Handle Exception

**1. Phân biệt throw và throws**

**a. throw:**

* **Định nghĩa**: throw được sử dụng để **ném ra một ngoại lệ** (exception) cụ thể từ khối lệnh hoặc phương thức. Đây là cách mà lập trình viên có thể tự tạo và ném ngoại lệ.
* **Sử dụng**: Thường dùng khi bạn muốn **tạo ra một ngoại lệ thủ công** trong code.
* **Vị trí**: Được dùng trong thân hàm hoặc khối lệnh.

Vd:

public void checkAge(int age) {

if (age < 18) {

throw new IllegalArgumentException("Tuổi phải lớn hơn 18");

}

}

**b. throws:**

* **Định nghĩa**: throws được sử dụng để **khai báo rằng một phương thức có thể ném ra một hoặc nhiều ngoại lệ**. Đây là cách bạn khai báo với trình biên dịch rằng phương thức này có thể phát sinh ngoại lệ mà không xử lý nó tại chỗ.
* **Sử dụng**: Khai báo tại **chữ ký của phương thức** để thông báo các ngoại lệ có thể xảy ra.
* **Vị trí**: Được sử dụng trong khai báo phương thức.

**Vd**:

public void readFile(String fileName) throws FileNotFoundException {

File file = new File(fileName);

Scanner sc = new Scanner(file);

}

**2. Phân biệt Checked Exception và Unchecked Exception**

**a. Checked Exception:**

* **Định nghĩa**: Là các ngoại lệ được kiểm tra tại **thời điểm biên dịch**. Trình biên dịch yêu cầu bạn phải xử lý (bằng try-catch hoặc khai báo throws) các ngoại lệ này.
* **Ví dụ**: IOException, SQLException, FileNotFoundException.
* **Đặc điểm**:
  + Bắt buộc phải xử lý hoặc khai báo trong throws nếu phương thức có thể phát sinh ngoại lệ.

**b. Unchecked Exception:**

* **Định nghĩa**: Là các ngoại lệ được kiểm tra tại **thời gian chạy** (runtime). Trình biên dịch **không bắt buộc** bạn phải xử lý ngoại lệ này.
* **Ví dụ**: NullPointerException, ArrayIndexOutOfBoundsException, ArithmeticException.
* **Đặc điểm**:
  + Không bắt buộc phải xử lý ngoại lệ này trong try-catch hoặc khai báo throws.

**3. Sự khác nhau giữa try-catch và try-with-resources**

**a. try-catch:**

* **Định nghĩa**: Dùng để bắt và xử lý ngoại lệ. Khi một ngoại lệ xảy ra trong khối try, nó sẽ nhảy tới khối catch tương ứng để xử lý.
* **Đặc điểm**:
  + Bạn phải tự đóng tài nguyên (resource) như file hoặc kết nối cơ sở dữ liệu nếu cần sau khi sử dụng.
  + Được sử dụng khi bạn không có tài nguyên cần tự động giải phóng.

**b. try-with-resources:**

* **Định nghĩa**: Là một phiên bản mở rộng của try-catch, dùng để **tự động đóng các tài nguyên** (như file, kết nối cơ sở dữ liệu) khi hoàn thành.
* **Đặc điểm**:
  + Các tài nguyên được khai báo trong try sẽ **tự động được đóng** sau khi thực thi khối try kết thúc, ngay cả khi có ngoại lệ.
  + Dùng khi bạn cần xử lý tài nguyên (như luồng, file) một cách an toàn và tự động giải phóng chúng.

**4. Làm thế nào để tạo một Custom Exception?**

Bạn có thể tạo một ngoại lệ tùy chỉnh (custom exception) bằng cách kế thừa lớp Exception hoặc RuntimeException. Điều này phụ thuộc vào việc bạn muốn tạo **checked exception** (kế thừa Exception) hay **unchecked exception** (kế thừa RuntimeException).

a. Tạo một **checked exception**:

// Kế thừa từ Exception để tạo checked exception

public class CustomCheckedException extends Exception {

public CustomCheckedException(String message) {

super(message);

}

}

b. Tạo một **unchecked exception**:

// Kế thừa từ RuntimeException để tạo unchecked exception

public class CustomUncheckedException extends RuntimeException {

public CustomUncheckedException(String message) {

super(message);

}

}

c. Sử dụng custom exception: