**MỤC TIÊU:**

Kết thúc bài thực hành này bạn có thể nắm được tổng quan các khái niệm và cách ứng dụng của từng mức độ kiểm thử phần mềm

* Component Testing - Kiểm thử thành phần
* Smoke Testing - Kiểm thử khói
* Sanity Testing - Kiểm thử độ tỉnh táo
* Regression Testing - Kiểm thử hồi quy
* Integration Testing - Kiểm thử tích hợp
* System Testing - Kiểm thử hệ thống
* Acceptance Testing - Kiểm thử chấp nhận

# PHẦN I

## Bài 1 (2 điểm)

Hãy liệt kê các trường hợp cần kiểm thử trong function test01

public class Test {

public void test01(int a, int b) {

String s = (a % b == 0) ? "a chia het cho b" : "a khong chia het cho b";

System.out.println(s);

}

}

Trả lời:

Dưới đây là các trường hợp cần kiểm thử trong hàm test01:

a chia hết cho b: Ở trường hợp này, kết quả mong đợi là chuỗi "a chia hết cho b".

Ví dụ: test01(10, 5)

a không chia hết cho b: Ở trường hợp này, kết quả mong đợi là chuỗi "a không chia hết cho b".

Ví dụ: test01(10, 3)

b = 0: Trường hợp này kiểm tra khi b = 0. Điều này xảy ra khi chúng ta chia một số cho 0, không có kết quả xác định.

Ví dụ: test01(10, 0)

a = 0: Trường hợp này kiểm tra khi a = 0. Chúng ta muốn đảm bảo rằng phép chia được thực hiện đúng khi a bằng 0.

b < 0: Trường hợp này kiểm tra khi b là số âm. Chúng ta cần đảm bảo rằng hàm xử lý đúng trong trường hợp này.

Ví dụ: test01(10, -5)

a và b đều là số âm: Trường hợp này kiểm tra khi cả a và b đều là số âm. Chúng ta cần đảm bảo rằng hàm xử lý đúng trong trường hợp này.

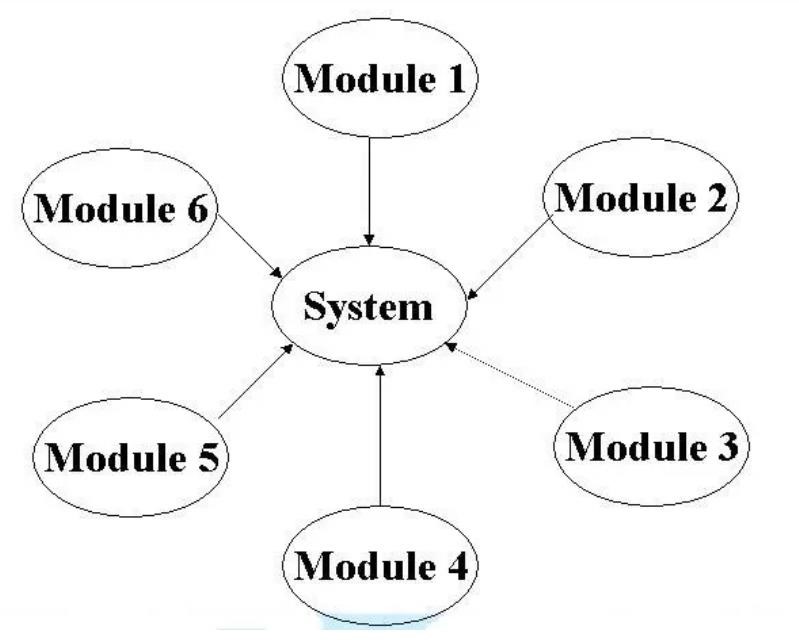
Ví dụ: test01(-10, -5)

a và b = 0: Trường hợp này kiểm tra khi cả a và b đều bằng 0. Chúng ta muốn đảm bảo rằng phép chia không gây ra lỗi và kết quả đúng.

Ví dụ: test01(0, 0)

## Bài 2 (2 điểm)

Hình bên dưới là phương pháp kiểm thử gì ?



Hình trên là phương pháp kiểm thử(Kiểm thức tích hợp)**Intergration Testing**

**Bài 3 (1 điểm)**

Giảng viên cho thêm

# PHẦN II

## Bài 4 (2 điểm)

Giả định khách hàng thay đổi requirement ở bất cứ thời điểm nào. Vì thế để thỏa mãn sự thay đổi của khách hàng, developer cần thay đổi logic và code của họ. Sau khi developer thay đổi code, chúng ta cần thực thi mức kiểm thử nào ?

Trả lời:

Sau khi developer thay đổi code để đáp ứng yêu cầu mới của khách hàng, ta cần thực hiện kiểm thử để đảm bảo rằng các thay đổi không gây ra lỗi hoặc tác động tiêu cực đến hệ thống. Kiểm thử được thực hiện để đảm bảo tính đúng đắn, hiệu suất và khả năng chịu tải của hệ thống.

Mức kiểm thử cần thực hiện sau khi thay đổi code phụ thuộc vào phạm vi và tầm quan trọng của các thay đổi. Dưới đây là một số mức kiểm thử phổ biến mà chúng ta có thể áp dụng:

Kiểm thử đơn vị (Unit testing): Kiểm thử đơn vị được thực hiện để kiểm tra từng phần riêng lẻ của mã nguồn, đảm bảo rằng các thành phần hoạt động chính xác theo dự kiến. Đây là mức kiểm thử cơ bản và phải được thực hiện sau mỗi thay đổi code.

Kiểm thử tích hợp (Integration testing): Kiểm thử tích hợp được thực hiện để kiểm tra tích hợp giữa các thành phần trong hệ thống. Nó giúp đảm bảo rằng các thành phần hoạt động cùng nhau một cách chính xác sau khi thay đổi. Khi thay đổi code, kiểm thử tích hợp nên được thực hiện để đảm bảo tính liên kết và tương tác đúng đắn giữa các thành phần.

Kiểm thử chức năng (Functional testing): Kiểm thử chức năng kiểm tra hệ thống từ góc độ chức năng, đảm bảo rằng nó hoạt động đúng theo yêu cầu của khách hàng. Sau khi thay đổi code, kiểm thử chức năng cần được thực hiện để đảm bảo tính hoàn thiện và chính xác của các chức năng mới.

Kiểm thử hiệu năng (Performance testing): Kiểm thử hiệu năng được thực hiện để đánh giá hiệu suất của hệ thống, đảm bảo rằng nó hoạt động ổn định và đáp ứng tải công việc được yêu cầu. Nếu thay đổi code ảnh hưởng đến hiệu suất, kiểm thử hiệu năng cần được thực hiện để đảm bảo rằng hệ thống không gặp vấn đề đáng kể.

Ngoài ra còn nhiều mức kiểm thử khác như kiểm thử bảo mật (Security testing), kiểm thử tải (Load testing) và kiểm thử phi chức năng (Non-functional testing) tùy thuộc vào yêu cầu và phạm vi của dự án. Tuy nhiên, việc thực hiện các mức kiểm thử phụ thuộc cũng phần nào vào quy trình phát triển phần mềm và các tiêu chí riêng của dự án.

## Bài 5 (2 điểm)

Nếu bạn có phần mềm máy tính có khả năng phép cộng, trừ, nhân, chia thì khi ra phiên bản mới bạn cần thực thi mức kiểm thử nào ? Và nêu một số trường hợp bạn có thể kiểm thử dựa trên mức kiểm thử bạn chọn.

Trả lời:

Khi ra phiên bản mới của phần mềm máy tính có khả năng phép cộng, trừ, nhân, chia, bạn cần thực hiện các mức kiểm thử sau:

Kiểm thử đơn vị (Unit testing): Kiểm thử đơn vị được thực hiện để kiểm tra từng thành phần riêng lẻ của phần mềm, đảm bảo rằng họat động chính xác, bao gồm các phép tính cộng, trừ, nhân và chia. Các trường hợp kiểm thử đơn vị có thể bao gồm:

Kiểm tra phép cộng hai số nguyên dương.

Kiểm tra phép trừ khi số đầu vào âm.

Kiểm tra phép nhân với số 0.

Kiểm tra phép chia khi số bị chia là 0.

Kiểm thử tích hợp (Integration testing): Kiểm thử tích hợp được thực hiện để kiểm tra tích hợp giữa các thành phần trong phần mềm, đảm bảo rằng chúng hoạt động cùng nhau một cách chính xác sau khi ra phiên bản mới. Các trường hợp kiểm thử tích hợp có thể bao gồm:

Kiểm tra tích hợp giữa phép cộng và phép trừ.

Kiểm tra tích hợp giữa phép nhân và phép chia.

Kiểm thử chức năng (Functional testing): Kiểm thử chức năng được thực hiện để kiểm tra các chức năng của phần mềm, bao gồm cả phép cộng, trừ, nhân và chia. Các trường hợp kiểm thử chức năng có thể bao gồm:

Kiểm tra phép cộng hai số âm.

Kiểm tra phép trừ khi số đầu vào là số nguyên dương.

Kiểm tra phép nhân với số âm.

Kiểm tra phép chia khi số bị chia không chia hết.

Kiểm thử hiệu năng (Performance testing): Kiểm thử hiệu năng được thực hiện để đánh giá hiệu suất của phần mềm, đảm bảo rằng nó hoạt động ổn định và có thể xử lý một lượng lớn phép tính một cách hiệu quả. Các trường hợp kiểm thử hiệu năng có thể bao gồm:

Kiểm tra khả năng xử lý của phần mềm khi thực hiện nhiều phép tính cộng, trừ, nhân và chia cùng lúc.

Tuy nhiên, các trường hợp kiểm thử cụ thể có thể khác nhau tùy thuộc vào yêu cầu và phạm vi của phần mềm máy tính của bạn. Việc lựa chọn mức kiểm thử và các trường hợp kiểm thử cần được thực hiện công phu và dựa trên việc hiểu rõ yêu cầu và logic của phần mềm.

**Bài 6 (1 điểm)**

# Giảng viên cho thêm