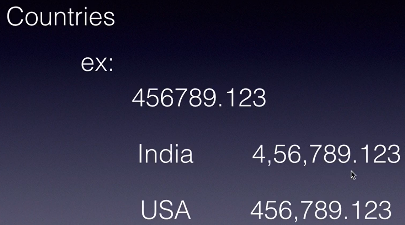
1. **Internationalization** (I18N - có 18 chữ giữa I và N)

Hỗ trợ ngôn ngữ và định dạng tùy vào các khu vực nhất định (Locale)

Các framework như spring sẽ sử dụng resourceBundles để truyền vào

* 1. **NumberFormat**

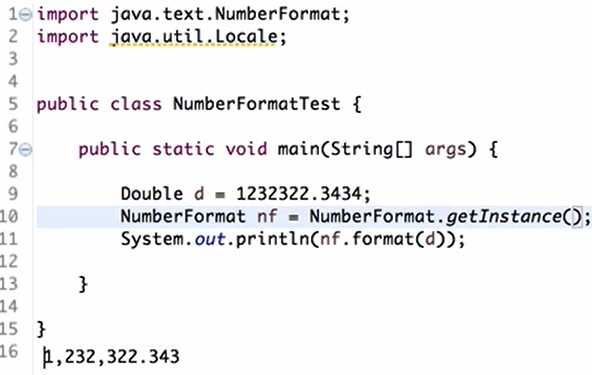
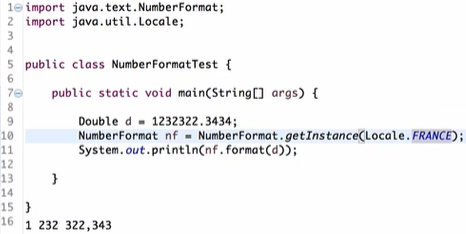
Hiển thị định dạng number ở countries là khác nhau (định dạng số theo Locale)



*Phím tắt: Alt + Shift + D + J*

Các method hay dùng

* **getInstance()**

* **getCurrencyInstance()**

biểu diễn đơn vị tiền tệ

* **getPercentageInstance()**

Tính toán tỉ lệ %

* **getNumberInstance()**
* **setMinimumFractionDigits(int n)**
* **setMaximumFractionDigits(int n)**

set tối thiếu (tối đa) có bao nhiêu số đằng sau dấu ,

* **setMinimunIntegerDigits(int n)**
* **setMaximunIntegerDigits(int n)**

set tối thiếu (tối đa) có bao nhiêu số đằng trước dấu ,

* 1. **DateFormat**

Có 2 đối tượng là DateFormat và SimpleDateFormat

* + 1. **DateFormat**



Các method hay dùng

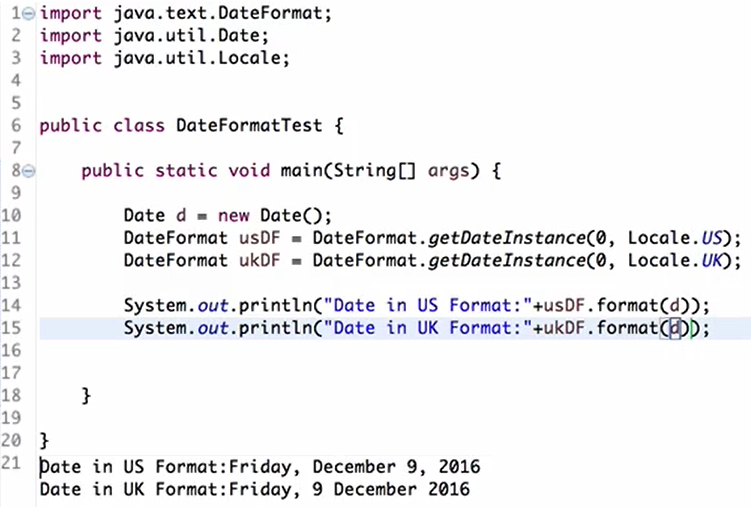
* **getInstance()**
* **getDateInstance([int style])**



* **getDateInstance([int style])**
* **format(Date d)**
  + style
    - DateFormat.FULL
    - DateFormat.LONG
    - DateFormat.MEDIUM
    - DateFormat.SHORT



VD:

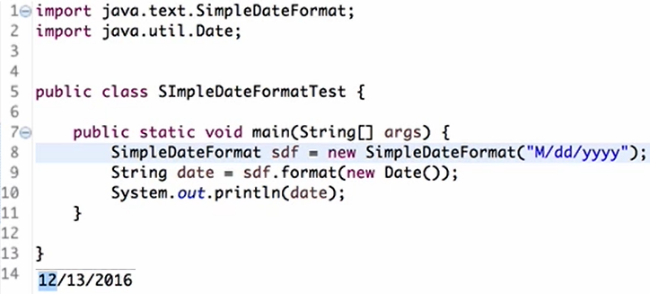
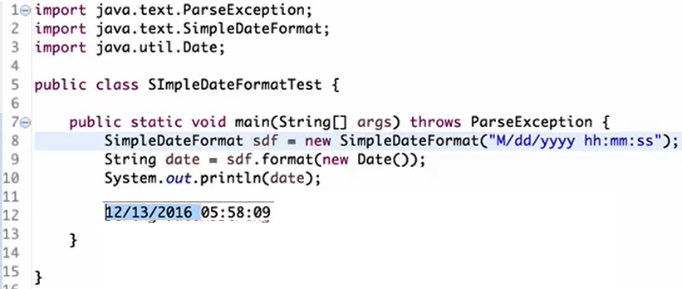


* + 1. **SimpleDateFormat**

Các method hay dùng

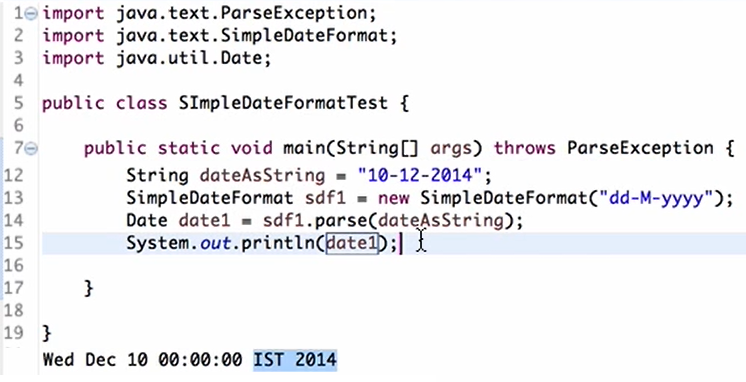
* + - * **format(Date date)**

Convert Date 🡪 String

* + - * **parse(String date)**

Convert String 🡪 Date

****

1. **Annotations**

* Có từ Java 5
* Là 1 dạng meta data (đặc tả dữ liệu cho đối tượng, giá trị nào đó)
* Dùng để
  + Chỉ dẫn cho trình biên dịch

VD: @Deprecated, @Override và @SuppressWarnings

* + Chỉ dẫn trong compile time
  + Chỉ dẫn trong runtime

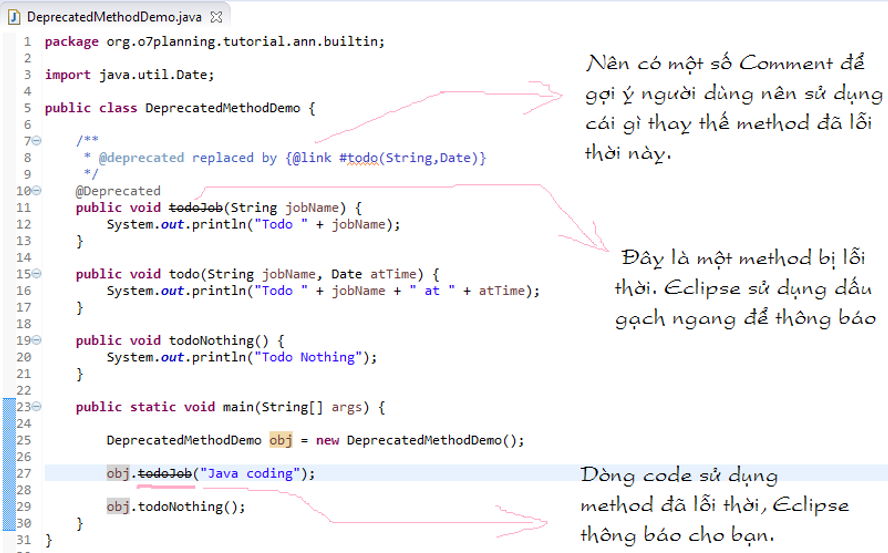
VD: Java Reflection

* **1 số Annotation sẵn có**
  + **@Deprecated**

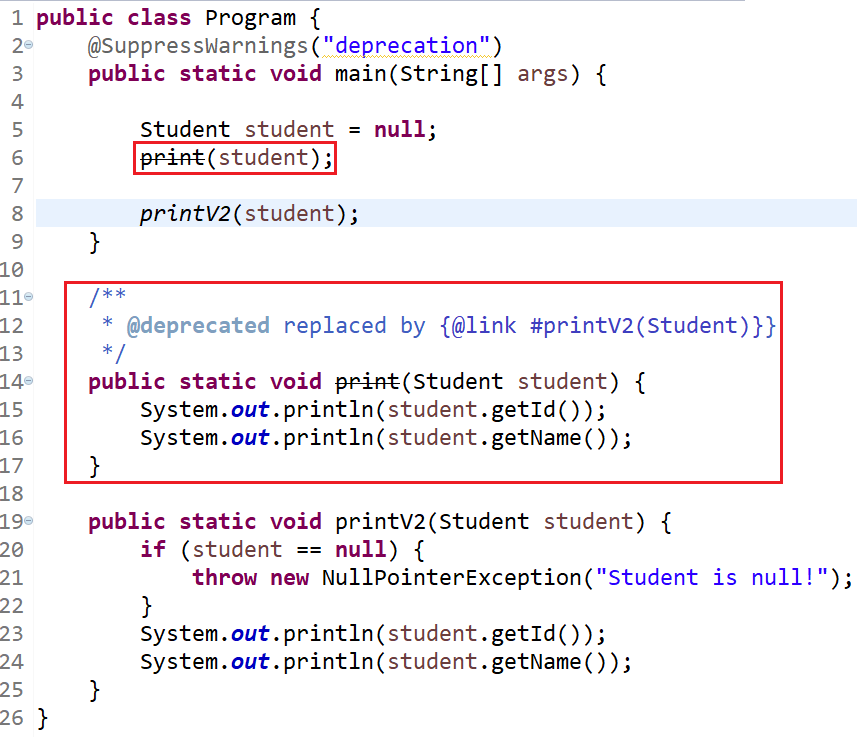
Để chú thích 1 cái gì đó bị lỗi thời, không nên sử dụng.

Java sẽ gợi ý class hoặc method thay thế

VD1:



VD2:



* + **@Override**

Sử dụng để đánh dấu ghi đè method trong Sub Class.

Nếu method này không hợp lệ với một method trong class cha, trình biên dịch sẽ thông báo cho bạn một lỗi

Không bắt buộc phải khai báo @Override

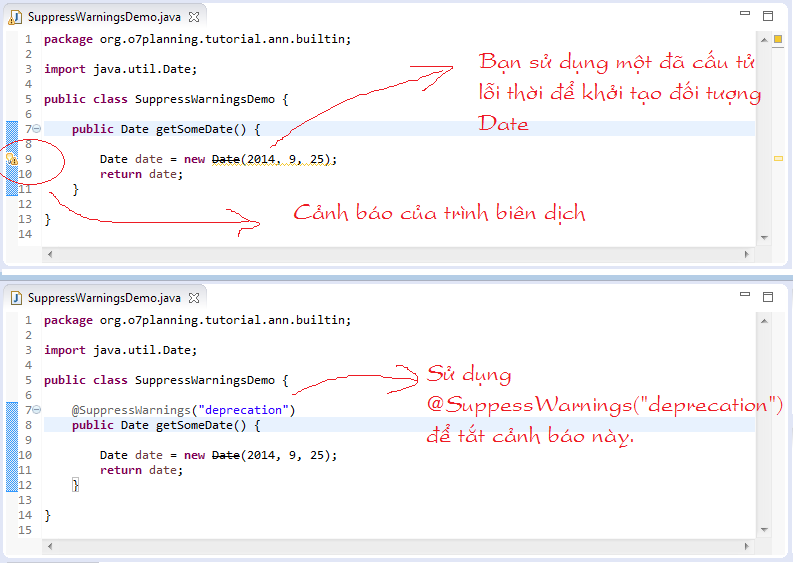
Khi sửa tên method ở Super Class thì Java sẽ không tự động sửa tên method ở Sub Class 🡪 không Override nữa 🡪 compile báo lỗi

* + **@SuppressWarnings**

Làm cho trình biên dịchkhông cảnh báo một vấn đề của method nào đó.

VD: nếu trong một method có gọi tới một method khác đã lỗi thời, hoặc bên trong method có một ép kiểu không an toàn, trình biên dịch có thể tạo ra một cảnh báo.

* Có thể tắt các cảnh báo này bằng cách chú thích method này bằng @SuppressWarnings

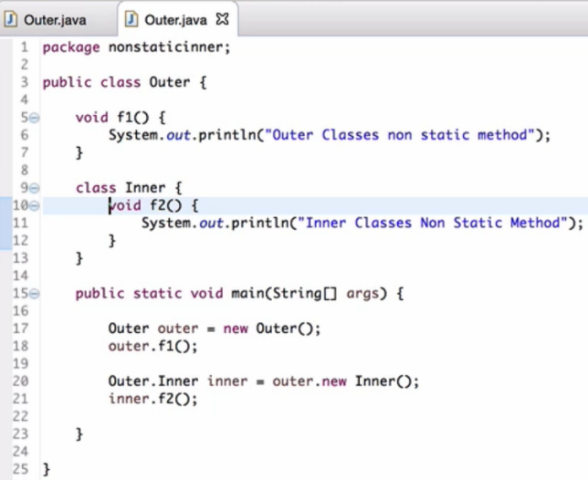


Custom annotation

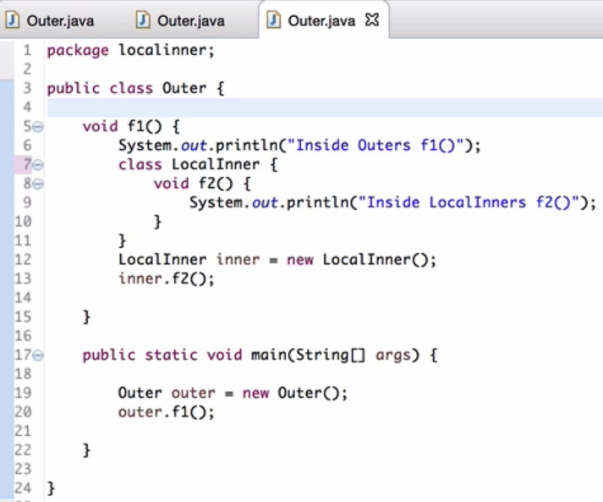
Tham khảo ở đây: <https://o7planning.org/vi/10197/huong-dan-java-annotation#a18827>

1. **Inner class**

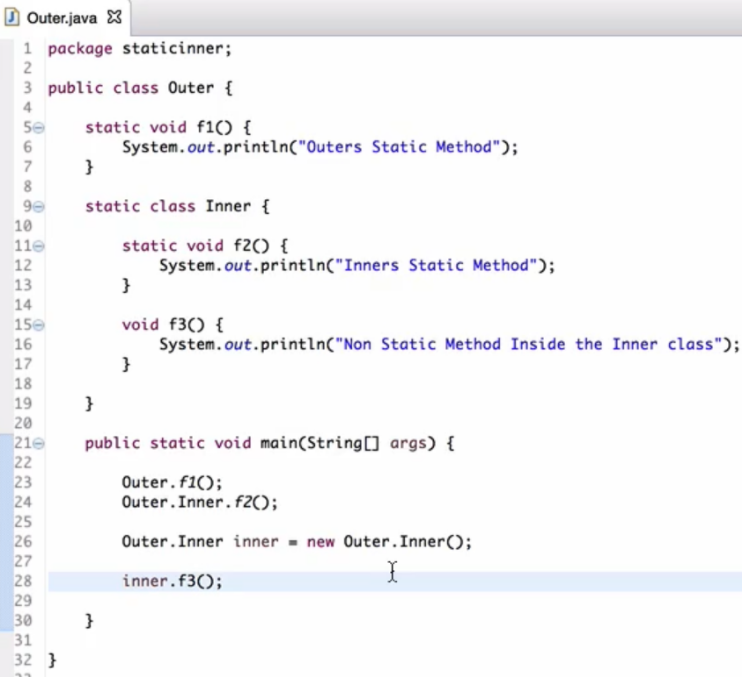
* Trong 1 file .java chỉ có 1 class được public
* Có 2 cách khai báo là:
  + **Trong class**

****

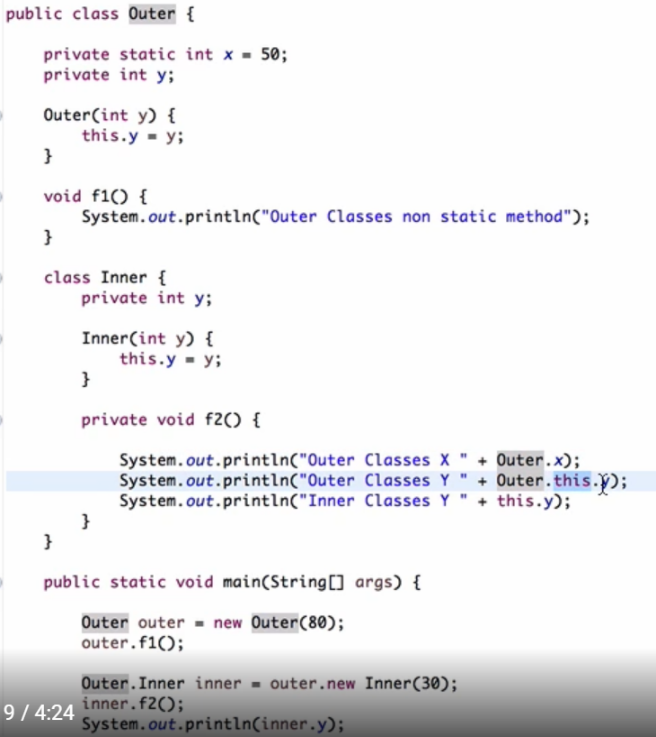
* + **Trong method** (giống CTE, chạy xong method là xóa luôn)

****

* Static & Inner Class

****

Cách truy cập variable, method

****

1. **Garbage Collector**

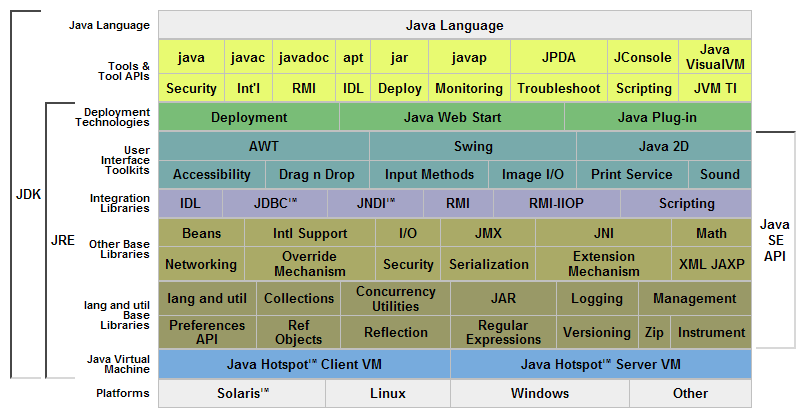
* Là cơ chế dọn rác của Java, là quá trình tự động thực thi nhiệm vụ quản lý bộ nhớ, thu hồi những Object không có tham chiếu
* Tự động Java sẽ kiểm soát các Object không có tham chiếu (không cần gọi để hủy như C/C++)
* Trong quá trình chạy chương trình, các object được tạo ở vùng nhớ heap. Sau cùng, sẽ có một vài object mà chương trình không cần đến. Các object này sẽ được garbage collector truy tìm và xóa bỏ để thu hồi lại dung lượng bộ nhớ
* Garbage Collector có nhiệm vụ
  + Đảm bảo rằng bất kỳ đối tượng nào có các tham chiếu thì cần phải duy trì trong bộ nhớ heap
  + Lấy lại bộ nhớ được sử dụng bởi các object không còn tham chiếu
* 1 Object trở thành rác khi:
  + Tất cả các tham chiếu của đối tượng đều = null (object = null)
  + Đối tượng cha là 1 tập null
  + 1 đối tượng được tạo bên trong 1 khối và tham chiếu của nó nằm ngoài phạm vi của khối đó, khi chương trình chạy qua khối đó thì đối tượng được tạo bên trong khối sẽ trở thành rác (VD: tạo biến i trong vòng for)
* Hoạt động theo mô hình
  + Bước 1: xác định và đánh dấu các object không có tham chiếu (unreferenced object).
  + Bước 2: xóa các object được đánh dấu sau bước 1

Bộ nhớ sau khi tiến hành bước 2 có thể được "dồn" lại, điều này đồng nghĩa với việc các object đang hoạt động sẽ nằm ở các ô nhớ sát nhau tại phần bắt đầu của heap. Quá trình "dồn" này giúp cấp phát bộ nhớ cho các object mới dễ dàng hơn.

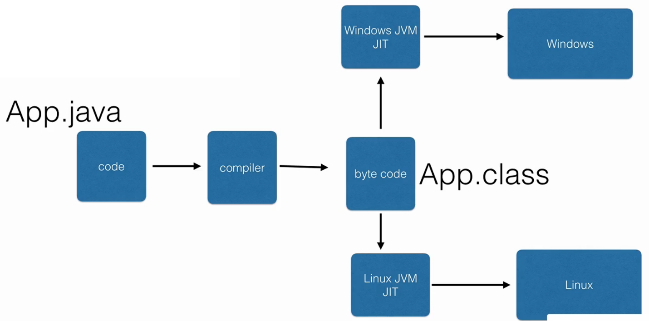
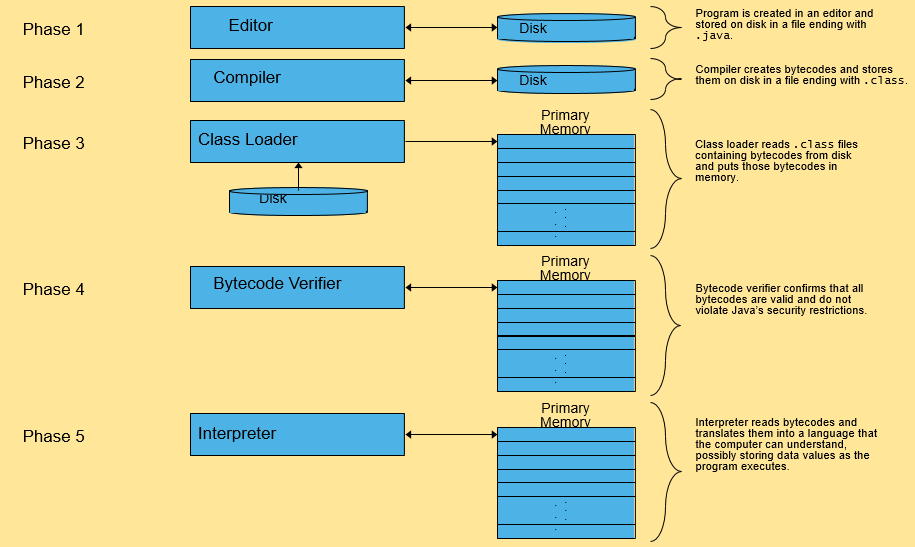
* Để yêu cầu Java xóa object thì có thể sử dụng method
  + System.gc()
  + finalize() trong Object

Tìm hiểu thêm: <https://www.baeldung.com/java-finalize>

1. **Java structure**



* 1. **JVM**



*Biên dịch không phụ thuộc vào platform (do java compile ra byte code) 🡺 Java có thể chạy đa nền tảng*

Phase 1: gõ code vào file có đuôi .java trên eclipse (hoặc editor khác)

Phase 2: gõ lệnh javac name\_file.java trên cmd hoặc run program bằng eclipse, sau đó java sẽ compile ra file .class

Phase 3: Khi run program, Java sẽ load file .class lên ram và convert ra bytecode

Phase 4: Java sẽ check xem byteCode có hợp lệ hay không

Phase 5: dịch byteCode ra mã máy và chạy chương trình

Cách run 1 file trên cmd

B1: Mở cmd lên

B2: Gõ "cd *HelloWorldFolder*" // link tới folder chứa file java

B3: Gõ "javac *HelloWorld*.java" // Create name\_file.class

B4: Gõ "dir" // xem các file trong directory - xem đã tạo thành công file name\_file.class chưa

B5: Gõ "java HelloWorld" // run file

* 1. **Interpretion & Compilation**
     1. **Interpretion** (Thông dịch)
     + Lần lượt thi hành từng lệnh 1 thông qua chương trình
     + Mỗi lần cần chạy chương trình thì code mới được dịch để thực hiện từng lệnh 1 (line by line).
     + Thường dịch trước thành ngôn ngữ máy
     + Tất cả các ngôn ngữ không biên dịch ra mã máy điều phải sử dụng trình thông dịch, các ngôn ngữ theo trình thông dịch thường gọi là script (kịch bản).
     1. **Compilation** (Biên dịch)
* Dịch toàn bộ chương trình thông qua đối tượng compiler

VD: Tạo ra file thưc thi \*.exe(sẽ chạy 1 lần mà không cần thông dịch lại). Hoặc tạo ra file .class

* Code được biên dịch ra thành ngôn ngữ máy trên một hệ điều hành xác định và chỉ chạy trên hệ điều hành đó

VD: C++ chỉ chạy trên win, C++ không chạy trên Unix/Linux .

* + 1. **Java vừa là biên dịch vừa là thông dịch**
* Biên dịch

từ file java 🡪 class

* Thông dịch

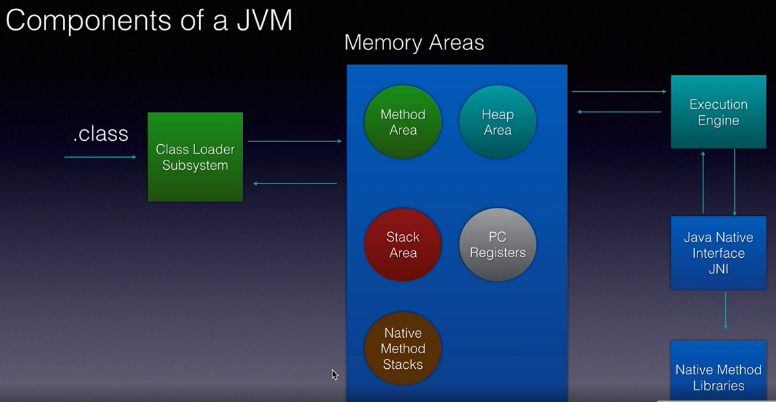
Lúc run file class 🡪 byteCode và chạy trên JVM

* 1. **Runtime & compileTime**

Compiletime là Build Project

Runtime là lúc Run Project

* 1. **Memory**



1. **Interview**

Xem các câu hỏi hay đi phỏng vấn trong file interview

* **Reference**