**Case Study for Singleton Design Pattern**

**Intent**

* Ensure a class has only one instance, and provide a global point of access to it.
* Encapsulated "just-in-time initialization" or "initialization on first use".

**Problem**

Application needs one, and only one, instance of an object. Additionally, lazy initialization and global access are necessary.

**Discussion**

Make the class of the single instance object responsible for creation, initialization, access, and enforcement. Declare the instance as a private static data member. Provide a public static member function that encapsulates all initialization code, and provides access to the instance.

The client calls the accessor function (using the class name and scope resolution operator) whenever a reference to the single instance is required.

Singleton should be considered only if all three of the following criteria are satisfied:

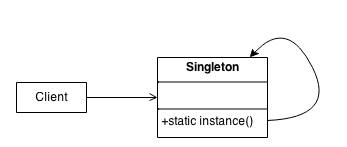
* **Ownership of the single instance cannot be reasonably assigned**
* **Lazy initialization is desirable**
* **Global access is not otherwise provided for**

If ownership of the single instance, when and how initialization occurs, and global access are not issues, Singleton is not sufficiently interesting.

The Singleton pattern can be extended to support access to an application-specific number of instances.

The "static member function accessor" approach will not support subclassing of the Singleton class. If subclassing is desired, refer to the discussion in the book.

**Structure**



Make the class of the single instance responsible for access and "initialization on first use". The single instance is a private static attribute. The accessor function is a public static method.



**Check list**

1. Define a private static attribute in the "single instance" class.
2. Define a public static accessor function in the class.
3. Do "lazy initialization" (creation on first use) in the accessor function.
4. Define all constructors to be protected or private.
5. Clients may only use the accessor function to manipulate the Singleton.