# Phân biệt throw và throws

1. **throw**

* Dùng để ném một ngoại lệ cụ thể trong mã nguồn.
* Cú pháp: throw new ExceptionType("message");
* Có thể ném ngoại lệ trong bất kỳ phần nào của mã, không cần phải khai báo.

1. **throws**

* Dùng trong khai báo phương thức để chỉ ra rằng phương thức có thể ném một hoặc nhiều loại ngoại lệ.
* Cú pháp: public void methodName() throws ExceptionType1, ExceptionType2 { ... }
* Bắt buộc phải khai báo nếu phương thức có thể ném ngoại lệ checked.

# Checked và Unchecked Exception

1. **Checked Exception**

* Là những ngoại lệ mà trình biên dịch yêu cầu phải xử lý (thông qua try-catch hoặc khai báo throws).
* Ví dụ: IOException, SQLException.

1. **Unchecked Exception**

* Là những ngoại lệ không bị trình biên dịch yêu cầu phải xử lý. Chúng thường do lỗi lập trình như lỗi logic, truy cập mảng ngoài giới hạn, v.v.
* Ví dụ: NullPointerException, ArrayIndexOutOfBoundsException.

# try-catch và try-with-resources

1. **try-catch**

* Dùng để xử lý các ngoại lệ trong khối mã.
* Cú pháp:

try {

// Code có thể ném ngoại lệ

} catch (ExceptionType e) {

// Xử lý ngoại lệ

}

1. **try-with-resources**

* Được sử dụng khi làm việc với các đối tượng cần đóng (như InputStream, OutputStream, v.v.). Java tự động đóng các đối tượng này khi ra khỏi khối try.
* Cú pháp:

try (ResourceType resource = new ResourceType()) {

// Code sử dụng resource

} catch (ExceptionType e) {

// Xử lý ngoại lệ

}

# Tạo Custom Exception

1. **Tạo một class mới:**

Kế thừa từ Exception hoặc RuntimeException.

1. **Xác định các constructor:**

Có thể thêm các constructor để truyền thông điệp hoặc nguyên nhân.

1. **Ví dụ:**

public class MyCustomException extends Exception {

public MyCustomException(String message) {

super(message);

}

}

public void myMethod() throws MyCustomException {

// Logic của phương thức

if (someCondition) {

throw new MyCustomException("Something went wrong!");

}

}