Cập nhật tháng 8 năm 2024

[Bài đọc] Xử lý ngoại lệ trong Java

1. Khối Try
- Khái niệm:
    - Khối try chứa các đoạn mã mà bạn muốn theo dõi lỗi hoặc ngoại lệ
    - Nếu bất kỳ ngoại lệ nào xảy ra trong khối try, nó sẽ được chuyển đến khối catch tương ứng
- Cú pháp:

```
try {
    // Đoạn mã có khả năng gây ra ngoại lệ
}
```

- Lưu ý:
    - Bạn phải sử dụng khối try cùng với khối catch hoặc finally. Một mình try không hợp lệ
    - Chỉ những lỗi xảy ra trong khối try mới được chuyển đến catch

2. Khối Catch
- Khái niệm:
    - Khối catch được sử dụng để xử lý ngoại lệ được ném ra từ khối try
    - Bạn có thể có nhiều khối catch để xử lý các loại ngoại lệ khác nhau
- Cú pháp:

```
catch (ExceptionType e) {
    // Xử lý ngoại lệ
}
```

- Đặc điểm:
    - ExceptionType là loại ngoại lệ bạn muốn xử lý (ví dụ: IOException, ArithmeticException, v.v.)

    - Biến e đại diện cho đối tượng ngoại lệ
- Ví dụ:

```
try {
    int result = 10 / 0; // Gây ra ArithmeticException
} catch (ArithmeticException e) {
    System.out.println("Lỗi: " + e.getMessage());
}
```

3. Khối Finally

- Khái niệm:
    - Khối finally chứa mã mà bạn muốn thực thi **dù ngoại lệ có xảy ra hay không**
    - Thường được sử dụng để giải phóng tài nguyên như đóng file, kết nối cơ sở dữ liệu, v.v
- Cú pháp:

```
finally {
    // Đoạn mã luôn được thực thi
}
```

- Đặc điểm:
    - Khối finally không bắt buộc nhưng rất hữu ích trong các tình huống cần dọn dẹp tài nguyên
    - Khối finally luôn được thực thi, trừ khi chương trình kết thúc đột ngột do lệnh System.exit()
- Ví dụ:

```
try {
    int[] numbers = {1, 2, 3};
    System.out.println(numbers[5]); // Gây ra ArrayIndexOutOfBoundsException
} catch (ArrayIndexOutOfBoundsException e) {
    System.out.println("Lỗi: " + e.getMessage());
} finally {
    System.out.println("Khối finally luôn được thực thi.");
}
```

4. Sự kết hợp của try, catch, finally

- Cú pháp đầy đủ:

- Ví dụ:

```java
public class TryCatchFinallyExample  new *
{
    public static void main(String[] args)  new *
    {
        try
        {
            int result = 10 / 0; // Gây ra ngoại lệ
            System.out.println("Kết quả: " + result);
        }
        catch (ArithmeticException e)
        {
            System.out.println("Ngoại lệ xảy ra: " + e.getMessage());
        }
        finally
        {
            System.out.println("Khối finally luôn được thực thi.");
        }
    }
}
```

- Kết quả:

```
Ngoại lệ xảy ra: / by zero
Khối finally luôn được thực thi.
```

5. **Một số lưu ý quan trọng**

- Số lượng khối catch:
    - Bạn có thể có nhiều khối catch sau một khối try để xử lý các loại ngoại lệ khác nhau
    - Thứ tự các khối catch rất quan trọng. Các ngoại lệ cụ thể hơn phải được bắt trước các ngoại lệ tổng quát hơn (ví dụ: ArithmeticException trước Exception)
- Khối finally luôn chạy:
    - Khối finally được thực thi ngay cả khi:
        - Ngoại lệ xảy ra hoặc không xảy ra
        - Có lệnh return trong khối try hoặc catch
- Khối try không thể đứng một mình:
    - Phải có ít nhất một khối catch hoặc finally

6. **Từ khóa throw**

- **Khái niệm:**
    - Từ khóa throw được sử dụng để **ném một ngoại lệ** cụ thể trong chương trình
    - throw thường được dùng trong thân của phương thức hoặc khối try-catch để tạo ra ngoại lệ tùy chỉnh hoặc ném ngoại lệ đã có sẵn
- **Cú pháp:**

```
throw new ExceptionType("Message");
```

- **Đặc điểm:**
    - Chỉ có thể ném một đối tượng ngoại lệ tại một thời điểm
    - Đối tượng ngoại lệ phải là một instance của lớp Throwable hoặc các lớp con của nó
    - Khi sử dụng throw, chương trình sẽ dừng lại tại vị trí ngoại lệ và chuyển quyền kiểm soát đến trình xử lý ngoại lệ
- **Ví dụ:**

```java
public class ThrowExample  new *
{
    public static void main(String[] args)  new *
    {
        try
        {
            validateAge(15); // Gây ra ngoại lệ
        }
        catch (IllegalArgumentException e)
        {
            System.out.println("Lỗi: " + e.getMessage());
        }
    }

    public static void validateAge(int age)  1 usage  new *
    {
        if (age < 18)
        {
            throw new IllegalArgumentException("Tuổi phải lớn hơn hoặc bằng 18.");
        }
        System.out.println("Tuổi hợp lệ.");
    }
}
```

- **Kết quả:**

```
Lỗi: Tuổi phải lớn hơn hoặc bằng 18.
```

7. Từ khóa throws

- **Khái niệm:**
  - Từ khóa throws được sử dụng để **khai báo ngoại lệ** mà một phương thức có thể ném ra
  - Nó chỉ định rằng phương thức có khả năng ném một hoặc nhiều ngoại lệ và không xử lý trực tiếp trong phương thức
- **Cú pháp:**

```
returnType methodName() throws ExceptionType1, ExceptionType2 {
    // code
}
```

- **Đặc điểm:**
  - Dùng để khai báo nhiều loại ngoại lệ, được phân tách bằng dấu phẩy
  - Giúp người dùng biết rằng cần xử lý các ngoại lệ khi gọi phương thức đó
  - Thường được sử dụng cho Checked Exception (ngoại lệ kiểm tra trong compile-time)
- **Ví dụ:**

```java
public class ThrowsExample  new *
{
    public static void main(String[] args)  new *
    {
        try
        {
            readFile( fileName: "nonexistent.txt"); // Gây ra FileNotFoundException
        }
        catch (Exception e)
        {
            System.out.println("Lỗi: " + e.getMessage());
        }
    }

    public static void readFile(String fileName) throws Exception  1 usage  new *
    {
        File file = new File(fileName);
        FileReader fr = new FileReader(file); // Có thể ném FileNotFoundException
        System.out.println("Tệp đã được đọc.");
    }
}
```

- Kết quả:

```
Lỗi: nonexistent.txt (The system cannot find the file specified)
```

8. **So sánh throw và throws**

Đặc điểm	**Throw**	**Throws**
Mục đích	Ném một ngoại lệ cụ thể trong chương trình	Khai báo ngoại lệ mà một phương thức có thể ném
Vị trí sử dụng	Trong thân phương thức hoặc khối try-catch	Trong khai báo phương thức
Ngoại lệ ném ra	Chỉ một ngoại lệ tại một thời điểm	Có thể khai báo nhiều loại ngoại lệ
Loại ngoại lệ	Thường dùng cả Checked và Unchecked	Chủ yếu dùng cho Checked Exception
Thời điểm kiểm tra	Kiểm tra tại runtime	Kiểm tra tại compile-time

Tài nguyên đọc thêm:

- Try-catch, finally: https://www.geeksforgeeks.org/flow-control-in-try-catch-finally-in-java
- Throw và Throws: https://www.geeksforgeeks.org/difference-between-throw-and-throws-in-java