* Exception: là một sự kiện xảy ra trong khi thực thi chương trình, làm gián đoạn flow thông thường của chương trình.
* Các cách để bắt và xử lý ngoại lệ: bao gồm try/catch/finally blocks
* Có thể nest try- catch block ở trong cả try block, catch block và finally block để thực hiện một số yêu cầu xử lý ngoại lệ đặc biệt.
* **Throws**: được sử dụng để tuyên bố rằng

1. Exception là gì?

* Exception: là một sự kiện xảy ra trong khi thực thi chương trình, làm gián đoạn flow thông thường của chương trình.
* Exception hierarchy

A picture containing text, screenshot, font, line

Description automatically generated

* Có 3 kiểu ngoại lệ chính:
* Check Exceptions
* Unchecked Exceptions/Runtime exceptions
* Errors
  1. Checked Exceptions: Là một Exceptions mà Java compiler yêu cầu chúng ta phải handle. - Chúng ta phải throws Exception lên ngăn xếp, hoặc chúng ta có thể xử lý nó.

Tài liệu của Oracle cho chúng ta biết sử dụng các ngoại lệ được kiểm tra khi chúng ta có thể mong đợi một cách hợp lý rằng người gọi phương thức của chúng ta có thể khôi phục.

* 1. Unchecked Exceptions: Là các ngoại lệ mà trình biên dịch không bắt ta xử lý.
* Và mặc dù điều này nghe có vẻ thuận tiện, nhưng tài liệu của Oracle cho chúng ta biết rằng có những lý do chính đáng cho cả hai khái niệm, chẳng hạn như phân biệt giữa lỗi tình huống (được chọn) và lỗi sử dụng (không được chọn).
* Một số ví dụ về ngoại lệ không được kiểm tra là NullPulumException, IllegalArgumentException và SecurityException.
  1. Errors.
* Lỗi biểu thị các tình trạng nghiêm trọng và thường không thể khôi phục được như thư viện không tương thích, đệ quy vô hạn hoặc rò rỉ bộ nhớ.
* Và mặc dù chúng không mở rộng RuntimeException, nhưng chúng cũng không được chọn.
* Trong hầu hết các trường hợp, sẽ rất lạ khi chúng tôi xử lý, khởi tạo hoặc mở rộng Lỗi. Thông thường, chúng tôi muốn những thứ này lan truyền hết cỡ.
* Một vài ví dụ về lỗi là StackOverflowError và OutOfMemoryError.

1. Throws: để tuyên bố rằng phương thức sẽ ném ra một ngoại lệ mà không xử lý nó. Nó sẽ chuyển trách nhiệm xử lý ngoại lệ đó cho hàm gọi. Hàm gọi có thể xử lý ngoại lệ bằng “try-catch” block, hoặc cũng có thể ném ra ngoại lệ đó mà không xử lý bằng từ khóa “throws”.
2. Try/catch

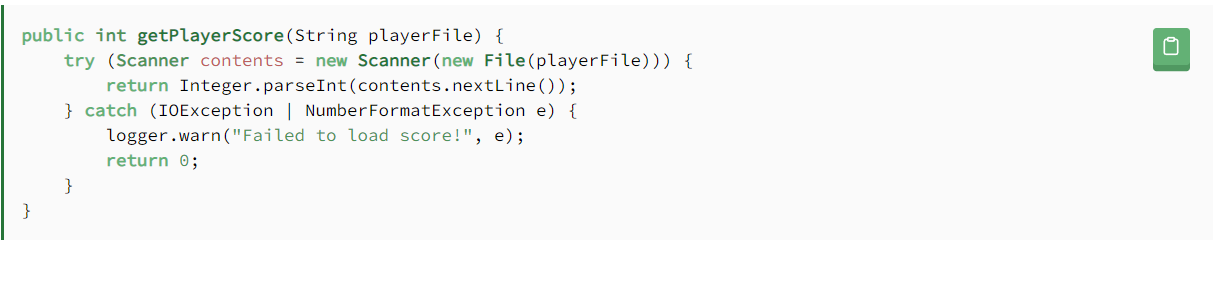
* Dùng để xử lý ngoại lệ theo ý muốn.
* Có thể rethowing ngoại lệ.

1. Finally

* Chứa đoạn mã cần thực thi bất kể ngoại lệ có xảy ra hay không.

1. Try-with-resources
2. Multiple catch Blocks
3. Union catch blocks

* Xử lý nhiều error theo cùng một kiểu như nhau



1. Throwing Exceptions

* Trong trường hợp không muốn xử lý ngoại lệ, hoặc muốn generate ngoại lệ để người khác xử lý: ***throw*** keyword.
  1. Throwing CheckedException.
  2. Throwing UncheckedException
  3. Wrapping and Rethrowing
* Chúng ta có thể ném ra lại ngoại lệ mà chúng ta vừa bắt
* Hoặc đóng gói lại Exception và ném ra lại.
  1. Rethrowing Throwable hoặc Exception
* Nếu như một đoạn code có khả năng quăng ra một unchecked exception, chúng ta có thể catch hoặc rethow Throwable hoặc Exception mà không cần add chúng vào trong method
  1. Kế thừa
* a method overrided in a subclass can have "less risky" signature but not a "more riskier" signature
* Điều đó bởi vì các contracts được xác định ngay từ thời điểm biên dịch theo loại tham chiếu.

1. Anti-Patterns
   1. Swallowing Exceptions

* Để catch block rỗng, hoặc chỉ in ra lỗi. Trong trường hợp chúng ta tự tin rằng lỗi đó không thể xảy ra. Tuy nhiên, nếu có lỗi thì nó không giải quyết đc vấn đề và khiến các code khác cũng bị cuốn theo lỗi đó.

A screenshot of a computer

Description automatically generated with medium confidence

1. Advantages of Exceptions

* Tách code xử lý lỗi khỏi các code triển khai thông thường.
* Lan truyền lỗi lên Call Stack.
* Nhóm và phân biệt các loại lỗi.