Singleton Pattern

=====框架、無套例子=====

======Version Date：2019/09/12======

======小提醒：建議多看看不同例子(有發現錯的話，請告知 Leo。υ◉ω◉υ)。======

```
/** Singleton code重點
*   1. 未防止被別人任意[new]出來，所以將[建構子]設為[private](private EagerSingleton())。
*   2. 自己擁有且僅有一個自己的實例，所以宣告一個自己型態(private static EagerSingleton
*   instance)的實例出來。
*   3. 有一個[static]的方法(public static EagerSingleton getInstance())用來回傳自己的實例
*   給別人。
**/
/** Static
*   1.不會被物件(object)所擁有，而是被類別(class)所擁有。
*   2.具唯一的概念，被宣告static後，在執行時，將永遠且僅佔著一組記憶體位置。
*   3.可直接被呼叫、存取。
**/
/** synchronized(class_name.class) method
*   1.鎖定class，不管有多少個instance或多少個線程，在同一個時間只有一個線程能使用此class。
**/
```

/* Eager Initialization of Singleton */

```
public class EagerSingleton {
    private static EagerSingleton instance = new EagerSingleton();

    private EagerSingleton() {
        /* Can do something... */
    }

    public static EagerSingleton getInstance() {
        return instance;
    }

    /* Can do something... */
}
```

/* Lazy Initialization of Singleton-Double check */

```
public class LazySingleton {
    private static LazySingleton instance = null;

    private LazySingleton() {
        /* Can do something... */
    }

    public static LazySingleton getInstance() {
        if(instance == null ) { /* First Check */
            synchronized(LazySingleton.class) { /* synchronized method鎖定class */
                if(instance == null) { /* Second Check */
                    instance = new LazySingleton();
                }
            }
        }

        return instance;
    }

    /* Can do something... */
}

/** Lazy Singleton
*   1.在確認同一線程中尚未有該class的instance後，利用synchronized method來鎖定class，然後
*   再確認一次，此class是否有instance。
*   2.Lazy Singleton不一定要有用synchronized method才是 Lazy Singleton，只要能在第一次使
*   用Singleton class的時候才將class的instance new出來，就是Lazy Singleton。
**/
```