# Handle Exception

# 1. Ngoại lệ (Exception) là gì trong Java?

Ngoại lệ là một sự kiện bất thường xảy ra trong quá trình thực thi của chương trình, gây gián đoạn luồng thực thi bình thường. Java sử dụng các cơ chế ngoại lệ để xử lý các lỗi này.

# 2. Các loại ngoại lệ trong Java là gì?

* Checked Exception: Các ngoại lệ được kiểm tra tại thời điểm biên dịch (compile-time). Ví dụ: IOException, SQLException.
* Unchecked Exception: Các ngoại lệ xảy ra tại thời điểm runtime và không bắt buộc phải xử lý (Runtime Exception). Ví dụ: NullPointerException, ArrayIndexOutOfBoundsException.
* Error: Đây là những vấn đề nghiêm trọng, không thể hoặc không nên cố gắng xử lý. Ví dụ: OutOfMemoryError.

# 3. Sự khác biệt giữa "throw" và "throws" là gì?

* throw dùng để ném ra một ngoại lệ cụ thể trong một phương thức.
* throws là một khai báo trong phần định nghĩa phương thức để chỉ ra rằng phương thức có thể ném ngoại lệ.

# 4. Tại sao cần sử dụng khối finally?

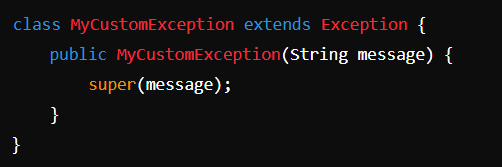
* Khối finally được thực thi cho dù có xảy ra ngoại lệ hay không. Nó thường được sử dụng để thực hiện dọn dẹp tài nguyên (như đóng file, kết nối database, v.v.).

# 5. Checked Exception và Unchecked Exception khác nhau như thế nào?

* Checked Exception yêu cầu phải được xử lý hoặc khai báo trong chữ ký phương thức (throws), còn Unchecked Exception không bắt buộc phải xử lý.

# 6. Làm cách nào để tạo ngoại lệ tùy chỉnh (custom exception)?

* Để tạo ngoại lệ tùy chỉnh, bạn có thể kế thừa từ lớp Exception hoặc RuntimeException, ví dụ:

****

# 7. Thế nào là checked và unchecked exception?

### **Checked Exception**

1. **Định nghĩa**:
   * Checked Exception là các ngoại lệ mà trình biên dịch yêu cầu phải được xử lý (bằng cách sử dụng khối try-catch) hoặc phải được khai báo trong chữ ký của phương thức bằng từ khóa throws.
2. **Thời điểm xảy ra**:
   * Checked Exception xảy ra trong quá trình biên dịch (compile-time), tức là nếu không xử lý chúng, mã sẽ không biên dịch thành công.
3. **Ví dụ**:
   * Một số ví dụ điển hình của Checked Exception bao gồm:
     + IOException: Khi có lỗi xảy ra khi đọc hoặc ghi file.
     + SQLException: Khi có lỗi xảy ra khi thực hiện truy vấn cơ sở dữ liệu.
     + FileNotFoundException: Khi một file không được tìm thấy.

### **Unchecked Exception**

1. **Định nghĩa**:
   * Unchecked Exception là các ngoại lệ không yêu cầu trình biên dịch phải xử lý. Người lập trình không bắt buộc phải bắt hoặc khai báo chúng.
2. **Thời điểm xảy ra**:
   * Unchecked Exception xảy ra trong quá trình thực thi (runtime), tức là chúng có thể xảy ra ở bất kỳ đâu trong mã và thường là do lỗi lập trình.
3. **Ví dụ**:
   * Một số ví dụ điển hình của Unchecked Exception bao gồm:
     + NullPointerException: Khi cố gắng truy cập một đối tượng mà chưa được khởi tạo (null).
     + ArrayIndexOutOfBoundsException: Khi cố gắng truy cập vào một chỉ số ngoài giới hạn của mảng.
     + ArithmeticException: Khi thực hiện phép chia cho 0.
     + **Checked Exception**: Yêu cầu phải xử lý, xảy ra tại compile-time, ví dụ: IOException, SQLException.
     + **Unchecked Exception**: Không yêu cầu phải xử lý, xảy ra tại runtime, ví dụ: NullPointerException, ArrayIndexOutOfBoundsException.

# 8. Thế nào là checked và unchecked exception?

try-catch và try-with-resources là hai cơ chế trong Java dùng để xử lý ngoại lệ, nhưng chúng phục vụ các mục đích khác nhau và có cách sử dụng khác nhau. Dưới đây là sự khác biệt giữa hai cơ chế này:

### **1. Try-Catch**

* **Định nghĩa**:
  + try-catch là cơ chế cơ bản để xử lý ngoại lệ trong Java. Bạn đặt mã có khả năng gây ra ngoại lệ trong khối try và xử lý ngoại lệ đó trong khối catch.
* **Cách sử dụng**:
  + Nếu có ngoại lệ xảy ra trong khối try, chương trình sẽ chuyển sang khối catch để xử lý. Nếu không có ngoại lệ nào xảy ra, khối catch sẽ bị bỏ qua.
* **Tài nguyên**:
  + Trong try-catch, bạn phải tự quản lý việc đóng tài nguyên (như file, socket, v.v.) trong khối finally, nếu cần.

### **2. Try-With-Resources**

* **Định nghĩa**:
  + try-with-resources là một cơ chế được giới thiệu từ Java 7, cho phép bạn tự động đóng các tài nguyên như file, stream, hoặc kết nối database sau khi sử dụng.
* **Cách sử dụng**:
  + Bạn có thể khai báo các tài nguyên trong phần đầu của khối try. Khi khối try kết thúc, Java sẽ tự động gọi phương thức close() trên các tài nguyên này, ngay cả khi có ngoại lệ xảy ra.

**Tài nguyên**:

* Tài nguyên được tự động đóng mà không cần phải viết mã riêng trong khối finally, giúp giảm thiểu lỗi và làm mã trở nên ngắn gọn hơn.

|  |  |  |
| --- | --- | --- |
| Tiêu chí | Try-Catch | Try-With-Resources |
| Mục đích | Xử lý ngoại lệ | Xử lý ngoại lệ và quản lý tài nguyên tự động |
| Quản lý tài nguyên | Cần phải tự đóng tài nguyên trong khối finally | Tài nguyên tự động đóng khi kết thúc khối try |
| Sử dụng | Dùng cho mã có thể gây ra ngoại lệ | Dùng cho các tài nguyên cần được đóng (phải implements AutoCloseable) |