**UNIT TEST**

1.Khái niệm Unit Test.

-Unit Test là một loại kiểm thử phần mềm trong đó các đơn vị hay thành phần riêng lẻ của phần mềm được kiểm thử. Kiểm thử đơn vị được thực hiện trong quá trình phát triển ứng dụng. Mục tiêu của Kiểm thử đơn vị là cô lập một phần code và xác minh tính chính xác của đơn vị đó.

**1.1 Khái niệm Unit.**

- Một Unit là một thành phần phần mềm nhỏ nhất mà ta có thể kiểm tra được như các hàm (Function), thủ tục (Procedure), lớp (Class), hoặc các phương thức (Method).

- Vì Unit được chọn để kiểm tra thường có kích thước nhỏ và chức năng hoạt động đơn giản, chúng ta không khó khăn gì trong việc tổ chức, kiểm tra, ghi nhận và phân tích kết quả kiểm tra nên việc phát hiện lỗi sẽ dễ dàng xác định nguyên nhân và khắc phục cũng tương đối dễ dàng vì chỉ khoanh vùng trong một Unit đang kiểm tra.

Mỗi UT sẽ gửi đi một thông điệp và kiểm tra câu trả lời nhận được đúng hay không, bao gồm:

* Các kết quả trả về mong muốn
* Các lỗi ngoại lệ mong muốn

- Các đoạn mã UT hoạt động liên tục hoặc định kỳ để thăm dò và phát hiện các lỗi kỹ thuật trong suốt quá trình phát triển, do đó UT còn được gọi là kỹ thuật kiểm nghiệm tự động. UT có các đặc điểm sau:

* Đóng vai trò như những người sử dụng đầu tiên của hệ thống.
* Chỉ có giá trị khi chúng có thể phát hiện các vấn đề tiềm ẩn hoặc lỗi kỹ thuật.

2. Công cụ thực hiện Unit test.

- Có nhiều công cụ thực hiện Unit Test như:

1. [Junit](https://junit.org/junit5/?ref=200lab.io): Junit là công cụ miễn phí được sử dụng cho ngôn ngữ lập trình Java. Nó cung cấp assertions để xác định phương pháp kiểm tra. Đầu tiên, công cụ này kiểm tra dữ liệu và sau đó chèn vào đoạn code.
2. [NUnit](https://nunit.org/?ref=200lab.io): NUnit là framework unit-testing được sử dụng rộng rãi cho tất cả các ngôn ngữ .NET. Nó là công cụ mã nguồn mở, có thể viết script theo cách thủ công. Ngoài ra, nó còn hỗ trợ các bài kiểm tra data-driven có thể chạy song song.
3. [JMockit](https://jmockit.github.io/index.html?ref=200lab.io): JMockit là công cụ Unit testing mã nguồn mở. Nó là công cụ code coverage với các số liệu đường (line) và đường dẫn (path). Công cụ này cung cấp Line coverage, Path Coverage, và Data Coverage.
4. [EMMA](http://emma.sourceforge.net/?ref=200lab.io): EMMA là một toolkit mã nguồn mở để phân tích và báo cáo code được viết bằng ngôn ngữ Java. Emma hỗ trợ các loại coverage như method, line, basic block. Nó dựa trên Java nên không có dependencies thư viện bên ngoài và có thể truy cập source code.
5. [PHPUnit](https://phpunit.de/?ref=200lab.io): PHPUnit là công cụ unit testing dành cho lập trình viên PHP. Nó sử dụng các phần nhỏ của code được gọi là các đơn vị và kiểm tra từng phần trong số chúng một cách riêng biệt. Công cụ này cũng cho phép các dev sử dụng các phương pháp pre-define assertion để khẳng định hệ thống hoạt động theo cách nhất định.

3. Các thuật ngữ khi sử dụng Unit Test.

1. **Assertion**: Là một phát biểu mô tả các công việc kiểm tra cần tiến hành, thí dụ: AreEqual(), IsTrue(), IsNotNull()… Mỗi một UT gồm nhiều assertion kiểm tra dữ liệu đầu ra, tính chính xác của các lỗi ngoại lệ ra và các vấn đề phức tạp khác như: – Sự tồn tại của một đối tượng – Điều kiện biên: Các giá trị có vượt ra ngoài giới hạn hay không – Thứ tự thực hiện của các luồng dữ liệu …
2. **Test Point**: Là một đơn vị kiểm tra nhỏ nhất, chỉ chứa đơn giản một assertion nhằm khẳng định tính đúng đắn của một chi tiết mã nào đó. Mọi thành viên dự án đều có thể viết một test point. Test Case: Là một tập hợp các test point nhằm kiểm tra một đặc điểm chức năng cụ thể, thí dụ toàn bộ giai đoạn người dùng nhập dữ liệu cho đến khi thông tin được nhập vào cơ sở dữ liệu. Trong nhiều trường hợp kiểm tra đặc biệt và khẩn cấp có thể không cần đến test case.
3. **Test Suite**: Là một tập hợp các test case định nghĩa cho từng module hoặc hệ thống con.
4. **Regression Testing (hoặc Automated Testing)**: Là phương pháp kiểm nghiệm tự động sử dụng một phần mềm đặc biệt. Cùng một loại dữ liệu kiểm tra giống nhau nhưng được tiến hành nhiều lần lặp lại tự động nhằm ngăn chặn các lỗi cũ phát sinh trở lại. Kết hợp Regression Testing với Unit Testing sẽ đảm bảo các đoạn mã mới vẫn đáp ứng yêu cầu thay đổi và các đoạn mã cũ sẽ không bị ảnh hưởng bởi các hoạt động bảo trì.
5. **Production Code**: Phần mã chính của ứng dụng được chuyển giao cho khách hàng.
6. **Unit Testing Code**: Phần mã phụ để kiểm tra mã ứng dụng chính, không được chuyển giao cho khách hàng.

4. Vòng đời của Unit Test.

UT test có 3 trạng thái cơ bản

* Fail (trạng thái lỗi)
* Ignore (tạm ngừng thực hiện)
* Pass (trạng thái làm việc)
* Toàn bộ UT được vận hành trong một hệ thống tách biệt. Thông thường, trạng thái của UT được biểu hiện bằng các màu khác nhau: màu xanh (pass), màu vàng (ignore) và màu đỏ (fail).

UT hoạt động hiệu quả khi

* Được vận hành lặp nhiều lần.
* Tự động hoàn toàn.
* Độc lập với UT khác.
* Test case đơn giản dễ đọc, dễ bảo trì.
* Sử dụng data dễ hiểu, dễ đọc.

**5.Ưu điểm và nhược điểm của Unit Test.**

**Ưu điểm của Unit Testing bao gồm:**

* Chúng ta có thể kiểm thử từng phần riêng rẽ mà không phải đợi các phần khác hoàn thành.
* Vấn đề được phát hiện càng sớm thì càng ít lỗi phức hợp xảy ra.
* Quá trình debug được thực hiện dễ dàng hơn.
* Các dev có thể nhanh chóng thực hiện những thay đổi đối với code base.
* Các dev cũng có thể sử dụng lại code, chuyển nó sang các dự án mới.
* Các lỗi được phát hiện sớm sẽ giúp giảm chi phí sửa lỗi. Bởi vì chi phí sửa lỗi trong giai đoạn **unit test**sẽ ít hơn so với những giai đoạn sau.

**Nhược điểm của Unit Testing gồm:**

* Các bài test sẽ không phát hiện ra tất cả các bug.
* Unit tests chỉ kiểm tra các tập dữ liệu và chức năng của nó. Chứ unit tests không tìm ra các lỗi tích hợp.
* Có thể cần viết nhiều dòng test code hơn để kiểm tra một dòng code.
* Unit testing cần đầu tư thời gian, công sức để học tập. Bởi vì đôi khi bạn cần học cách sử dụng các công cụ phần mềm tự động.

**6. Thiết kế Unit test**

Mỗi UT đều được tiết kế theo trình tự sau:

* Thiết lập các điều kiện cần thiết: khởi tạo các đối tượng, xác định tài nguyên cần thiết, xây dựng các dữ liệu giả…
* Triệu gọi các phương thức cần kiểm tra.
* Kiểm tra sự hoạt động đúng đắn của các phương thức.
* Dọn dẹp tài nguyên sau khi kết thúc kiểm tra.

Ví dụ về Unit Test Java

Bước 1 : Tại 1 class ở …/src/main/java

A black screen with colorful text

Description automatically generated

Bước 2: Tạo 1 class để test ở ../src/test/java

A screen shot of a computer program

Description automatically generated

IntelliJ hỗ trợ chạy từng test case hoặc cả class test thông qua các nút hiện kế bên số dòng.

Khi chạy test, nó sẽ kiểm tra tất cả các trường hợp, nó chỉ cho pass khi tất cả đều thành công, chỉ cần có 1 test sai xem như test fail.

Khi có sai sót xảy ra sẽ hiện thông báo lên màn hình là test nào sai và sai cái gì.

A screenshot of a computer program

Description automatically generated

Kết quả sẽ in ra Expected value: Giá trị mà chúng ta mong đợi khối lệnh trả về Actual value: Giá trị thực tế mà khối lệnh trả về

Báo lỗi test 2 giá trị thực tế là 2 mà câu lệnh trả về 3