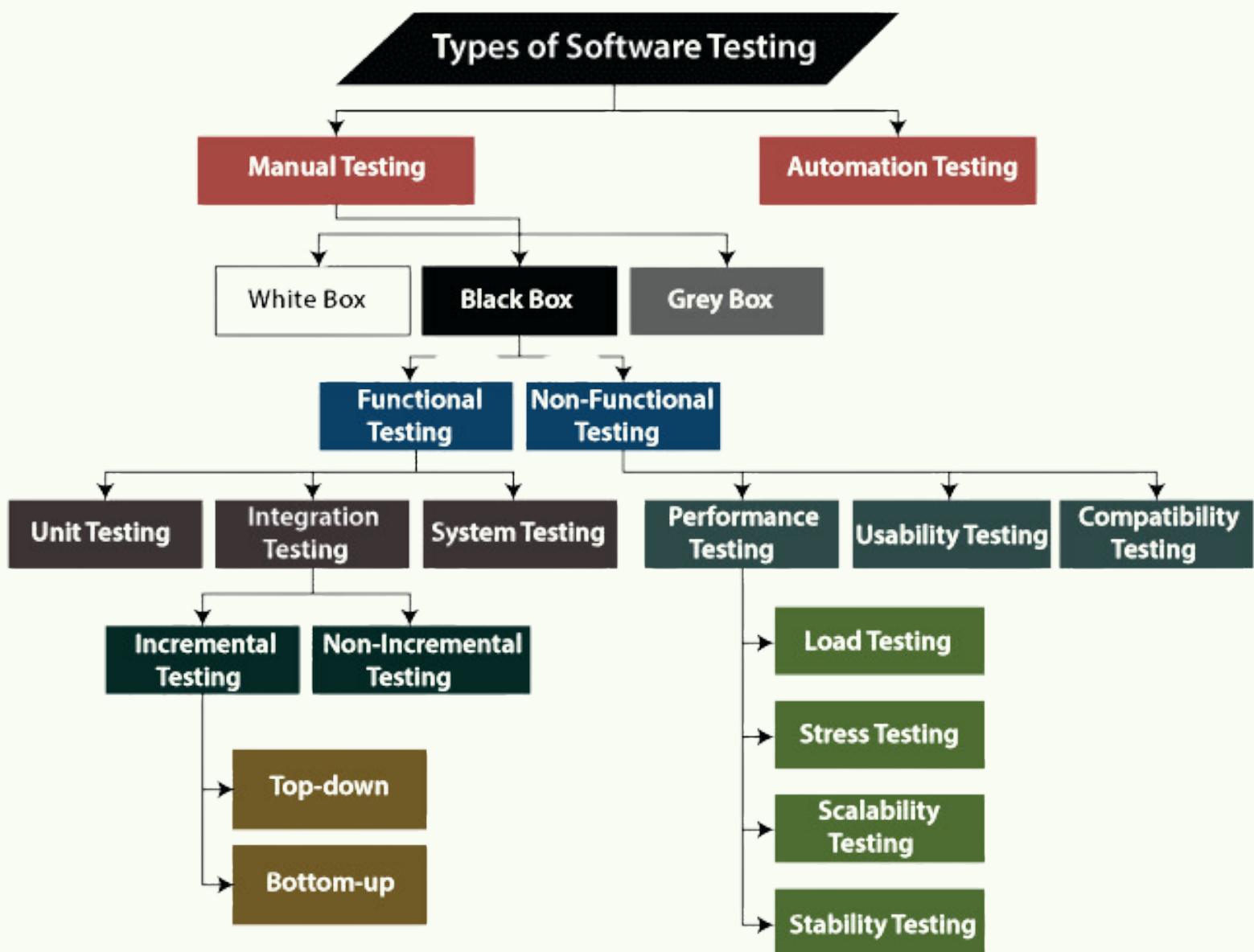


# Các loại Test bạn có thể đã nghe

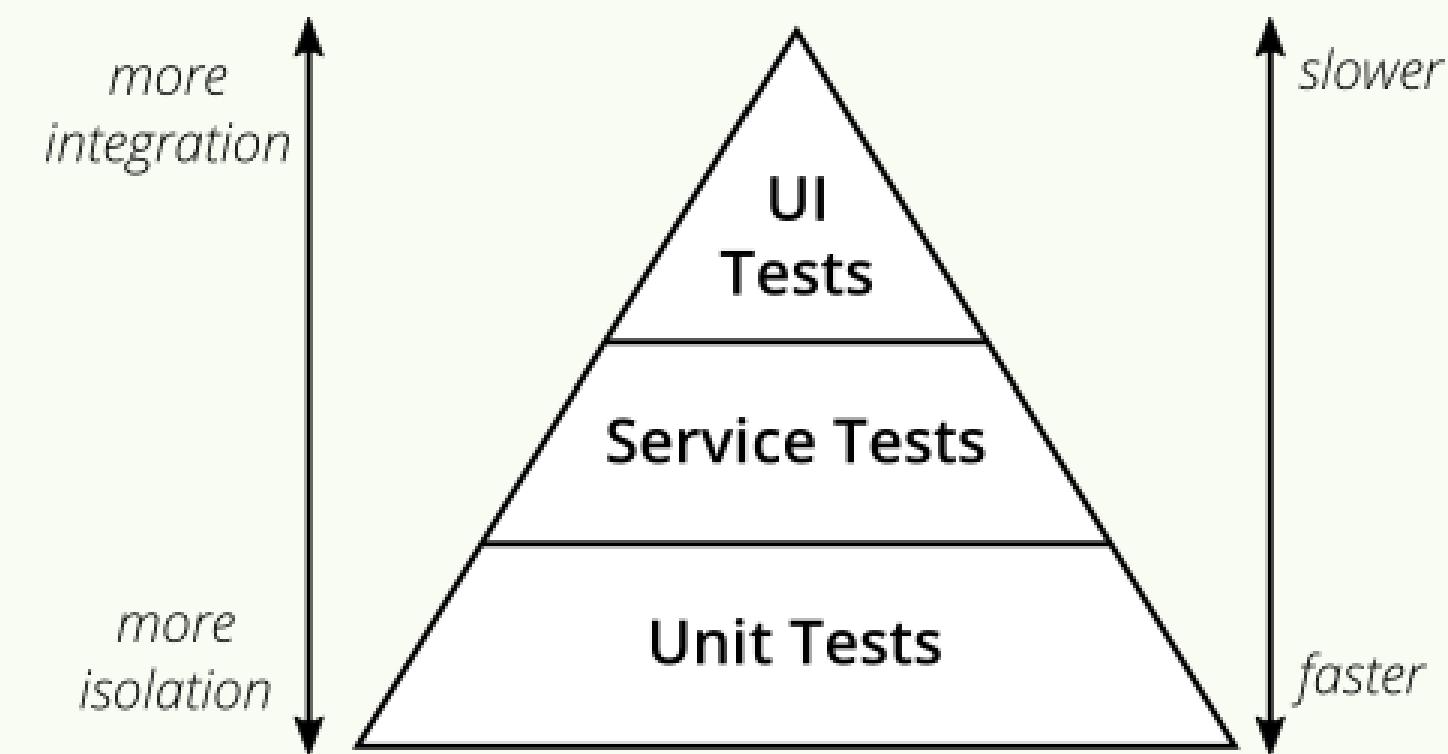
**Unit Test:** Kiểm thử 1 đoạn code, 1 hàm, 1 class

**Integration Test:** Các test có sự tương tác nhiều tầng, ví dụ tương tác với database, gọi api, ...

**End-To-End (e2e) test:** Những kịch bản test dài hơn, về cả 1 luồng giữa 2 thiết bị cuối (end)



**Performance Testing:** Kiểm thử về hiệu năng của sản phẩm



# Kỹ thuật kiểm thử phần mềm

[Quay lại Trang Chương trình](#)

Kiểm thử hộp đen (Black box)	Kiểm thử hộp trắng (White box)
<ul style="list-style-type: none"><li>• Ta không quan tâm code bên trong viết thế nào như thế nào, cấu trúc logic, ..</li><li>• Chỉ quan tâm là tính năng đã đúng với requirement của sản phẩm</li><li>• Có thể là sẽ kiểm tra từ phía GUI/Front end, ... từ việc ấn 1 nút và expect kết quả như thế nào</li></ul>	<ul style="list-style-type: none"><li>• Có thể không cần có GUI/Front end giao diện để test</li><li>• Từ code suy ra các trường hợp có thể xảy ra và có test đảm bảo điều đó</li></ul>

[Quay lại Trang Chương trình](#)

# Code Demo

Java JUnit về Unit Test

# JUnit Naming Convention

## ***JUnit naming conventions***

- "Test" suffix at the end of test classes names.

 LoginControllerTest  
 StudentControllerTest

- a test name should explain what the test does

```
@Test  
public void createStudentCourse() throws Exception {  
    fail();  
}
```

```
@Test  
public void test213123() {  
    fail();  
}
```

Given[ExplainYourInput]When[WhatIsDone]Then[ExpectedResult]

```
@Test  
public void GivenNullUsernameWhenCreateStudentThenShouldThrowException()
```

- You should use the "Test" suffix for test classes. The Maven automatically includes such classes in its test scope

# Tính độc lập

## *Test execution order*

JUnit assumes that all test methods can be executed in an arbitrary order.

- test code should not assume any order,
- tests should not depend on other tests.

`@FixMethodOrder(MethodSorters.NAME_ASCENDING)` annotation.

```
✓ testMockReturnsZero  
✓ testSpyReturnsStubbedValues2  
✓ testSpyReturnsRealValues  
✓ testSpyReturnsStubbedValues
```

```
✓ testMockReturnsZero  
✓ testSpyReturnsRealValues  
✓ testSpyReturnsStubbedValues  
✓ testSpyReturnsStubbedValues2
```

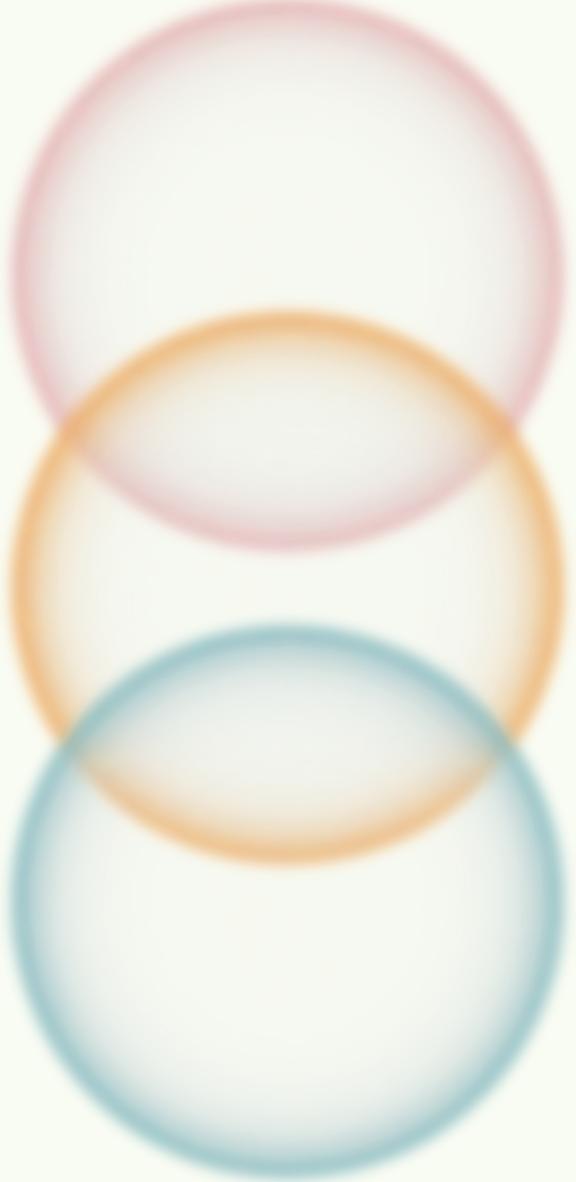
# Mock Testing

- Đôi lúc, ví dụ ta có các chức năng cần gọi xuống database, gọi sang 1 service khác

VD: Ứng dụng ta tích hợp thanh toán qua Momo, lúc này sẽ có chức năng gọi API sang Momo để làm gì đó

- Tuy nhiên khi test, ta không thể cài cắm luôn các thông tin bảo mật của môi trường thật được

=> Ta có các framework để mock các phần bên ngoài logic chính, đủ để giả lập các kịch bản của test



Quay lại Trang Chương trình

# Code Demo

Mock Test

# Performance Testing

- Đảm bảo hệ thống đạt được các yêu cầu về hiệu năng



- Các thông số như p90, p95, p99
- CCU – Concurrent user
- TPS (Transaction Per Second)
- Respond Time: Thời gian phản hồi của hệ thống, tính từ lúc request được gửi đi tới khi nhận được respond trả về từ hệ thống.

# Code Coverage

- Là những thông số đo lường là bao nhiêu dòng code đã được test kiểm thử, bao nhiêu hàm, bao nhiêu class
- Đây là những metric thông số để đảm bảo là code luôn được test đầy đủ

# Cảm ơn bạn!

Bạn có câu hỏi nào không?