SINGLETON CLASS WITH PUBLIC CONSTRUCTOR

The Singleton Design Pattern requires Constructors to be defined as private member of the class (Default constructor, Copy constructors and Overloaded Assignment operator should all be defined private in case of C++). This pattern is explained here..

If the programming language you are using does not allow you to define the constructor as private or protected (ex. Actionscript), then how will you implement the Singleton class?

There are few Object Oriented languages like Action script (Flex) where a constructor must be defined as public ?

```
1   /** Code in Actionscript (Flex) language.
2    */
3   public class Singleton
4   {
5       private function singleton{}  // ERROR. Constructor must be public
6   }
```

Solution:

If the Constructor cannot be made private, it is not possible to implement the Singleton Design Patern in full capacity.

But there are cases, when you want only one object of a class to be created. In such a cases, the definition of the class, should clearly state such intentions (even if it is not able to restrict the user completely from creating multiple objects via codding). At least the message should be conveyed clearly to the user who is trying to create an instance of the class, that not more than one object of this class should be created.

You should follow the following steps to indicate that your class is Singleton.:

1. Make a private static object inside the class to hold the only instance

```
1   /** FLEX CODE
2    */
3   public class
4   {
5       // Static Instance to hold the object
6       private static var _instance:Singleton = null;
7   }
```

The object is initialized to null.

When a request for the object is made for the very first time, a new object will be created and assigned to _instance, and in all further requests, this instance (_instance) will be returned. (Hence only one instance of this class will be present).

2. Make a public static function returning the only object of the class

```
1   /** FLEX CODE
2    */
3   public class
4   {
5       // Static Instance to hold the object
6       private static var _instance:Singleton = null;
7
8       public static function get instance():Singleton
9       {
10          if(_instance == null)
11              _instance = new Singleton(new SingletonEnforcer());
12
13          return _instance;
14      }
15  }
```

When the function instance will be called (don't bother about the get keyword before the function name, its an actionscript feature) for the first time it will call the constructor, create a new object and will assign it to _instance variable. In all further calls, the already created object (in _instance) will be returned.

3. Define the public constructor.

Since we cannot make the constructor private, lets create a public constructor.

```
1       /** FLEX CODE
2        */
3       public class
4       {
5            // Static Instance to hold the object
6            private static var _instance:Singleton = null;
7
8           // Public Constructor
9           public function Singleton()
10          {
11              ;
12          }
13
14          public static function get instance():Singleton
15          {
16              if(_instance == null)
17                  _instance = new Singleton(new SingletonEnforcer());
18
19              return _instance;
20          }
```

```
21      }
```

4. Try to restrict users to accidently create an object.

We will not be able to completely stop users from creating multiple objects of the Singleton class.

But for those obedient users who code according to the guidelines, we should write sufficient code to block creation of multiple objects of the class because of any side effect.

We can put two such checks.

4-(A) Throwing exception on multiple object creation

When user attempts to create more than one object, throw an Exception in the construtor.

```
1       // Public Constructor
2       public function Singleton()
3       {
4           if (_instance != null)
5           {
6               throw new Error("Singleton can only be accessed through Singleton.instance");
7           }
8       }
```

4-(A) Make constructor parameterized

Create a dummy class and change the Constructor to receive object of that class.

```
1    // Public Constructor
2    public function Singleton(obj:SingletonEnforcer)
3    {
4        if (_instance != null)
5        {
6            throw new Error("Singleton can only be accessed through Singleton.instance");
7        }
8    }
```

Where SingletonEnforcer is a dummy class defined in the same file

```
1    class SingletonEnforcer{}
```

Now, because there is no default constructor, hence object of the class Singleton can only be created by passing SingletomEnforcerclass object in the constructor

```
1    var obj:Singleton = new Singleton(new SingletonEnforcer());
```

It cannot be an accident.

Plus, The above code will also throw an exception if _instance is already created.

Final Code (In Flex):

```
1   package com.ritambhara.examples.design
2   {
3       public class Singleton
4       {
5           // ... REST OF THE CODE ...
6
7           // Static Instance to hold the object
8           private static var _instance:Singleton = null;
9
10          // Public Constructor
11          public function Singleton(obj:SingletonEnforcer)
12          {
13              if (_instance != null)
14              {
15                  throw new Error("Singleton can only be accessed through Singleton.instance");
16              }
17          }
18
19          public static function get instance():Singleton
20          {
21              if(_instance == null)
```

```
22                  _instance = new Singleton(new SingletonEnforcer());
23
24              return _instance;
25          }
26      }
27  }
28
29  class SingletonEnforcer{}
```

Source: http://www.ritambhara.in/singleton-class-with-public-constructor/