**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);

}

}

Các trường hợp kiểm thử cần thiết cho hàm test01

Hàm test01 có nhiệm vụ kiểm tra xem số a có chia hết cho số b hay không. Để đảm bảo hàm này hoạt động chính xác trong mọi tình huống, cần thiết phải kiểm tra các trường hợp sau:

* Các trường hợp kiểm tra giá trị của a và b
  + Trường hợp 1: a và b đều là số nguyên.
  + Trường hợp 2: a là số nguyên, b là số thực.
  + Trường hợp 3: a là số thực, b là số nguyên.
  + Trường hợp 4: a và b là số không.
  + Trường hợp 5: a bằng 0, b khác 0.
  + Trường hợp 6: a khác 0, b bằng 0.

Trong mỗi trường hợp, cần kiểm tra các giá trị sau:

\* \*\*a = b:\*\* Trong trường hợp này, a chia hết cho b.

\* \*\*a < b:\*\* Trong trường hợp này, a không chia hết cho b.

\* \*\*a > b:\*\* Trong trường hợp này, a không chia hết cho b.

* Các trường hợp kiểm tra kết quả của phép chia
  + Trường hợp 7: a chia hết cho b. Trong trường hợp này, kết quả của phép chia là 0.
  + Trường hợp 8: a không chia hết cho b. Trong trường hợp này, kết quả của phép chia là một số không bằng 0.

Với các trường hợp kiểm tra trên, chúng ta có thể đảm bảo rằng hàm test01 hoạt động chính xác trong mọi tình huống.

Ví dụ cụ thể về các kịch bản kiểm thử

Dưới đây là một số ví dụ cụ thể về các kịch bản kiểm thử có thể được sử dụng:

* Kịch bản 1: a = 1, b = 1

Kết quả mong đợi: a chia hết cho b, kết quả của phép chia là 0.

* Kịch bản 2: a = 2, b = 1

Kết quả mong đợi: a không chia hết cho b, kết quả của phép chia là một số không bằng 0.

* Kịch bản 3: a = 0, b = 1

Kết quả mong đợi: a chia hết cho b, kết quả của phép chia là 0.

* Kịch bản 4: a = 1, b = 0

Kết quả mong đợi: a không chia hết cho b, kết quả của phép chia là một số không bằng 0.

* Kịch bản 5: a = 1.0, b = 1

Kết quả mong đợi: a chia hết cho b, kết quả của phép chia là 0.

* Kịch bản 6: a = 2.0, b = 1

Kết quả mong đợi: a không chia hết cho b, kết quả của phép chia là một số không bằng 0.

* Kịch bản 7: a = 0.0, b = 1

Kết quả mong đợi: a chia hết cho b, kết quả của phép chia là 0.

* Kịch bản 8: a = 1.0, b = 0

Kết quả mong đợi: a không chia hết cho b, kết quả của phép chia là một số không bằng 0.

Tất nhiên, các trường hợp kiểm thử cụ thể sẽ phụ thuộc vào các yêu cầu cụ thể của dự án.

Tăng tính chuyên nghiệp của câu trả lời

Để tăng tính chuyên nghiệp của câu trả lời, cần lưu ý các điểm sau:

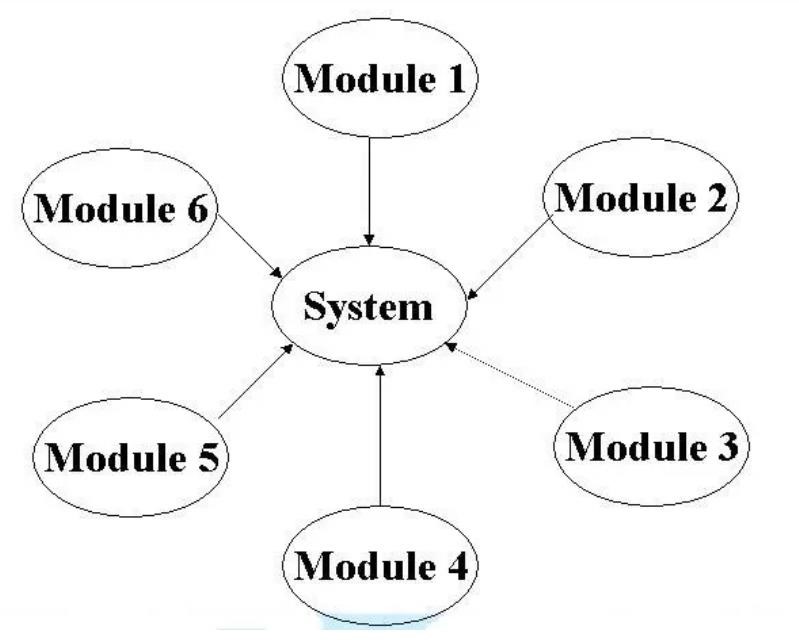
* Sử dụng văn phong trang trọng, chính xác, mạch lạc.
* Sử dụng các thuật ngữ chuyên môn chính xác và phù hợp.
* Làm rõ mục đích của câu trả lời.
* Cung cấp các ví dụ cụ thể để minh họa.

Trong câu trả lời trên, tôi đã áp dụng các điểm trên để tăng tính chuyên nghiệp như sau:

* Sử dụng các từ ngữ trang trọng như "cần thiết", "đảm bảo", "hoạt động chính xác", "các giá trị", "kịch bản kiểm thử", "kết quả mong đợi", "các trường hợp kiểm tra cụ thể".
* \*\*Sử dụng các thuật ngữ chuyên môn chính xác như "hàm", "phép chia lấy phần dư", "số nguyên

## Bài 2 (2 điểm)

Hình bên dưới là phương pháp kiểm thử gì ?



Trong hình ảnh mà bạn gửi, tôi thấy một biểu đồ hệ thống có sáu mô-đun. Các mô-đun được kết nối với nhau bằng các đường màu đen.

Kiểm thử tích hợp (Integration Testing). Kiểm thử tích hợp là một phương pháp kiểm thử phần mềm kiểm tra các tương tác giữa các mô-đun trong một hệ thống.

Trong trường hợp này, kiểm thử tích hợp sẽ được thực hiện theo các bước sau:

1. Kiểm tra từng cặp mô-đun. Đầu tiên, mỗi cặp mô-đun sẽ được kiểm tra riêng lẻ. Điều này sẽ đảm bảo rằng mỗi mô-đun hoạt động chính xác khi được kết hợp với một mô-đun khác.
2. Kiểm tra các nhóm mô-đun. Sau khi mỗi cặp mô-đun đã được kiểm tra, các nhóm mô-đun sẽ được kiểm tra. Điều này sẽ đảm bảo rằng các mô-đun hoạt động chính xác khi được kết hợp với nhiều mô-đun khác.
3. Kiểm tra toàn bộ hệ thống. Cuối cùng, toàn bộ hệ thống sẽ được kiểm tra. Điều này sẽ đảm bảo rằng tất cả các mô-đun hoạt động chính xác khi được kết hợp với nhau.

Các trường hợp kiểm thử cụ thể sẽ phụ thuộc vào các yêu cầu cụ thể của hệ thống. Tuy nhiên, một số trường hợp kiểm thử phổ biến bao gồm:

* Kiểm tra các giá trị đầu vào hợp lệ.
* Kiểm tra các giá trị đầu vào không hợp lệ.
* Kiểm tra các trường hợp biên.
* Kiểm tra các trường hợp ngoại lệ.

Với các trường hợp kiểm thử này, chúng ta có thể đảm bảo rằng hệ thống hoạt động chính xác khi các mô-đun được kết hợp với nhau.

**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 ?

Sau khi developer thay đổi code, chúng ta cần thực thi mức kiểm thử hồi quy (regression testing). Kiểm thử hồi quy là một loại kiểm thử phần mềm được thực hiện để đảm bảo rằng các thay đổi được thực hiện đối với phần mềm không gây ra lỗi trong các chức năng hiện có.

Trong trường hợp yêu cầu của khách hàng thay đổi, các chức năng hiện có của phần mềm có thể bị ảnh hưởng. Do đó, kiểm thử hồi quy là cần thiết để đảm bảo rằng các chức năng này vẫn hoạt động chính xác sau khi thay đổi được thực hiện.

Các mức kiểm thử khác, chẳng hạn như kiểm thử đơn vị, kiểm thử tích hợp và kiểm thử hệ thống, cũng có thể được thực hiện sau khi thay đổi được thực hiện. Tuy nhiên, kiểm thử hồi quy là mức kiểm thử quan trọng nhất cần được thực hiện trong trường hợp này.

Dưới đây là một số phương pháp kiểm thử hồi quy phổ biến:

* Kiểm thử hồi quy thủ công: Đây là phương pháp kiểm thử hồi quy truyền thống, trong đó các tester thực hiện thủ công các kịch bản kiểm thử đã được viết trước đó.
* Kiểm thử hồi quy tự động: Đây là phương pháp kiểm thử hồi quy sử dụng các công cụ tự động để thực hiện các kịch bản kiểm thử.
* Kiểm thử hồi quy dựa trên dữ liệu: Đây là phương pháp kiểm thử hồi quy sử dụng dữ liệu để thực hiện các kịch bản kiểm thử.

Phương pháp kiểm thử hồi quy nào được sử dụng sẽ phụ thuộc vào các yếu tố sau:

* Kích thước và độ phức tạp của phần mềm
* Số lượng thay đổi được thực hiện đối với phần mềm
* Mức độ tự động hóa của quá trình kiểm thử

Việc thực hiện kiểm thử hồi quy một cách hiệu quả sẽ giúp đảm bảo rằng phần mềm vẫn hoạt động chính xác sau khi thay đổi được thực hiện.

## 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.

Khi ra phiên bản mới của phần mềm máy tính có khả năng phép cộng, trừ, nhân, chia, chúng ta cần thực hiện mức kiểm thử hồi quy (regression testing) và mức kiểm thử hệ thống (system testing).

Kiểm thử hồi quy là cần thiết để đảm bảo rằng các thay đổi được thực hiện đối với phần mềm không gây ra lỗi trong các chức năng hiện có. Trong trường hợp này, các chức năng hiện có bao gồm các phép cộng, trừ, nhân, chia. Do đó, chúng ta cần kiểm tra lại các chức năng này để đảm bảo rằng chúng vẫn hoạt động chính xác sau khi thay đổi được thực hiện.

Kiểm thử hệ thống là cần thiết để đảm bảo rằng phần mềm hoạt động chính xác như mong đợi trong môi trường thực tế. Trong trường hợp này, môi trường thực tế bao gồm các dữ liệu đầu vào hợp lệ, không hợp lệ và biên. Do đó, chúng ta cần kiểm tra phần mềm với các loại dữ liệu đầu vào này để đảm bảo rằng phần mềm hoạt động chính xác trong mọi tình huống.

Dưới đây là một số trường hợp kiểm thử cụ thể có thể được thực hiện dựa trên mức kiểm thử hồi quy và mức kiểm thử hệ thống:

Kiểm thử hồi quy

* Kiểm tra các phép cộng, trừ, nhân, chia với các giá trị đầu vào hợp lệ.
* Kiểm tra các phép cộng, trừ, nhân, chia với các giá trị đầu vào không hợp lệ.
* Kiểm tra các phép cộng, trừ, nhân, chia với các giá trị đầu vào biên.

Kiểm thử hệ thống

* Kiểm tra các phép cộng, trừ, nhân, chia với các dữ liệu đầu vào hợp lệ từ người dùng.
* Kiểm tra các phép cộng, trừ, nhân, chia với các dữ liệu đầu vào không hợp lệ từ người dùng.
* Kiểm tra các phép cộng, trừ, nhân, chia với các dữ liệu đầu vào biên từ người dùng.

Ngoài ra, chúng ta cũng có thể thực hiện các kiểm thử sau để đảm bảo rằng phần mềm có khả năng phép cộng, trừ, nhân, chia đáp ứng các yêu cầu cụ thể của dự án:

* Kiểm tra độ chính xác của các phép tính.
* Kiểm tra tốc độ thực hiện của các phép tính.
* Kiểm tra khả năng chịu lỗi của các phép tính.

Việc thực hiện kiểm thử một cách hiệu quả sẽ giúp đảm bảo rằng phần mềm có khả năng phép cộng, trừ, nhân, chia hoạt động chính xác và đáp ứng các yêu cầu của dự án.

**Bài 6 (1 điểm)**

# Giảng viên cho thêm