**LAB3: CÁC MỨC ĐỘ TRONG KIỂM THỬ PHẦN MỀM**

**Bài 1**

**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);**

**}**

**}**

1. Trường hợp a chia hết cho b:

- Đầu vào: a = 10, b = 2

- Kết quả mong đợi: "a chia hết cho b"

2. Trường hợp a không chia hết cho b:

- Đầu vào: a = 7, b = 3

- Kết quả mong đợi: "a không chia hết cho b"

3. Trường hợp khi giá trị của biến `b` là số không (0):

- Đầu vào: a = 15, b = 0

- Kết quả mong đợi: (Phép chia cho 0 không xác định trong toán học)

4. Trường hợp khi b là số âm:

- Đầu vào: a = 12, b = -3

- Kết quả mong đợi: "a chia hết cho b" (vì 12 chia hết cho -3)

- Đầu vào: a = 12, b = -5

- Kết quả mong đợi: "a không chia hết cho b" (vì 12 không chia hết cho -5)

5. Trường hợp khi giá trị của biến `a` là số không (0):

- Đầu vào: a = 0, b = 5

- Kết quả mong đợi: "a chia hết cho b" (vì 0 chia hết cho bất kỳ số khác không nào)

6. Trường hợp khi a là số âm:

- Đầu vào: a = -8, b = 2

- Kết quả mong đợi: "a chia hết cho b" (vì -8 chia hết cho 2)

- Đầu vào: a = -9, b = 2

- Kết quả mong đợi: "a không chia hết cho b" (vì -9 không chia hết cho 2)

7. Trường hợp khi a bằng b:

- Đầu vào: a = 5, b = 5

- Kết quả mong đợi: "a chia hết cho b"

8. Trường hợp khi cả a và b đều bằng 0:

- Đầu vào: a = 0, b = 0

- Kết quả mong đợi: (Phép chia cho 0 không xác định, trường hợp đặc biệt.)

**Bài 2**

**Hình bên dưới là phương pháp kiểm thử gì ?**

A diagram of a system

Description automatically generated

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

**Bài 3 Cho biết các mức kiểm thử phần mềm và giải thích từng mức** **kiểm thử Đối tượng kiểm thử mỗi mức là gì?**

1. Kiểm thử đơn vị (Unit testing): Đây là mức kiểm thử tập trung vào việc kiểm tra từng đơn vị code nhỏ nhất của phần mềm, chẳng hạn như hàm hoặc module. Mục tiêu của kiểm thử đơn vị là xác minh tính chính xác và hoạt động đúng của các thành phần cụ thể này.

Đối tượng kiểm thử: Hàm, phương thức, lớp hoặc module nhỏ nhất.

Mục tiêu: Kiểm thử từng đơn vị code nhỏ nhất để đảm bảo rằng nó hoạt động đúng.

2. Kiểm thử tích hợp (Integration testing): Mức kiểm thử này tập trung vào việc kiểm tra sự tương tác giữa các thành phần khác nhau trong hệ thống. Nó giúp xác minh rằng các thành phần đã được tích hợp lại với nhau theo cách mong muốn và hoạt động ổn định.

Đối tượng kiểm thử: Kết hợp các đơn vị đã kiểm thử thành các nhóm lớn hơn hoặc toàn bộ ứng dụng.

Mục tiêu: Đảm bảo rằng các đơn vị đã kiểm thử hoạt động đúng khi kết hợp và giao tiếp với nhau.

3. Kiểm thử hệ thống là (System Testing): tập trung vào việc đảm bảo rằng toàn bộ hệ thống làm việc như dự kiến và đáp ứng đúng các yêu cầu chức năng và phi chức năng. Mục tiêu của kiểm thử hệ thống là kiểm tra và đánh giá hệ thống toàn diện từ góc độ người sử dụng cuối cùng.

Đối tượng kiểm thử: Toàn bộ ứng dụng hoặc hệ thống.

Mục tiêu: Đảm bảo rằng ứng dụng hoặc hệ thống làm việc như dự kiến và đáp ứng đúng các yêu cầu chức năng và phi chức năng.

4. Kiểm thử hiệu năng (Performance testing): Mức kiểm thử này tập trung vào việc đánh giá hiệu suất và khả năng chịu tải của phần mềm trong các tình huống khác nhau. Nó đảm bảo rằng hệ thống có thể xử lý được công việc với hiệu suất cao và không gặp vấn đề về tải.

Đối tượng kiểm thử: Hệ thống hoặc các thành phần quan trọng của nó.

Mục tiêu: Đánh giá và đảm bảo rằng hệ thống đáp ứng đúng với các yêu cầu hiệu suất, tải và thời gian phản hồi.

5. Kiểm thử bảo mật (Security testing): Mức kiểm thử này nhằm kiểm tra tính bảo mật của phần mềm, xác minh rằng hệ thống không có lỗ hổng nào cho các cuộc tấn công từ bên ngoài hoặc từ người dùng không được ủy quyền.

Đối tượng kiểm thử: Hệ thống, các thành phần quan trọng hoặc tính năng đặc biệt liên quan đến bảo mật.

Mục tiêu: Đảm bảo rằng hệ thống an toàn và không có lỗ hổng bảo mật.

6. Kiểm thử tự động (Automation testing): Đây là quá trình sử dụng công cụ và kịch bản kiểm thử để tự động hóa việc kiểm tra phần mềm. Nó giúp tiết kiệm thời gian và tăng tính nhất quán trong việc kiểm tra lại các chức năng đã được xác minh trước đó.

**Bài 4** **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 để thỏa mãn sự thay đổi của khách hàng, chúng ta cần thực hiện kiểm thử để đảm bảo rằng các tính năng mới hoạt động như mong đợi và không gây ra lỗi hoặc ảnh hưởng tiêu cực tới các tính năng hiện có. Mức kiểm thử cần được xác định dựa trên phạm vi và quy mô của sự thay đổi.

Sau khi developer thay đổi code, chúng ta cần thực thi mức kiểm thử Kiểm tra tích hợp (Integration testing) Kiểm tra sự tương tác giữa các thành phần, module hoặc dịch vụ mới và các thành phần, module hoặc dịch vụ hiện có để đảm bảo tính tương thích và không gây ra xung đột.

**Bài 5** **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, bạn cần thực hiện các mức kiểm thử sau:

**1.** Kiểm thử đơn vị (Unit testing): Kiểm tra từng thành phần riêng lẻ của mã nguồn để đảm bảo chúng hoạt động đúng và không gây ảnh hưởng tới các thành phần khác.

**2.** Kiểm thử tích hợp (Integration testing): Kiểm tra sự tương tác giữa các thành phần đã được kiểm tra ở mức unit testing để xác minh rằng chúng hoạt động hợp lý khi được kết hợp lại.

**3.** Kiểm thử chức năng (Functional testing): Đây là loại kiểm thử nhằm xác minh rằng toàn bộ chức năng của phần mềm hoạt động theo yêu cầu và mong muốn.

**4.** Kiểm thử hiệu suất (Performance testing): Đánh giá hiệu suất của phần mềm trong điều kiện tải cao để xem liệu nó có hoạt động ổn định và có khả năng xử lý dữ liệu lớn hay không.

**5.** Kiểm thử giao diện người dùng (User interface testing): Xác minh tính nhất quán và sự dễ sử dụng của giao diện người dùng, đảm bảo rằng các chức năng và tương tác với người dùng hoạt động một cách chính xác.

**6.** Kiểm thử bảo mật (Security testing): Đánh giá tính bảo mật của phần mềm để phát hiện và khắc phục các lỗ hổng bảo mật có thể được tấn công.

Một số trường hợp kiểm thử dựa trên các mức kiểm thử trên có thể là:

- Kiểm tra tính đúng đắn của kết quả khi cộng, trừ, nhân, chia hai số nguyên hoặc số thực.

- Kiểm tra xem phần mềm có xử lý được các biểu thức toán học phức tạp không.

- Kiểm tra việc xử lý ngoại lệ khi nhập vào giá trị không hợp lệ (ví dụ: chia cho 0).

- Kiểm tra khả năng xử lý số rất lớn hoặc rất nhỏ.

- Kiểm tra tính ổn định và hiệu suất khi sử dụng trong điều kiện tải cao.

**Bài 6 Sinh viên giải thích thêm về kiểm thử tỉnh táo, kiểm thử khói, kiểm thử hồi quy là gì và cho ví dụ**

1 Kiểm thử tỉnh táo (Ad Hoc Testing):

- Định nghĩa: Kiểm thử tỉnh táo là việc kiểm thử phần mềm mà không có kế hoạch hay kịch bản kiểm thử cụ thể. Thay vào đó, người kiểm thử sử dụng sự hiểu biết cá nhân, kinh nghiệm, và trực giác để tìm ra lỗi trong phần mềm.

- Ví dụ: Một sinh viên kiểm thử tỉnh táo một ứng dụng di động mới trên điện thoại thông minh. Anh ta mở ứng dụng, thực hiện các thao tác một cách ngẫu nhiên như người dùng thông thường và kiểm tra xem có lỗi hay không, chẳng hạn như việc ứng dụng bị treo khi thực hiện một thao tác cụ thể.

2 Kiểm thử khói (Smoke Testing):

- Định nghĩa: Kiểm thử khói là một hình thức kiểm thử sơ bộ để đảm bảo rằng hệ thống hoạt động cơ bản và có thể tiếp tục kiểm thử chi tiết. Nó thường được thực hiện sau mỗi lần triển khai hoặc sau mỗi chu kỳ phát triển lớn.

- Ví dụ: Khi một nhóm phát triển hoàn thành một tính năng mới, người kiểm thử có thể thực hiện kiểm thử khói bằng cách kiểm tra các chức năng chính của hệ thống để đảm bảo rằng không có lỗi nghiêm trọng nào xuất hiện sau quá trình triển khai.

3 Kiểm thử hồi quy (Regression Testing):

- Định nghĩa: Kiểm thử hồi quy là quá trình kiểm thử để đảm bảo rằng các thay đổi mới trong mã nguồn không gây ra lỗi hoặc ảnh hưởng đến các tính năng đã tồn tại trước đó. Nó giúp đảm bảo tính ổn định của hệ thống sau mỗi lần thay đổi.

- Ví dụ: Nếu một nhóm phát triển đã sửa một lỗi cụ thể trong mã nguồn, kiểm thử hồi quy sẽ được thực hiện để đảm bảo rằng việc sửa lỗi không làm ảnh hưởng đến các tính năng khác. Các kịch bản kiểm thử trước đây có thể được chạy lại để kiểm tra tính đồng nhất của hệ thống.