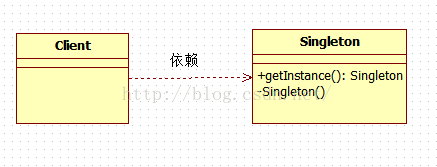
## 单例模式

1. 定义及使用场景：

定义：确保某一个类只有一个实例，而且自行实例化并向整个系统提供这个实例。

使用场景：避免产生多个对象消耗过多的资源，或者某种类型的对象应该有且只有一个；例如：访问IO和数据库等资源。

1. UML类图：



1）、构造函数不对外开放，一般为private

2）、通过一个静态方法或者枚举返回单例类对象

3）、确保单例类的对象有且只有一个，尤其是在多线程环境下

1. 实现方式：

1）、饿汉模式：在声明静态对象时就已经初始化单例实例。

2）、懒汉模式：先声明一个静态对象，并且在客户第一次调用getInstance时进行初始化。

Double CheckLock实现单例：既能够在需要时初始化单例，又能够保证线程安全

public class Singleton {  
  
 private static Singleton *sSingleton*;  
  
 private Singleton() {  
 }  
  
 public static Singleton getSingleton() {  
 if (*sSingleton* == null) {  
 synchronized (Singleton.class) {  
 if (*sSingleton* == null) {  
 *sSingleton* = new Singleton();  
 }  
 }  
 }  
 return *sSingleton*;  
 }  
}

3）、静态内部类单例模式：

public class Singleton1 {  
  
 private Singleton1() {  
 }  
  
 public static Singleton1 getInstance() {  
 return Singleton1Holder.*sSingleton1*;  
 }  
  
 private static class Singleton1Holder {  
 private static final Singleton1 *sSingleton1* = new Singleton1();  
 }  
}

4）、枚举单例：

public enum Singleton2 {  
 *INSTANCE*;  
  
 public void doSomething() {  
 Log.*i*("singleton2", "singleton2 do something...");  
 }  
}

5）、使用容器实现单例模式

1. 源码中的单例