**Try Catch trong Java - Xử lý Exception**

**1. Khối lệnh try (try block) trong Java**

Try block chứa tập hợp các câu lệnh có thể xảy ra ngoại lệ. Try block luôn được theo sau bởi catch block để xử lý ngoại lệ của nó. Try block phải được theo sau bởi catch block hoặc finally block hoặc cả hai.

**Cú pháp của try block**

try{

   // Các câu lệnh có thể tạo ra exception

}

Trong quá trình code, nếu bạn nghĩ rằng có một số câu lệnh trong chương trình có thể tạo ra exception, hãy đặt chúng trong catch block và xử lý ngoại lệ đó.

**2. Khối lệnh catch(catch block)**

Catch block là nơi bạn xử lý các exception, nó phải đi kèm với try block. Một try block có thể có một hoặc nhiều catch block. Ta có thể bắt các exception khác nhau trong các catch block khác nhau. Khi một exception được tạo ra trong try block, đoạn code trong catch block tương ứng xử lý exception đó sẽ được thực thi. Ví dụ: nếu một exception số học xảy ra trong try block thì các câu lệnh được bao trong catch block dùng để xử lý exception đó sẽ được thực thi.

**Cú pháp try catch trong java**

try

{

     // Các câu lệnh có thể tạo ra exception

}

catch (exception(type) e(object))‏

{

     // Code xử lí exception e

}

**Ví dụ về try catch**

Nếu một exception xảy ra trong try block thì chương trình sẽ thực thi các câu lệnh trong catch block tương ứng. Một try block có thể có nhiều catch block được liên kết với nó, bạn nên sắp xếp các catch block hợp lí sao cho catch block xử lý exception chung ở cuối cùng (xem trong ví dụ bên dưới).

Một exception chung có thể xử lý tất cả các ngoại lệ nhưng bạn nên đặt ở cuối, nếu bạn đặt nó ở trước tất cả các catch block khác thì nó sẽ hiển thị một thông báo chung cho tất cả trường hợp. Chắc chắn bạn sẽ không muốn điều đó xảy ra đâu.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | class Example1 {     public static void main(String args[]) {        int num1, num2;        try {           /\* Ta nghi ngờ rằng khối lệnh này sẽ throw ra            \* exception vì vậy ta sẽ xử lý nó bằng cách đặt các câu lệnh này            \* bên trong try block và xử lý exception trong catch block            \*/           num1 = 0;           num2 = 62 / num1;           System.out.println(num2);           System.out.println("Kết thúc try block.");        }        catch (ArithmeticException e) {           /\* Catch block này sẽ chỉ thực hiện nếu có ngoại lệ số học            \* xảy ra trong try block            \*/           System.out.println(" Lỗi: Số bị chia không thể là số 0");        }        catch (Exception e) {           /\* Đây là một exception chung, nó có thể xử lý            \* tất cả các exception.Catch block này sẽ thực thi nếu ngoại lệ không            \* được xử lý bởi catch block bên trên            \*/           System.out.println("Lỗi: một ngoại lệ đã xảy ra");        }        System.out.println("Ra khỏi try catch block.");     }  } |

Output:

|  |  |
| --- | --- |
| 1  2 | Lỗi: Số bị chia không thể là số 0  Ra khỏi try catch block. |

**Nhiều catch block trong Java**

Sau đây là vài quy tắc cần lưu ý khi sử dụng try catch:

**Quy tắc 1**. Một try block có thể nhiều catch block.

**Quy tắc 2**. Một catch block xử lí exception chung có thể xử lý tất cả các exception, cho dù đó là ArrayIndexOutOfBoundException hay ArithaturesException hay NullPulumException hay bất kỳ loại exception nào.

|  |
| --- |
| catch(Exception e){  // catch block này có thể xử lí tất cả các exception  } |

Nếu bạn đang thắc mắc tại sao chúng ta cần phải xử lí các exception khác trong khi có thể code gọn hơn bằng cách catch exception chung cho tất cả trường hợp. Thì bạn phải nhớ rằng việc catch một exception chung sẽ chỉ hiển thị cùng một thông báo cho tất cả các ngoại lệ, và người dùng hoặc có thể là bạn sẽ không biết được exception nào đang xảy ra. Đó là lý do bạn nên đặt catch block xử lí exception chung ở cuối.

**Quy tắc 3**. Nếu không có exception xảy ra trong try block thì catch block hoàn toàn bị bỏ qua.

**Quy tắc 4**. Các catch block tương ứng thực thi cho exception cụ thể đó:  
catch (ArithaturesException e) dùng để xử lí ngoại lệ ArithaturesException  
catch (NullPulumException e) dùng để xử lí ngoại lệ NullPulumException

**Quy tắc 5**. Bạn cũng có thể throw exception, đây là phần nâng cao, tôi sẽ đề cập đến phần này trong một bài viết khác.

**Ví dụ về nhiều catch block**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | class Example2{     public static void main(String args[]){       try{           int a[]=new int[7];           a[4]=30/0;           System.out.println("Câu lệnh in đầu tiên trong try block");       }       catch(ArithmeticException e){          System.out.println("Cảnh báo: ngoại lệ ArithmeticException");       }       catch(ArrayIndexOutOfBoundsException e){          System.out.println("Cảnh báo: ngoại lệ ArrayIndexOutOfBoundsException");       }       catch(Exception e){          System.out.println("Cảnh báo: ngoại lệ khác");       }     System.out.println("Ra khỏi try-catch block...");    }  } |

Output:

|  |  |
| --- | --- |
| 1  2 | Cảnh báo: ngoại lệ ArithmeticException  Ra khỏi try-catch block... |

Trong ví dụ trên có nhiều catch block và các catch block này thực hiện tuần tự khi có exception xảy ra trong try block. Điều đó có nghĩa là nếu bạn đặt catch block cuối cùng catch(Exception e) ở vị trí đầu tiên, ngay sau try block thì trong trường hợp có bất kỳ exception nào, khối này sẽ thực thi vì nó có thể xử lý tất cả các exception. Catch block này nên được đặt ở cuối cùng để tránh những tình huống như vậy.

**3. Finally block**

Trong bài này chúng ta sẽ không đào sâu về finally block, bạn chỉ cần biết rằng khối này sẽ thực thi cho dù có exception xảy ra hay không. Hãy đặt các câu lệnh cần phải được thực thi mà không phụ thuộc vào việc exception có xảy ra hay không tại finally block.