**Java là gì?**

Java là một ngôn ngữ lập trình hướng đối tượng, được thiết kế để chạy trên nhiều nền tảng khác nhau (platform-independent). Nó được Sun Microsystems (nay là Oracle) phát triển và phát hành lần đầu vào năm 1995. Java được biết đến với tính ổn định, bảo mật và hiệu năng cao, khiến nó trở thành một trong những ngôn ngữ lập trình phổ biến nhất thế giới.

**Tại sao Java được ưa chuộng?**

* **Hướng đối tượng:** Java hỗ trợ đầy đủ các khái niệm của lập trình hướng đối tượng như lớp, đối tượng, kế thừa, đa hình, đóng gói. Điều này giúp code dễ đọc, dễ bảo trì và mở rộng.
* **Độc lập nền tảng:** Nhờ có JVM (Java Virtual Machine), các chương trình Java có thể chạy trên bất kỳ hệ điều hành nào mà có cài đặt JVM.
* **Bảo mật:** Java có các cơ chế bảo mật mạnh mẽ để ngăn chặn các cuộc tấn công như virus, mã độc.
* **Hiệu năng cao:** Mặc dù là ngôn ngữ được biên dịch, Java vẫn có thể đạt được hiệu năng khá tốt nhờ vào việc tối ưu hóa của JVM.
* **Cộng đồng lớn:** Java có một cộng đồng người dùng rất lớn, điều này có nghĩa là bạn dễ dàng tìm thấy tài liệu, thư viện và hỗ trợ khi cần.
* **Thư viện phong phú:** Java có một kho thư viện khổng lồ (Java API) hỗ trợ cho nhiều lĩnh vực khác nhau như giao diện đồ họa, xử lý dữ liệu, kết nối cơ sở dữ liệu,...

**Java trong lập trình web**

Java đóng vai trò quan trọng trong việc phát triển các ứng dụng web. Dưới đây là một số lý do:

* **Servlet và JSP:** Đây là hai công nghệ cốt lõi của Java để xây dựng các ứng dụng web. Servlet là những lớp Java được sử dụng để xử lý các yêu cầu HTTP, trong khi JSP (JavaServer Pages) cho phép nhúng mã Java vào HTML để tạo ra nội dung động.
* **Framework:** Có rất nhiều framework Java được sử dụng để phát triển web như Spring, Hibernate, Struts. Các framework này cung cấp các tính năng sẵn có, giúp giảm thời gian phát triển và tăng chất lượng code.
* **Enterprise JavaBeans (EJB):** EJB là một tập hợp các giao diện và các lớp được sử dụng để viết các thành phần doanh nghiệp. EJB giúp đơn giản hóa việc phát triển các ứng dụng phức tạp, phân tán.
* **Web Services:** Java cung cấp các công nghệ để xây dựng và tiêu thụ các web service, giúp các ứng dụng có thể giao tiếp với nhau qua mạng.

**Lợi ích của Java trong lập trình web**

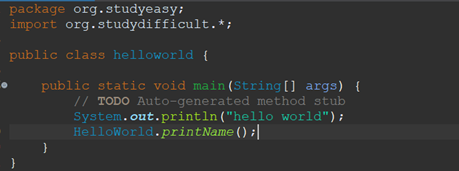
* **Ổn định và đáng tin cậy:** Các ứng dụng web Java thường rất ổn định và có thể chịu được tải lớn.
* **Bảo mật:** Java có các cơ chế bảo mật mạnh mẽ, giúp bảo vệ ứng dụng web khỏi các cuộc tấn công.
* **Hiệu năng cao:** Các ứng dụng Java có thể xử lý được một lượng lớn request đồng thời.
* **Phát triển nhanh:** Các framework Java giúp rút ngắn thời gian phát triển ứng dụng.
* **Khả năng mở rộng:** Các ứng dụng Java dễ dàng mở rộng để đáp ứng nhu cầu tăng trưởng.

**Các ứng dụng phổ biến của Java trong web**

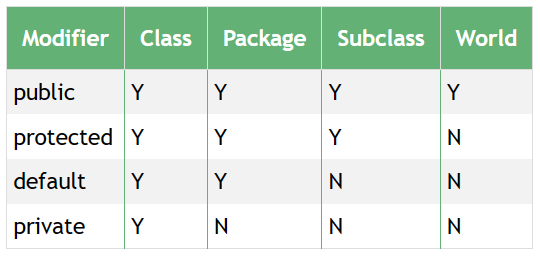
* **Ứng dụng web doanh nghiệp:** Java thường được sử dụng để xây dựng các ứng dụng web lớn, phức tạp cho các doanh nghiệp.
* **Ứng dụng thương mại điện tử:** Nhiều trang web thương mại điện tử lớn được xây dựng bằng Java.
* **Ứng dụng di động:** Java cũng được sử dụng để phát triển các ứng dụng di động chạy trên nền tảng Android.

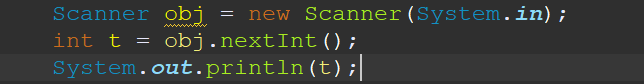
**Tóm lại,** Java là một lựa chọn tuyệt vời cho việc phát triển các ứng dụng web, đặc biệt là các ứng dụng đòi hỏi tính ổn định, bảo mật và hiệu năng cao.

IDE(Intergrated Developer Enviroment): eclipse

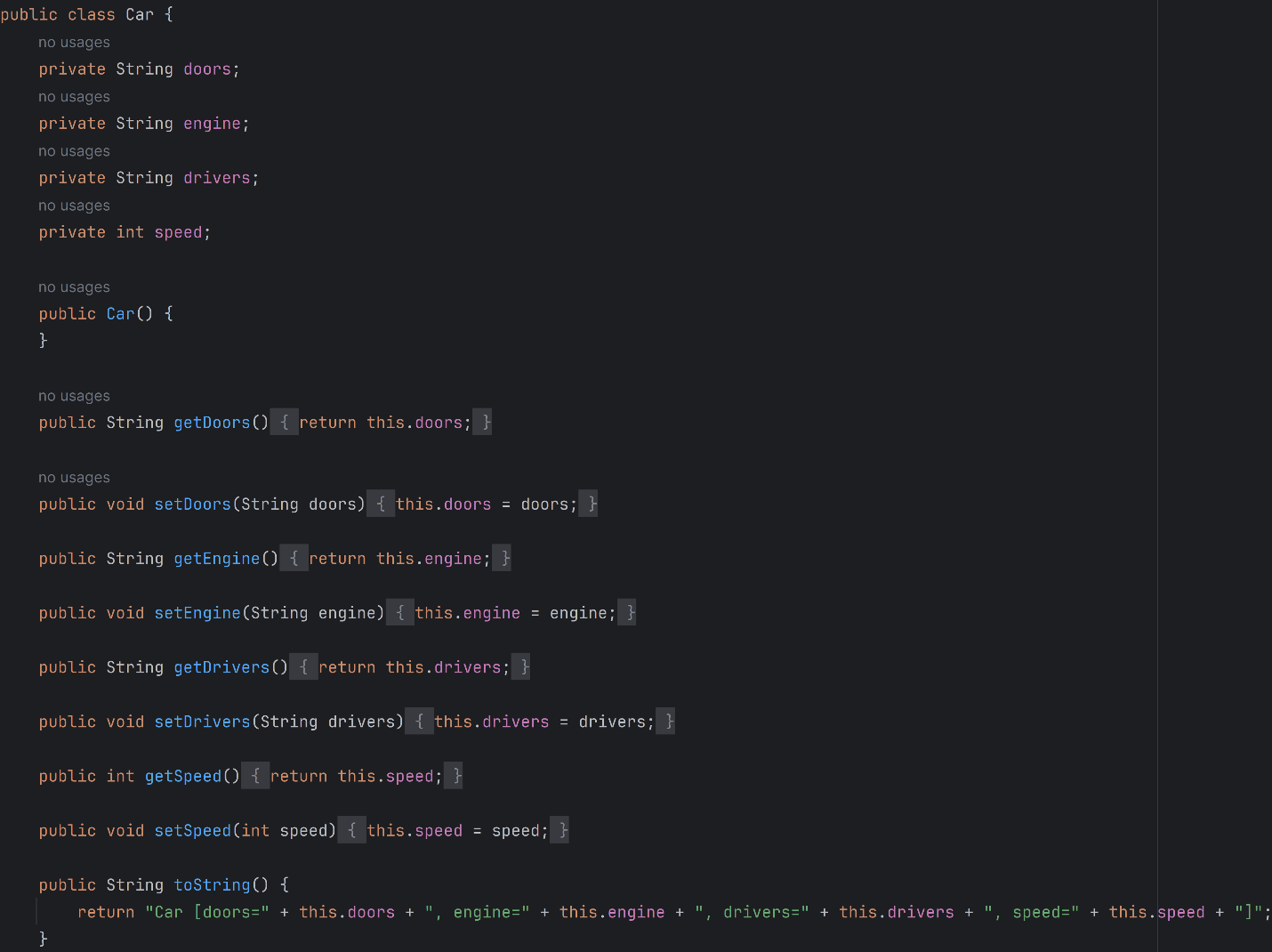
* Enviroment: JDK (17)
* Hello world
  + Package: cài đặt nơi chứa class
  + File .jar : chứa thư viện Java
  + Public, static, void: Java reference for process
    - Public: giống c++
    - Static: có thể truy cập, không thể override
    - Void: không có kiểu trả về
    - 

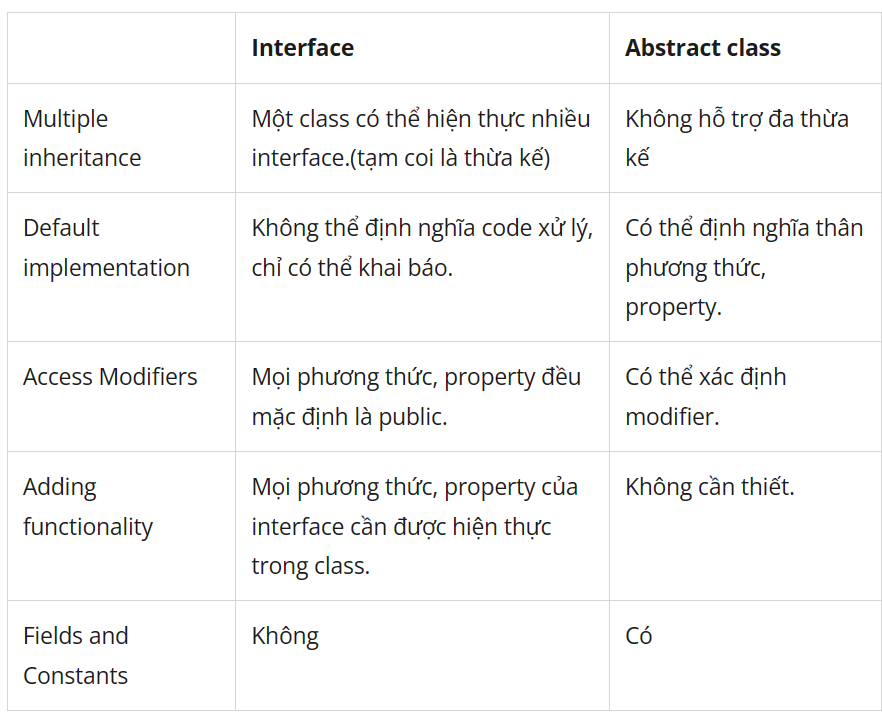
* Access Modifiders: (Khả năng truy cập)
  + **Private**: The access level of a private modifier is only within the class. It cannot be accessed from outside the class.
  + **Default**: The access level of a default modifier is only within the package. It cannot be accessed from outside the package. If you do not specify any access level, it will be the default.
  + **Protected**: The access level of a protected modifier is within the package and outside the package through child class. If you do not make the child class, it cannot be accessed from outside the package.
  + **Public**: The access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package.



* + Nhận kí tự từ bàn phím:
  + Xuất ra màn hình:
* 

* + Getter - Setter - ToString:
    - Getter - Setter: Dùng để kiểm soát biến trong class. Khi truy cập và thay đổi code
    - ToString: Ra đời khi muốn biểu diễn 1 đối tượng dưới dạng chuỗi. Khi không Override toString Java sẽ invoke toString method trên object.



* Abstract :
  + Abstract Class: Không thể khai báo object mà phải được kế thừa từ sub-class:
  + Abstract Method: chỉ có thể được sử dụng trong abstract class. Không được phép có body và sẽ phải override lại từ sub-class.
* Interface:
  + Có thể khai báo Object.
  + Kế thừa từ interface dùng từ khóa implements keywork:
  + Biến trong interface mặc định là: public static final.
  + Chỉ chứa các abstract public method.
* 
* Override Rule:

**Overriding and Access Modifiers:**

**Final methods can not be overridden**

**Static methods can not be overridden(Method Overriding vs Method Hiding):**

**Private methods can not be overridden**

**Invoking overridden method from sub-class**

* + Super keyword: Dùng để gọi Base Class Constructor, gọi Hàm, Biến trong Base Class

If the super-class overridden method does not throw an exception, the subclass overriding method can only throw the [unchecked exception](https://www.geeksforgeeks.org/checked-vs-unchecked-exceptions-in-java/), throwing a checked exception will lead to a compile-time error.

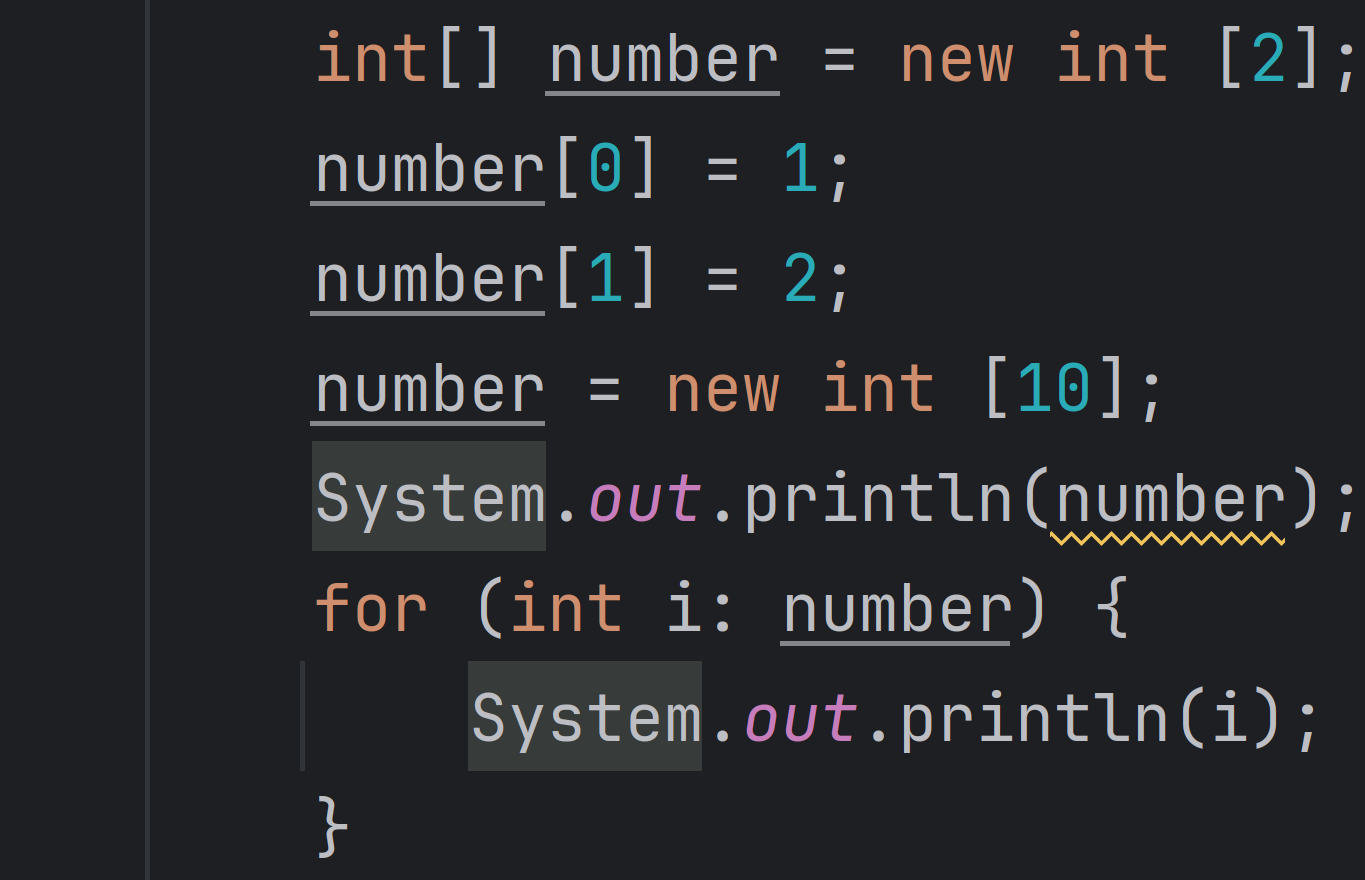
* @Override Annotation

1) You’ll get a compile-time error if the programmer makes a mistake while overriding, such as using the wrong method name or parameter types. Because you are informing the compiler that you are overriding this method by using this annotation. If you don’t use the annotation, the sub-class method will be treated as a new method in the subclass (rather than the overriding method).

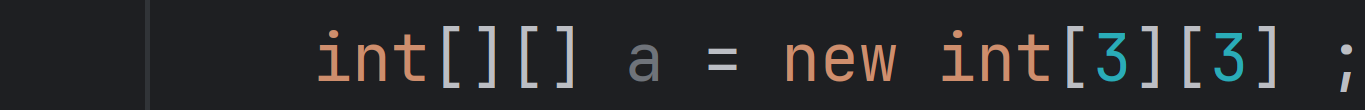
2) It improves the code’s Readability. If you change the signature of an overridden method, all sub-classes that override it will throw a compilation error, which will eventually lead to you changing the signature in the subclasses. If you have a large number of classes in your application, this chú thích will greatly assist you in identifying the classes that need to be changed when a method’s signature is changed.

From <<https://www.geeksforgeeks.org/the-override-annotation-in-java/>>

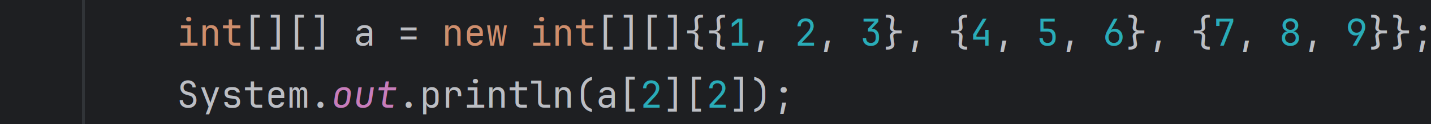
* Mảng: Khai báo, thêm, xóa, sửa.

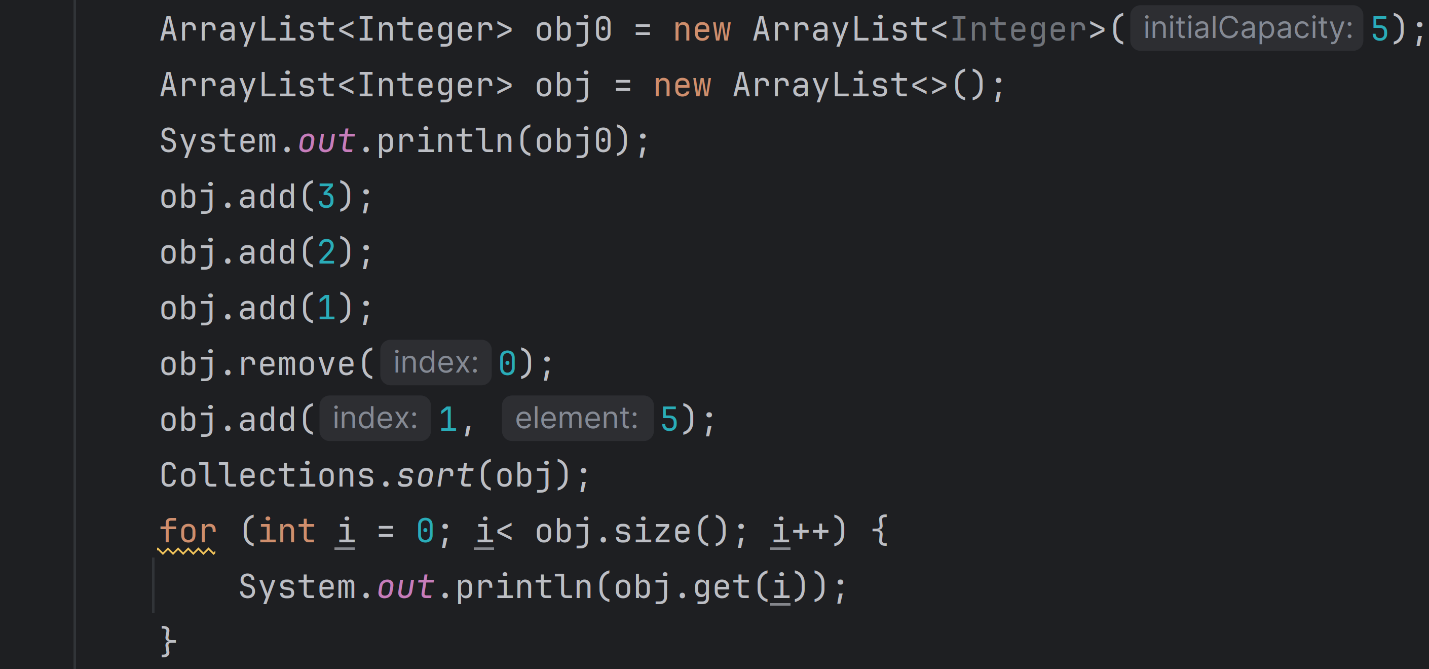


* Mảng 2 chiều:
  + Khởi tạo:

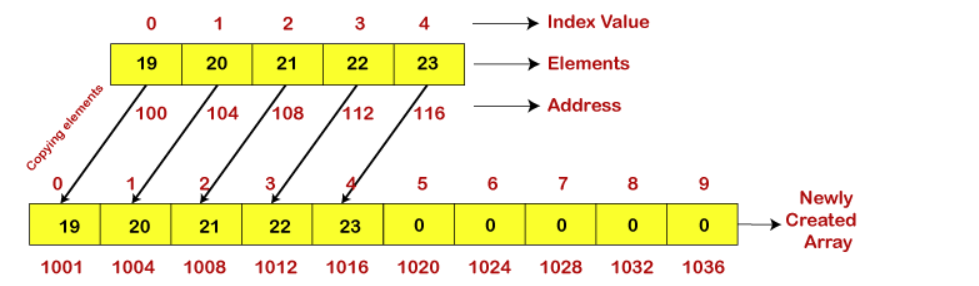


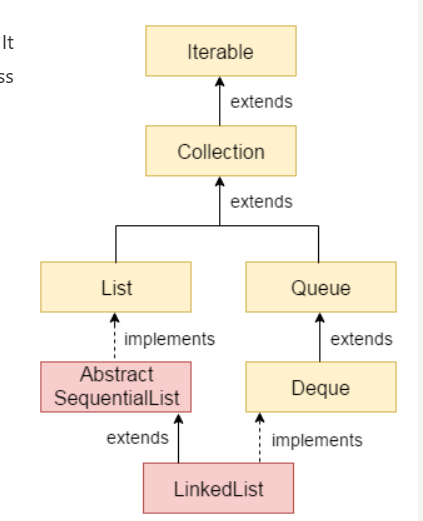
* Khởi tạo:

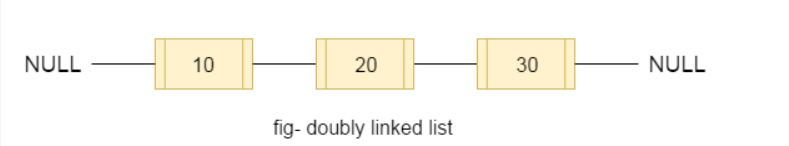
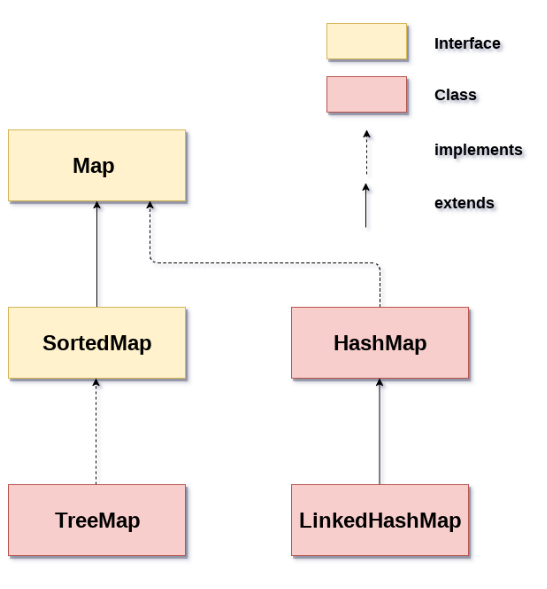


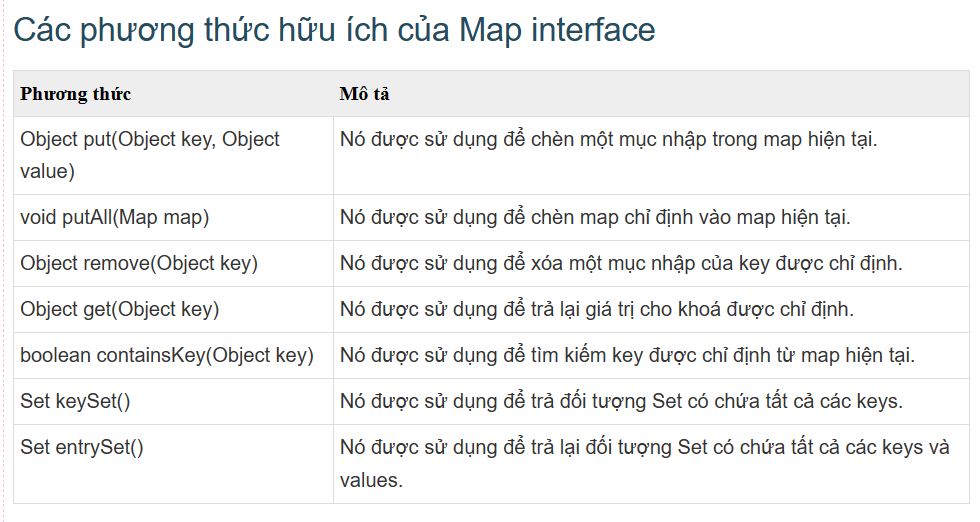
* Collection FrameWork:
  + Array List (Class):
    - Được implement từ Interface List.
* 
  + Được cung cấp các hàm để phục của một dynamic array (cùng nhớ không cố định, có thể tăng độ dài)
  + Khai báo, thêm, xóa, sửa, xắp xếp:
* 

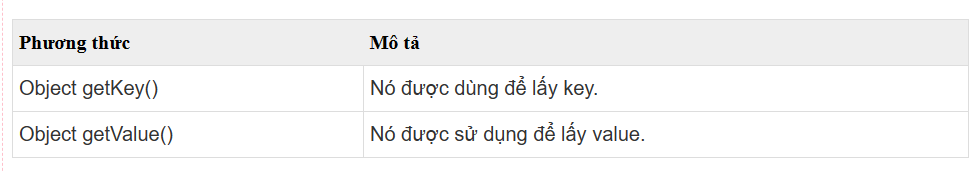
* Cách hoạt động của dynamic array: Trong dynamic array, khi khai báo 1 độ dài của mảng cố định, nhưng add thêm nhiều element hơn capacity, dynamic array sẽ chuyển đến 1 vùng nhớ khác nhiều không gian hơn và copy toàn bộ value của array cũ vào array mới.

* 

* Linked List:
  + 

* 
* Chức năng thêm, xóa, sửa, lật ngược(peek).
* Map - Set:
* 

* A Map can't be traversed, so you need to convert it into Set using keySet() or entrySet() method.
* 



* Try - Catch - Finally:

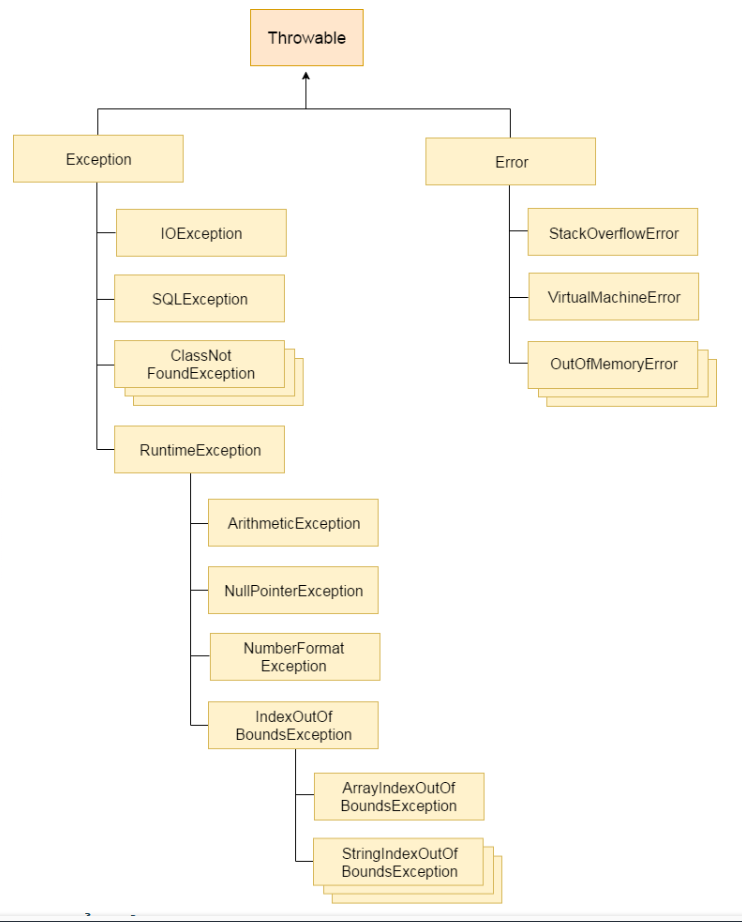
**Exception Handling** trong java hay xử lý ngoại lệ trong java là một cơ chế mạnh mẽ để xử lý các lỗi runtime để có thể duy trì luồng bình thường của ứng dụng.

Trong bài này, chúng ta sẽ tìm hiểu về ngoại lệ (Exception) trong java, các kiểu ngoại lệ và sự khác biệt giữa các ngoại lệ **checked** và **unchecked**.

From <<https://viettuts.vn/exception-handling>>

Giả sử có 10 câu lệnh trong chương trình của bạn và xảy ra trường hợp ngoại lệ ở câu lệnh 5, phần còn lại của chương trình sẽ không được thực thi, nghĩa là câu lệnh 6 đến 10 sẽ không chạy. Nếu chúng ta thực hiện xử lý ngoại lệ, phần còn lại của câu lệnh sẽ được thực hiện. Đó là lý do tại sao chúng ta sử dụng xử lý ngoại lệ trong java

From <<https://viettuts.vn/exception-handling>>



Có hai loại ngoại lệ chính là: **checked** và **unchecked**. Còn Sun Microsystem nói rằng có ba loại ngoại lệ:

1. Checked Exception
2. Unchecked Exception
3. Error

From <<https://viettuts.vn/exception-handling>>

1. Checked Exception

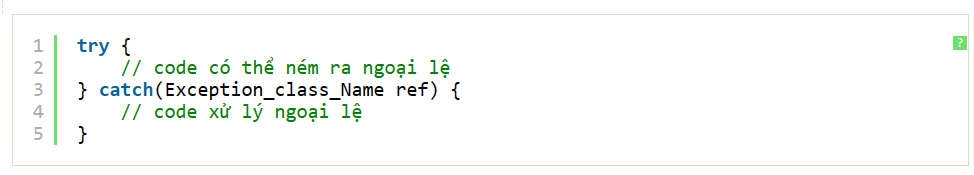
Các lớp extends từ lớp Throwable ngoại trừ RuntimeException và Error được gọi là checked exception, ví dụ như Exception, SQLException vv. Các checked exception được kiểm tra tại compile-time.

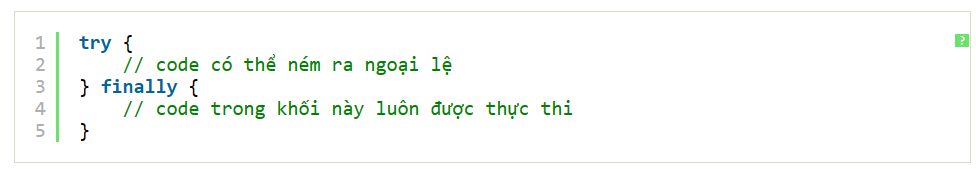
From <<https://viettuts.vn/exception-handling>>

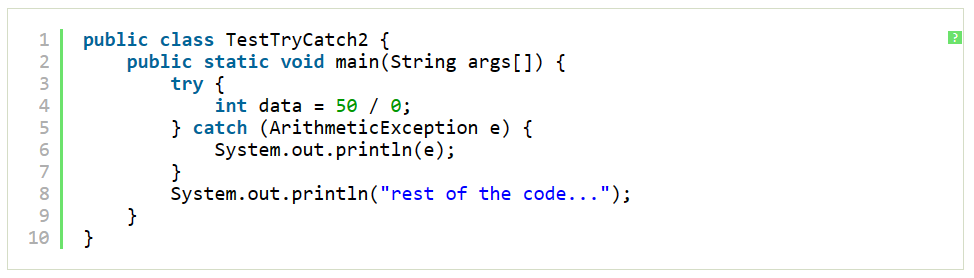
2. Unchecked Exception

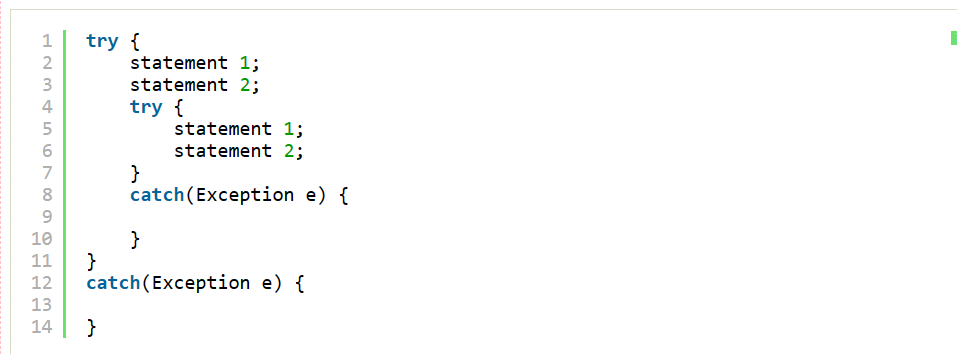
Các lớp extends từ RuntimeException được gọi là unchecked exception, ví dụ: ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException,... Các ngoại lệ unchecked không được kiểm tra tại compile-time mà chúng được kiểm tra tại runtime.

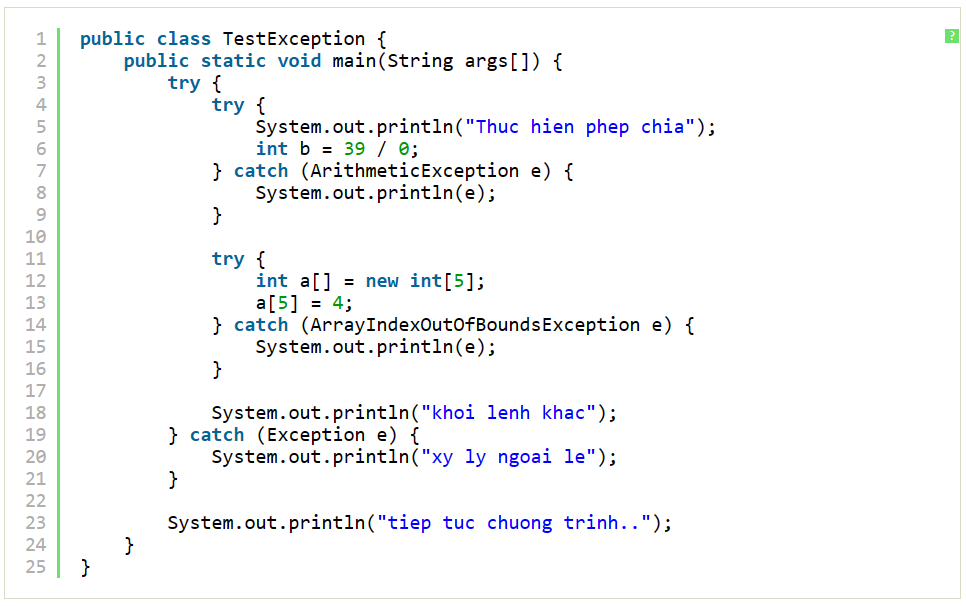
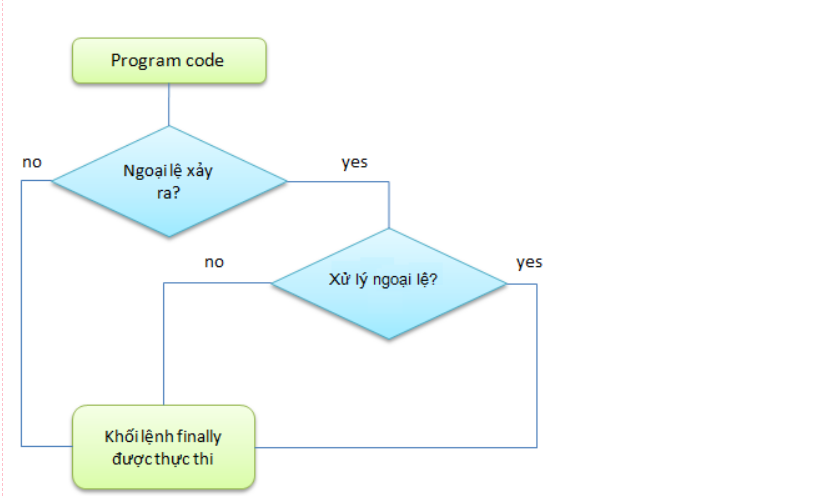
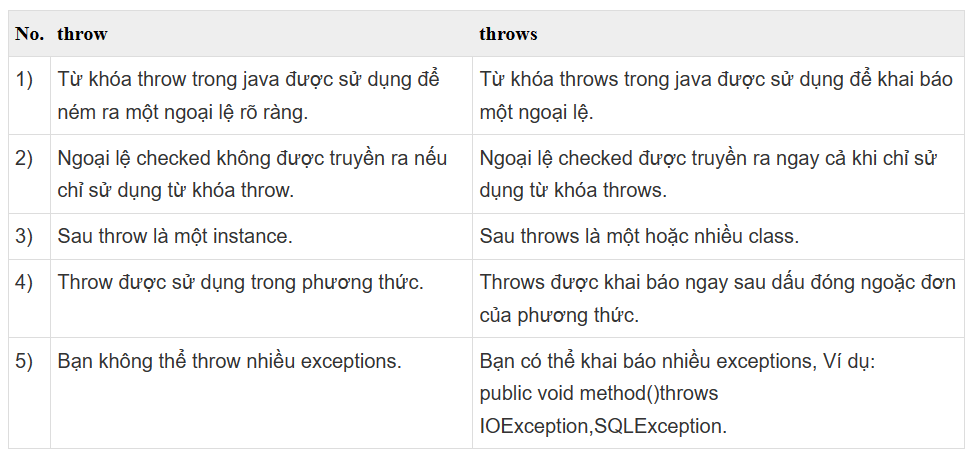
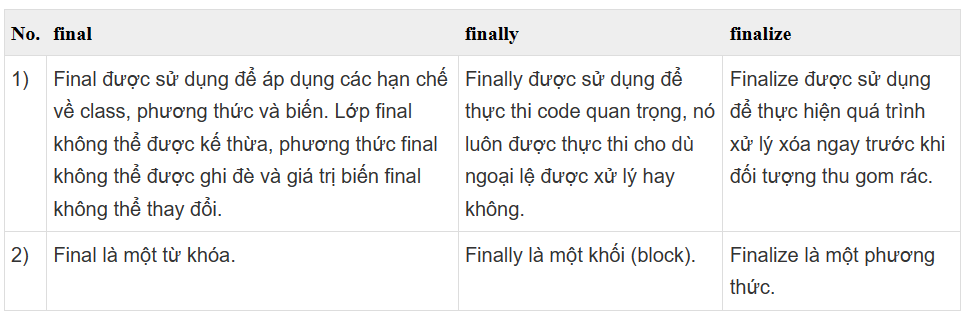
From <<https://viettuts.vn/exception-handling>>







* Khối Try Lồng nhau:
* 

* 
* Finally:
* 
* Thow - Thows:
* 
* Finally - Final - Finalize:
* 

* 