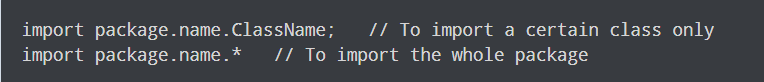
1. **Package trong Java**

* Một Package (gói) trong Java là một nhóm các class, interface và các package con tương tự, liên quan đến nhau.
* Lợi ích của việc sử dụng package
* Tái sử dụng: Trong một dự án, sẽ có lúc một logic được sử dụng ở nhiều nơi. Chúng ta sẽ tạo ra package chứa class implement logic đó và import vào những nơi cùng sử dụng. Đến lúc sửa chúng ta chỉ cần tìm đến package chứa class implement nó và sửa lại. Và đoạn code đó sẽ apply cho tất cả những chỗ khác.
* Quản lý code hiệu quả: Sử dụng package để phân tầng theo nhóm chức năng và chứa các class liên quan giúp chúng ta quản lý code hiệu quả
* Ngăn chặn được xung đột khi đặt tên. Ví dụ: Có thể có 2 class People (tên class giống hệt nhau) trong 2 package khác nhau.
* Các package có thể được coi là đóng gói dữ liệu (hoặc ẩn dữ liệu)
* Các package được chia làm hai loại:
* Các package được tích hợp sẵn từ Java API (Built-in packages)
* Các package do người dùng tự định nghĩa (đây là package do bạn tự tạo ra) – User defined packages

## Import packages trong Java

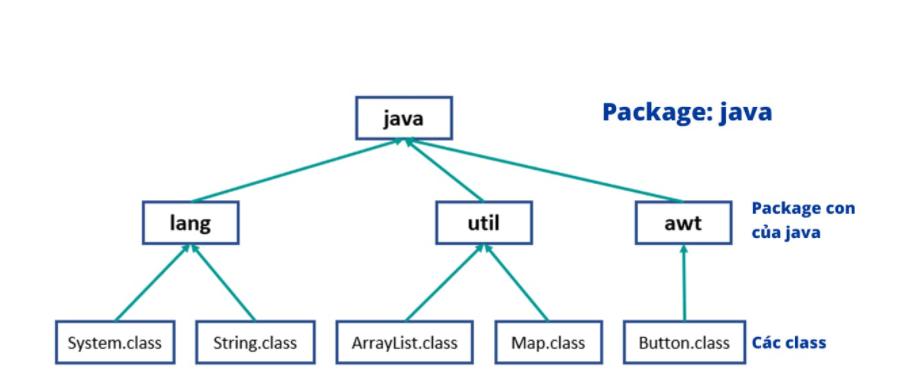
* Từ khoá import cho phép bạn nhập toàn bộ gói hoặc chỉ sử dụng các class và interface nhất định được xác định trong gói



* package: Tên package
* name: Tên package con
* ClassName: tên class của package con

## Built-in Package

* Các package tích hợp được java xây dựng sẵn dựa trên các nhu cầu thường xuyên của các lập trình viên.
* Sơ đồ minh hoạ



* Các package hay dùng được tích hợp sẵn trong Java bao gồm:
* [**java.lang**](https://docs.oracle.com/javase/8/docs/api/java/lang/package-summary.html): Chứa các lớp hỗ trợ ngôn ngữ (ví dụ: lớp được định nghĩa các kiểu dữ liệu nguyên thủy, các phép toán). Package này được import tự động.
* [**java.io**](https://docs.oracle.com/javase/8/docs/api/java/io/package-summary.html): Chứa lớp để hỗ trợ input / output (I/O)
* [**java.util**](https://docs.oracle.com/javase/8/docs/api/java/util/package-summary.html): Chứa các lớp tiện ích thực hiện các cấu trúc dữ liệu như danh sách liên kết, dictionary và hỗ trợ cho các hoạt động date / time.
* Ví dụ:



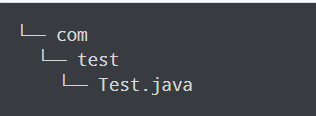
* java: package cha
* util : package con của java
* Scanner: là một class chứa trong package util.

1. **Use-defined packages**

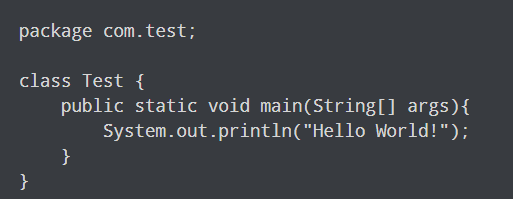
* Java cũng cho phép bạn tạo các gói theo nhu cầu của bạn. Các gói này được gọi là các gói do người dùng xác định.
* Define: sử dụng từ khoá package



* Ví dụ:



* Edit file Test.java



1. **Quy ước đặt tên package trong Java**

* Tên packages nên được viết thường hết tất cả các chữ cái.

1. **Xử lý Exception**

* Một exception là một sự kiện bất ngờ xảy ra trong quá trình thực thi chương trình. Nó ảnh hưởng đến dòng chảy của các hướng dẫn chương trình có thể khiến chương trình chấm dứt bất thường.
* Một ngoại lệ có thể xảy ra vì nhiều lý do. Một số trong số họ là: user input không hợp lệ, thiết bị hỏng, mất kết nối mạng, hết bộ nhớ đĩa, lỗi code, mở 1 tập tin không có sẵn
* Có 2 kiểu exception:
* Unchecked exception:
  + Do viết code sai, truyền đối null hoặc tham số không chính xác…
  + Các kiểu unchecked exceptions phổ biến:
    - **NullPointerException**: Ngoại lệ bị ném ra khi cố gắng truy cập một đối tượng có biến tham chiếu có giá trị hiện tại là null
    - **ArrayIndexOutOfBound**: Ngoại lệ khi cố gắng truy cập một phần tử vượt quá độ dài của mảng
    - **ArithmeticException**: Lỗi số học, chẳng hạn như chia cho 0.
* Checked exception:
  + Các ngoại lệ này thường là bị buộc phải bắt hoặc khai báo. Nếu quy tắc này không được tuân theo thì trình biên dịch sẽ không thực thi chương trình.
  + Chúng được kiểm tra bởi trình biên dịch tại thời điểm biên dịch và lập trình viên được nhắc để xử lý các ngoại lệ này.
  + Các kiểu unchecked exceptions phổ biến:
    - **IOException**: Ngoại lệ liên quan đến file input / output
    - **SQLException**: Ngoại lệ liên quan đến cú pháp SQL
* Xử lý ngoại lệ (Exception Handling) trong java là một cơ chế xử lý các lỗi runtime để có thể duy trì luồng bình thường của ứng dụng
* Quá trình xử lý exception được gọi là catch exception (bắt ngoại lệ), nếu Runtime System không xử lý được ngoại lệ thì chương trình sẽ kết thúc.

1. **Từ khoá try**

* Khối lệnh **try** trong java được sử dụng để chứa một đoạn code thực thi mà có thế trong quá trình thực thi nó sẽ xảy ra một ngoại lệ.
* Sau một khối lệnh **try**, bạn phải khai báo khối lệnh **catch** hoặc **finally**, hoặc cả hai.

1. **Từ khoá catch**

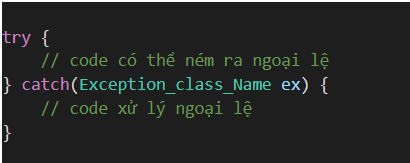
* Khối **catch** trong java được sử dụng để xử lý nếu xảy ra Exception, nếu không thì nó bị bỏ qua.
* Khối **catch** phải được sử dụng ngay sau khối **try**. Bạn có thể sử dụng nhiều khối **catch** với nhưng chỉ có một khối **try** duy nhất.

1. **Từ khóa finally**

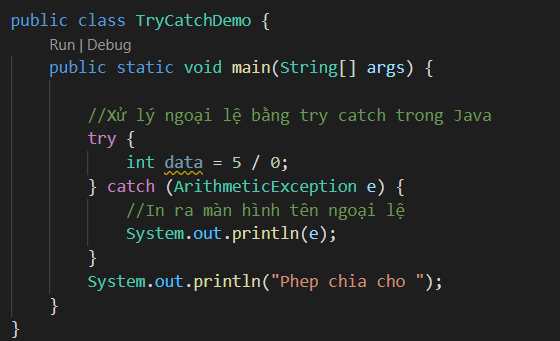
* Một khối **finally** cùng được thực thi sau khi khối **try** được thực thi thành công hoặc một ngoại lệ ném ra được xử lý. Nghĩa là khối **finally** luôn luôn được thực hiện dù có xảy ra ngoại lệ hay không.

1. **Try catch, Try finally, Try catch finally**

* Try catch:
* Cú pháp



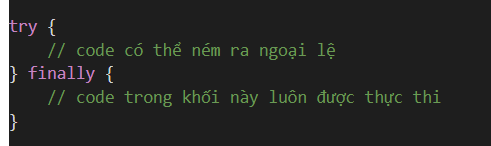
* Ví dụ



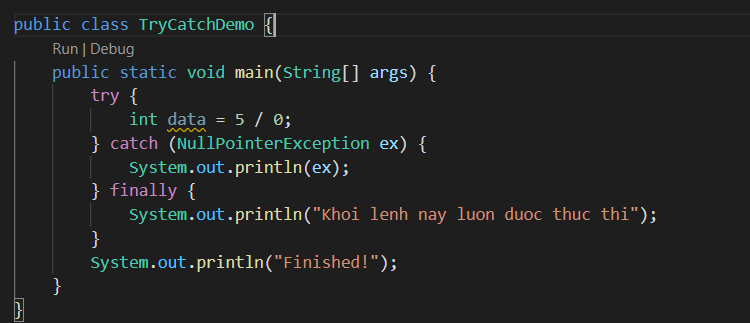
* Và đây là kết quả



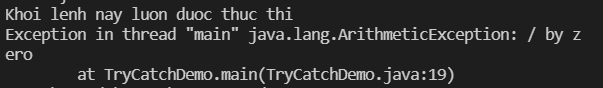
* Try finally: ngoại lệ xảy ra nhưng không được xử lý
* Cú pháp



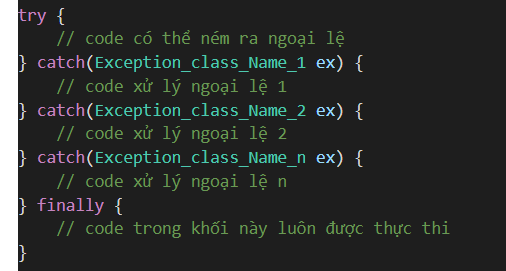
* Ví dụ:



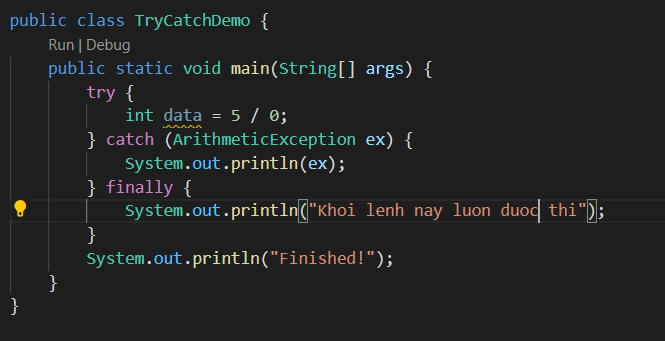
* Kết quả



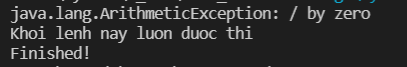
* Try catch finally: ngoại lệ xảy ra và được xử lý
* Cú pháp



* Ví dụ



* Kết quả

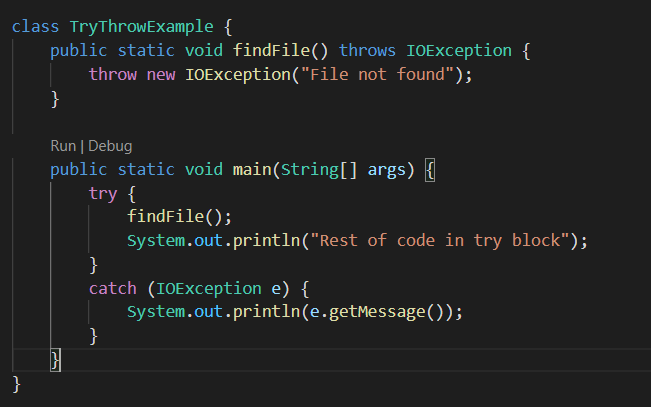


1. **Từ khóa throw và throws**

* Từ khóa throw
* Chúng ta có thể ném một trong hai ngoại lệ checked hoặc unchecked trong java bằng từ khóa *throw*. Từ khóa throw chủ yếu được sử dụng để ném [ngoại lệ tùy chỉnh](https://gpcoder.com/2461-custom-exception-trong-java/) (ngoại lệ do người dùng tự định nghĩa).
* Syntax:



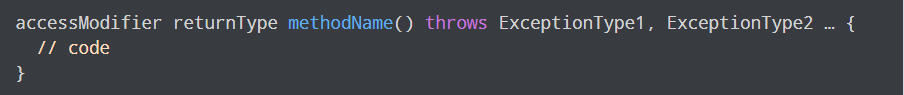
* Ví dụ:



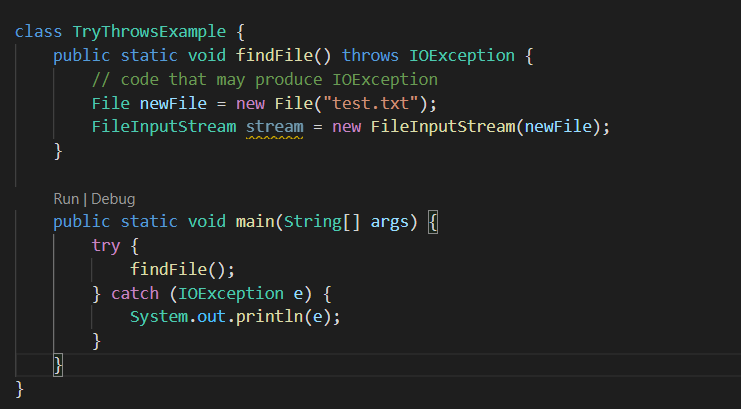
* Kết quả:



* Từ khóa throws
* Từ khóa throws trong java được sử dụng để khai báo một ngoại lệ. Nó thể hiện thông tin cho lập trình viên rằng có thể xảy ra một ngoại lệ, vì vậy nó là tốt hơn cho các lập trình viên để cung cấp các mã xử lý ngoại lệ để duy trì luồng bình thường của chương trình.
* Ngoại lệ nào nên được khai báo: Chỉ những ngoại lệ checked bởi:
  + Ngoại lệ **unchecked**: nằm trong sự kiểm soát của bạn.
  + **error**: nằm ngoài sự kiểm soát của bạn
* **Lợi ích của từ khóa throws trong java**
  + Ngoại lệ checked có thể được ném ra ngoài và được xử lý ở một hàm khác.
  + Cung cấp thông tin cho caller của phương thức về các ngoại lệ.
* Syntax



* Ví dụ



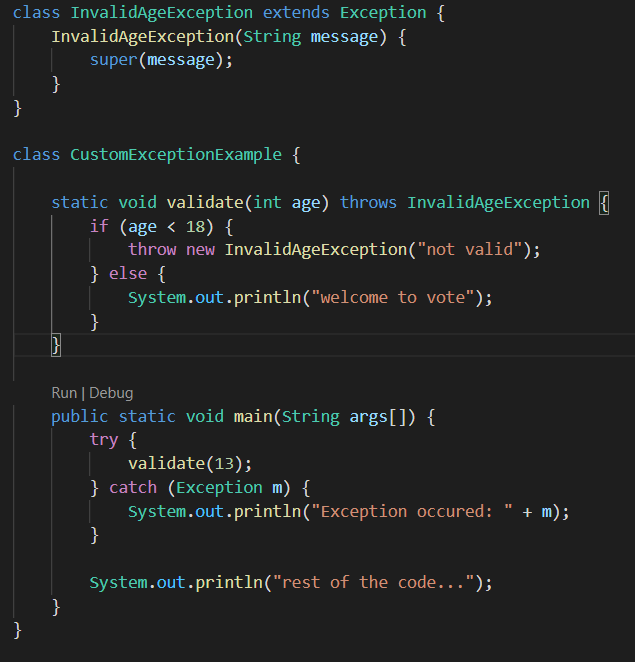
* Kết quả



* Throws vs try…catch
  + Viết try…catch cho mỗi phương thức sẽ tẻ nhạt và code trở nên dài và ít đọc được
  + Throws cũng hữu ích khi bạn đã kiểm tra ngoại lệ (ngoại lệ phải được xử lý) mà bạn không muốn catch trong phương thức hiện tại của mình

1. **Tự custom Exception class**

* **Custom Exception** là ngoại lệ do người dùng tự định nghĩa. Custom Exception trong Java được sử dụng để tùy biến ngoại lệ theo yêu cầu của người dùng. Bởi sự giúp đỡ của loại ngoại lệ này, người dùng có thể có riêng kiểu và thông điệp ngoại lệ riêng cho mình.
* Thông thường, để tạo ra custom exception thuộc loại **checked** chúng ta kế thừa từ lớp [**Exception**](https://gpcoder.com/2430-xu-ly-ngoai-le-trong-java-exception-handling/). Để tạo custom exception thuộc loại **unchecked** chúng ta kế thừa từ lớp **RuntimeException**.
* Trong các ứng dụng thực tế, thông thường custom exception được tạo là checked exception.
* Ví dụ:



* Kết quả:

