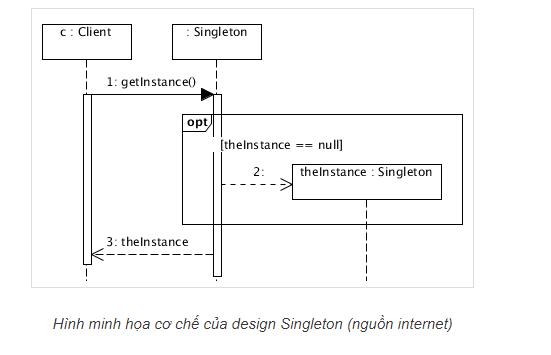
**Singleton**

* Trước hết mình sẽ khái quát về Singleton, Singleton thuộc loại Creational pattern (Java gồm 3 loại design pattern là Creational, Structural, Behavior). Do đó nó chức năng dùng để tạo object (đối tượng) kế thừa từ loại Creationa.
* Thiết kế Singleton đảm bảo cho class chỉ tạo được duy nhất một instance (instance là object trong vùng nhớ, ví dụ có 2 object cùng tham chiếu đến 1 vùng nhớ bằng cách clone thì ta hiểu rằng đó là một instance).

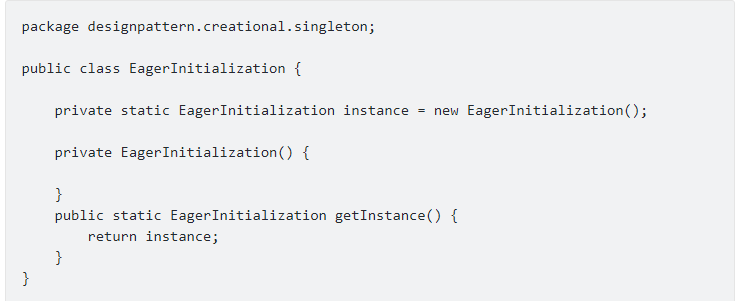


Singleton thường được sử dụng trong Logger, Configuration File, Cache, …Một số singleton class trong java java.lang.Runtime, java.awt.Desktop.

Một vài rule quan trọng:

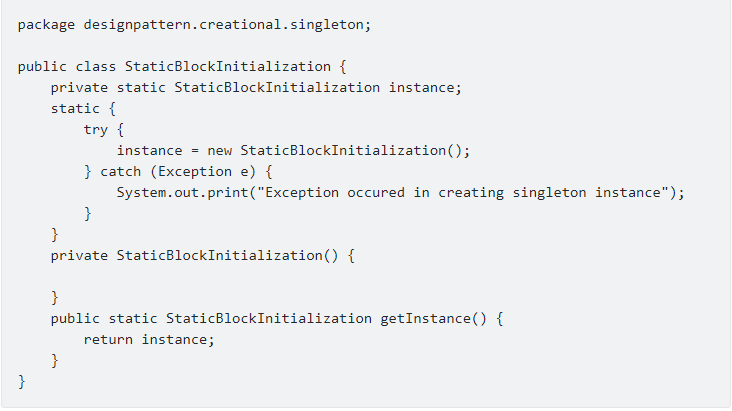
* Contructor phải để private.
* Sử dụng key word static.
* Có 1 public static method để các class bên ngoài có thể làm việc với singleton class.

**Cách thứ nhất: Eager initialization**



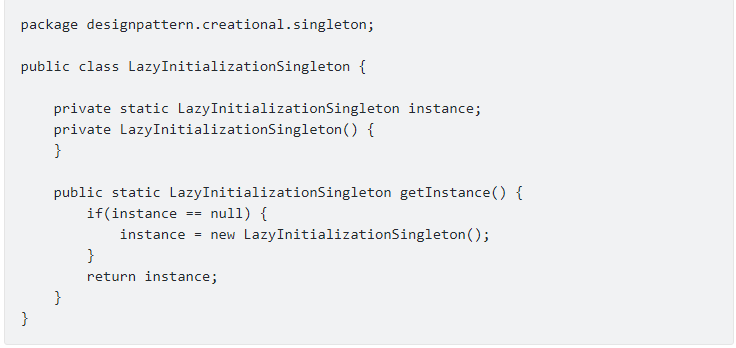
với cách này thì nếu class này **chưa được** hoặc **không sử dụng** thì object cũng đã được khởi tạo nên có thể gây lãng phí memory.

**Cách thứ hai: Dùng khổi static**



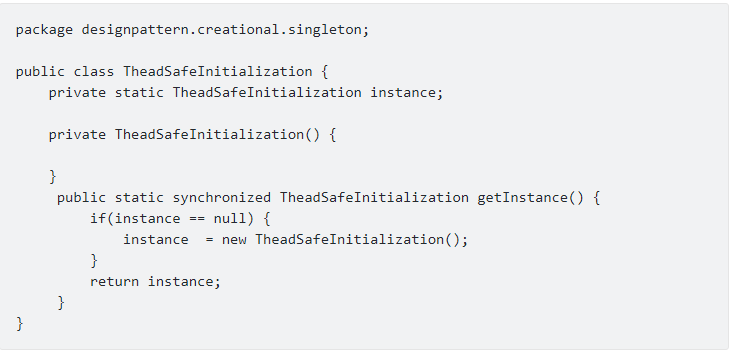
Object sẽ được khởi tạo khi class được load.

**Cách thứ 3: Lazy Initialization**



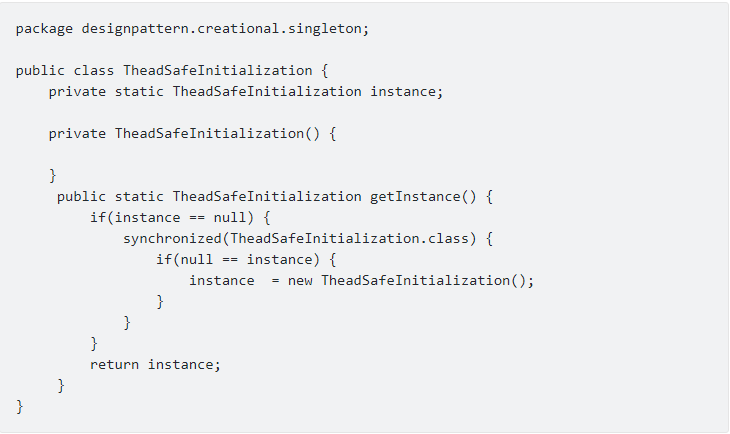
 Ở cách thực hiện này thì object sẽ đươc tạo ra khi getInstance() method được thực hiện lần đầu tiên.

**Cách thứ 4: Cách này dùng trong thread-safe**



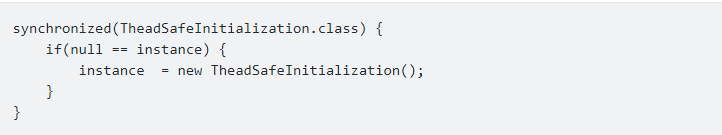
Với các cách khởi tạo thứ 3 ta rõ ràng thấy rằng nếu có nhiều nhiều thread cùng đồng thời gọi method getInstance thì sẽ có nhiều Singleton object được tạo ra và phá với code của chúng ta. Vậy nên để tránh trường hợp trên trong khi chạy nhiều thread thì ta thêm synchronized keyword vào getInstance() Các bạn có thể tìm hiểu về Đa luồng trong java.

**Cách thứ 5: Tối ưu cách 4.**



Có bạn nào thử đặt ra câu hỏi cách này tối ưu hơn cách 4 ở điểm nào? Xem xét code kĩ một chút ta thấy ở cách này mình không dùng synchronized ở getInstance() nữa mà mình dùng synchronized(){ }. Với chỉ một thay đổi nhỏ này sẽ có một sự thay đổi lớn trong khi chạy chương trình với lượng thread lớn.

Với cách 5 thì ta chỉ synchronized duy nhất một lần tại thời điểm gọi getInstance() lần đầu tiên, các gọi sau thì instance đã được khởi tao nên chương trình sẽ không thực hiện khối lệnh.



mà return lại giá trị của instance luôn.