**XỬ LÝ NGOẠI LỆ**

**1. Ngoại lệ là gì**

* Bản chất ngoại lệ là gì
  + Lỗi logic xảy ra trong thời gian chạy
  + 2 nhóm nguyên nhân cơ bản
    - Lỗi lập trình: chưa xử lý tốt yêu cầu bài toán
    - Các tình huống phát sinh không mong muốn do tác động từ bên ngoài.
  + Gốc của vấn đề: mỗi một đoạn mã nguồn có một “phạm vi – biên” có khả năng xử lý được. Ngoại lệ xảy ra khi đoạn mã nguồn đó được yêu cầu các xử lý nằm ngoài phạm vi – biên đó.
* Lớp quản lý ngoại lệ trong Java: Exception
  + Mô tả các đặc trưng và phương thức cần thiết nhất để quản lý Exception nói chung.
  + Stack Trace
* Có rất nhiều các lớp mô tả các kiểu ngoại lệ, là các lớp kế thừa từ lớp Exception
* Ngoại lệ sinh ra thế nào
  + Ngoại lệ được phát sinh qua từ khóa **throw new** – gọi đến hàm khởi tạo của lớp Exception tương ứng.
  + Sau đó ngoại lệ được đẩy qua các phương thức thông qua từ khóa **throws**
  + Cuối cùng, ngoại lệ xuất hiện khi được kích hoạt bởi dòng lệnh trong hàm main.

**2. Nguyên tắc chung để xử lý ngoại lệ trong Java**

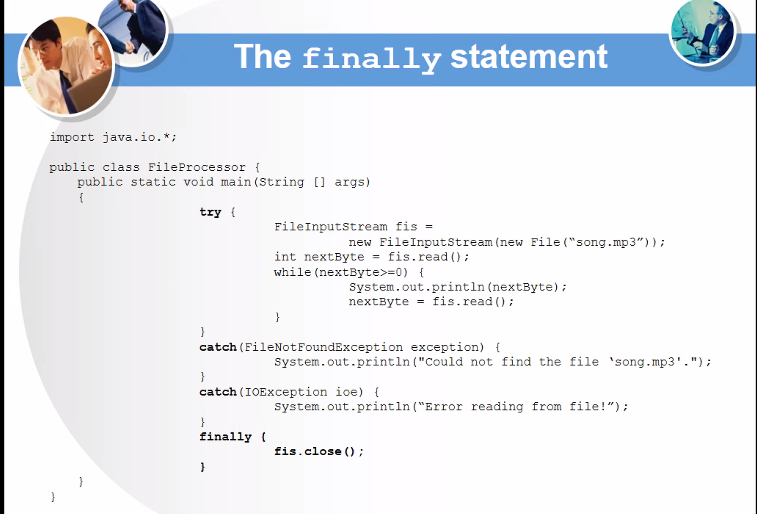
* **Catching**
  + Cú pháp try … catch … finally
  + Có thể có nhiều khối catch tương ứng với từng kiểu ngoại lệ được sinh ra. Các khối catch cần được sắp xếp theo thứ tự từ chi tiết đến chung chung
    - Thông thường, khi viết try … catch thì ta cần biết rõ cách thức xử lý tình huống có ngoại lệ.
* **Throwing**
  + Không xử lý ngoại lệ tại nơi nó xảy ra mà đẩy tiếp đến nơi gọi nó.
  + 2 từ khóa
    - throw (có thể đi kèm với new): phát sinh – đẩy đối tượng ngoại lệ ra khỏi khối lệnh.
    - throws: mệnh đề đi kèm với một phương thức, cho phép gán trách nhiệm xử lý ngoại lệ đó cho nơi gọi phương thức.

**3. Phân nhóm ngoại lệ - xác định cách tiếp cận phù hợp cho từng nhóm**

* **Checked Exception**
  + Ngoại lệ xảy ra do tác động bên ngoài
  + Nên try … catch
  + Vẫn cần đặt câu hỏi: bạn làm gì khi try … catch. Nếu không rõ ràng hoặc không chắc chắn thì bạn vẫn nên cân nhắc sử dụng throwing.
* **Unchecked Exception**
  + Ngoại lệ xảy ra do lỗi của lập trình viên
  + Bản chất: tạo cơ hội để lập trình viên nhìn thấy, truy vết và xử lý lỗi.
  + Không nên try … catch nhưng nên throws ….

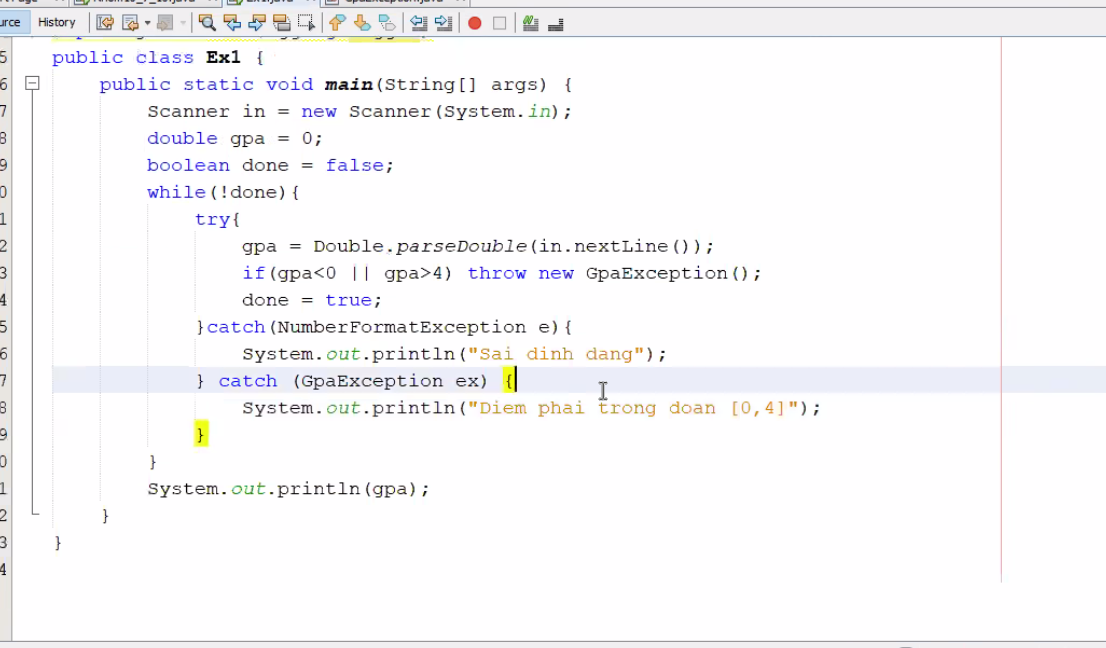
**4. Tự định nghĩa ngoại lệ**

**Câu lệnh finally:**

****

**Ưu tiên bắt cái nhỏ trước khi bắt cái to.**

**Try catch không dễ như mình nghĩ, ko được try catch tùy tiện.**

****

**Check và Uncheck**

**Ngoại lệ là cái xảy ra lúc chương trình đang chạy**

**Uncheck: Lỗi xảy ra do lập trình viên. Biên dịch báo và LTV sẽ phải báo lỗi phù hợp với throws … là 1 công cụ để sửa lỗi lập trình. Rơi vào nhóm lỗi số học / 0 , kích thước mảng vv,… ép kiểu sai,…**

**Check: Null, vv….. Chưa chắc lỗi lập trình. Phù hợp với try - catch. Phải biết và định nghĩa được tình huống đó và thêm vào để xử lý**

|  |  |
| --- | --- |
| **Uncheck Exception** | **Check Exception** |
| **+ Lỗi do lập trình viên**  **+ (chia cho không/ tràn mảng, v….)**  **+ Nên sử dụng throwing** | **+ lỗi do yếu tố bên ngoài.**  **+ buộc phải sử dụng try catch** |

**Tại mỗi thời điểm thì chỉ thực hiện một catch thôi**

**Catch sau phải là cha catch trước.**

|  |  |
| --- | --- |
| **Throw** | **Throws** |
| **+ Ném ra từ dòng lệnh** | **+ ném ra từ phương thức** |