Luồng vào ra I/O trong Java

Java cung cấp nhiều lớp và phương thức để thực hiện thao tác với dữ liệu từ các nguồn vào ra, như file, mạng, bàn phím,... Các lớp này được tổ chức trong gói java.io và java.nio.

1. Xử lý ngoại lệ

Ngoại lệ: sự kiện xảy ra trong quá trình thực thi chương trình, và có thể làm gián đoạn luồng xử lý thông thường của chương tình

=> Cần có cơ chế xử lý các ngoại lệ.

1. Xử lý ngoại lệ (Try-catch-finally)

- try: Khối chứa đoạn mã có thể gặp lỗi hoặc ném ra ngoại lệ khi thực thi. Nếu có ngoại lệ xảy ra, chương trình sẽ chuyển quyền kiểm soát đến khối catch

- catch: Khối chứa đoạn mã được thực thi nếu có ngoại lệ xảy ra trong khối try tương ứng

- finally: Khối này chứa đoạn mã sẽ được thực thi bất kể có ngoại lệ xảy ra hay không. Khối này có thể có hoặc không trong quá trình xử lý ngoại lệ

2. Các loại ngoại lệ

1. **Exception**

- RuntimeException (unchecked exception): được phát hiện trong runtime, là ngoại lệ có thể tránh được

* ArithmeticException
* NullPointerException
* NumberFormatException
* ClassCastException

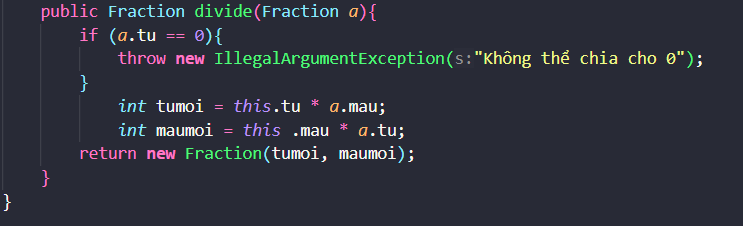
- Ngoại lệ khác (checked exception): được phát hiện trong quá trình biên dịch

* IOException
* ParseException
* ClassNotFoundException
* SQLException

3. Ngoại lệ tùy chỉnh

- Khai báo method với throws và ném ra ngoại lệ với từ khóa throw

Ví dụ :



1. **Làm việc với tệp tin**
2. Các lớp cơ bản :

Byte Stream

* InputStream: Lớp gốc cho tất cả các luồng đầu vào (input streams) xử lý dữ liệu dạng byte.
  + FileInputStream: Dùng để đọc dữ liệu từ file.
* **OutputStream**: Lớp gốc cho tất cả các luồng đầu ra (output streams) xử lý dữ liệu dạng byte.
  + **FileOutputStream**: Dùng để ghi dữ liệu vào file.

Character Stream

* **Reader**: Lớp gốc cho tất cả các luồng đầu vào xử lý dữ liệu dạng ký tự.
  + **FileReader**: Dùng để đọc dữ liệu từ file dạng ký tự.
* **Writer**: Lớp gốc cho tất cả các luồng đầu ra xử lý dữ liệu dạng ký tự.
  + **FileWriter**: Dùng để ghi dữ liệu dạng ký tự vào file.

1. **Buffered I/O**

* Buffered I/O giúp cải thiện hiệu suất bằng cách sử dụng bộ nhớ đệm (buffer). Nó cho phép xử lý dữ liệu theo từng khối lớn thay vì từng byte/char một.
* **BufferedReader**: Đọc dữ liệu từ một stream và lưu trữ vào bộ đệm để đọc hiệu quả hơn.
* **BufferedWriter**: Ghi dữ liệu ra file với hiệu suất cao hơn nhờ sử dụng bộ đệm.

1. File Classs

* Lớp File được sử dụng để đại diện cho đường dẫn đến một file hoặc thư mục trên hệ thống.