Bài: Try catch trong Java

Xem bài học trên website để ủng hộ Kteam: Try catch trong Java

Mọi vấn đề về lỗi website làm ảnh hưởng đến bạn hoặc thắc mắc, mong muốn khóa học mới, nhằm hỗ trợ cải thiện Website. Các bạn vui lòng phản hồi đến Fanpage How Kteam nhé!

Dẫn nhập

Ở bài trước, chúng ta đã tìm hiểu đầy đủ về PHƯƠNG THỨC MAIN TRONG JAVA. Hôm nay, Kteam sẽ hướng dẫn cho các bạn các xử lý những lỗi có thể xảy ra trong đời thực bằng **try catch**.

Nội dung

Để đọc hiểu bài này, tốt nhất các bạn nên có kiến thức cơ bản về các phần sau:

- CÁC BIẾN TRONG JAVA.
- CÁC KIỂU DỮ LIỆU TRONG JAVA.
- CÁC HẠNG TOÁN TỬ TRONG JAVA
- CẤU TRÚC RẼ NHÁNH TRONG JAVA
- VÒNG LẶP WHILE TRONG JAVA
- VÒNG LẶP FOR TRONG JAVA
- MẢNG TRONG JAVA
- VÒNG LẶP FOR-EACH TRONG JAVA
- VAI TRÒ BREAK, CONTINUE TRONG VÒNG LẶP JAVA
- SWITCH TRONG JAVA
- LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG
- CLASS TRONG LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG
- CÁC LOẠI PHẠM VI TRUY CẬP TRONG LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG
- TỪ KHÓA STATIC TRONG LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG
- TỪ KHÓA THIS TRONG LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG
- THỪA KẾ TRONG LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG
- SETTER & GETTER TRONG LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG
- OVERRIDING VÀ OVERLOADING TRONG LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG
- TÍNH TRỪU TƯỢNG TRONG LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG
- INTERFACE TRONG LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG
- PHƯƠNG THỨC MAIN TRONG JAVA

Bài này chúng ta sẽ tìm hiểu những vấn đề sau:

- Try Catch là gì?
- Cú pháp Try Catch
- Từ khóa finally
- Từ khóa throw

Try Catch là gì?

Khi chạy chương trình, có rất nhiều loại lỗi khác nhau có thể xảy ra như: lỗi do sai lầm người viết, lỗi do sai thông tin đầu vào hoặc những lỗi mà không thể lường trước được. Và khi có lỗi, Java sẽ dừng lại và hiện thị thông tin lỗi ra, kĩ thuật đó thường được gọi là '**throw an exception/error**'.

Chương trình Java sẽ ném lỗi như sau:

```
Exception in thread "main" java.lang.NumberFormatException: For input string: "a"
        at java.lang.NumberFormatException.forInputString(NumberFormatException.java:65)
        at java.lang.Integer.parseInt(Integer.java:580)
        at java.lang.Integer.parseInt(Integer.java:615)
        at HelloWorld.main(HelloWorld.java:8)
```

Và có những lỗi xuất phát từ người dùng, thì lúc đó ta không thể cho họ xem thông tin lỗi như thế này được. Với những người không thành thạo về máy vi tính hoặc tiếng Anh thì họ nghĩ chương trình bạn viết bị lỗi mà không phải lỗi từ họ.

Vì vậy, **Try Catch** có nhiệm vụ **bắt** (**Catch**) các lỗi mà thực tế có thể xảy ra để xử lý sao cho chương trình thân thiện với người dùng hơn.

Cú pháp Try Catch

Cú pháp:

```
try {
   //Những khối lệnh có thể phát sinh lỗi
} catch (Exception e) { //tham số e là tên lỗi muốn xử lý
   //Chương trình thực hiện khi gặp lỗi trên
}
```

Ví dụ: khi gặp lỗi lấy phần tử trong mảng mà không tồn tại

```
public class HelloWorld {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int[] a = {5,6,7};
        System.out.println(a[4]);
    }

}
```

```
<terminated> HelloWorld (1) [Java Application] C:\Program Files\Java\jdk1.8.0_191\bin\javaw.e
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 4
        at HelloWorld.main(HelloWorld.java:7)
```

Ta thấy chương trình sẽ throw lỗi là **ArrayIndexOutOfBoundsException**, ta sẽ **catch** lỗi đó như sau:

java:

```
public static void main(String[] args) {
        // TODO Auto-generated method stub
        try {
                int[] a = {5,6,7};
                System.out.println(a[4]);
        } catch (ArrayIndexOutOfBoundsException e) {
                System.out.println("Index does not exist");
        }

}
```

```
Index does not exist
```

Nếu bạn không thể lường trước toàn bộ lỗi, ta có thể đặt mặc định **Exception** để xử lý lỗi mà bạn chưa tính đến được.

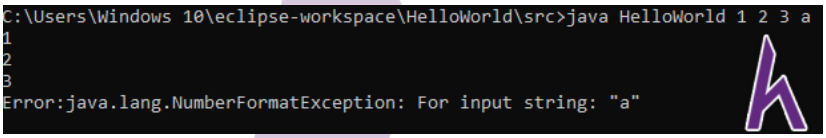
Ví dụ: Tính tổng các tham số truyền vào trong phương thức main

java:

```
public static void main(String[] args) {
	// TODO Auto-generated method stub
	int S=0;
	try {
		for (String arg : args) {
			S+= Integer.parseInt(arg);
			System.out.println(arg);
		}
	} catch (Exception e) {
		System.out.println("Error:"+e.toString());
	}

}
```

Ta dùng phương thức parseInt trong **class Integer** để chuyển giá trị kiểu **String** sang giá trị kiểu **int** (với điều kiện chuỗi đó mang ý nghĩa con số). Giả sử người dùng cho tham số truyền vào không phải kí tự số



Như vậy lỗi có tên là: **java.lang.NumberFormatException**

Từ khóa finally

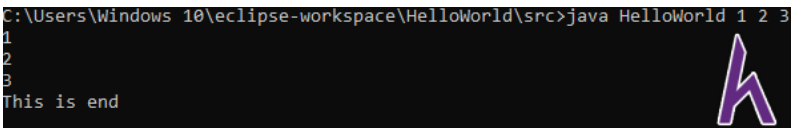
Từ khóa finally có ý nghĩa sẽ chạy những dòng code sau khi kết thúc try catch bất kì có lỗi hay không.

Ví dụ:

java:

```
public static void main(String[] args) {
	// TODO Auto-generated method stub
	int S=0;
	try {
		for (String arg : args) {
			S+= Integer.parseInt(arg);
			System.out.println(arg);
		}
	} catch (Exception e) {
		System.out.println("Error:"+e.toString());
	} finally {
		System.out.println("This is end");
	}

}
```



Từ khóa throw

Từ khóa throw mục đích chính là để ném lỗi, thường khi bạn viết các phương thức cho người khác người dùng và bắt người khác phải tự xử lý những trường hợp đó. Bản chất các **Exception** là các **class**, nên khi ném lỗi có nghĩa là: Bạn khởi tạo đối tượng **Exception** và **throw** cho người viết sau này phải **catch** lại:

Ví dụ: Throw lỗi nếu người dùng nhập tham số lớn hơn 100, ta sử dụng **class ArithmeticException** để xử lý:

java:

```
public class HelloWorld {

	static void inputValue(int value) {
		if (value>100) {
			throw new ArithmeticException("Value>100");
		}
	}

	public static void main(String[] args) {
		// TODO Auto-generated method stub
			inputValue(102);

	}

}
```

```
Exception in thread "main" java.lang.ArithmeticException: Value>100
	at HelloWorld.inputValue(HelloWorld.java:6)
	at HelloWorld.main(HelloWorld.java:12)
```

Kết

Như vậy chúng ta đã tìm hiểu try catch trong JAVA

Ở bài sau, Kteam sẽ giới thiệu đến bạn về 4 TÍNH CHẤT CỦA HƯỚNG ĐỐI TƯỢNG

Cảm ơn các bạn đã theo dõi bài viết. Hãy để lại bình luận hoặc góp ý của mình để phát triển bài viết tốt hơn. Đừng quên "**Luyện tập – Thử thách – Không ngại khó**".