**Singleton**

A class of which only a single instance can exist.

Ensure a class has only one instance and provide a global point of access to it.

The singleton pattern is one of the best-known patterns in software engineering. Essentially, a singleton is a class, which only allows a single instance of itself to be created, and usually gives simple access to that instance.

**What is Singleton Design Pattern?**

1. Ensures a class has only one instance and provides a global point of access to it.
2. A singleton is a class that only allows a single instance of itself to be created, and usually gives simple access to that instance.
3. Most commonly, singletons don't allow any parameters to be specified when creating the instance, since a second request of an instance with a different parameter could be problematic! (If the same instance should be accessed for all requests with the same parameter then the factory pattern is more appropriate.)
4. There are various ways to implement the Singleton Pattern in C#. The following are the common characteristics of a Singleton Pattern.

* A single constructor, that is private and parameterless.
* The class is sealed.
* A static variable that holds a reference to the single created instance, if any.
* A public static means of getting the reference to the single created instance, creating one if necessary.

**Advantages**

The advantages of a Singleton Pattern are:

1. Singleton pattern can be implemented interfaces.
2. It can be also inherit from other classes.
3. It can be lazy loaded.
4. It has Static Initialization.
5. It can be extended into a factory pattern.
6. It helps to hide dependencies.
7. It provides a single point of access to a particular instance, so it is easy to maintain.

**Disadvantages**

The disadvantages of a Singleton Pattern are:

1. Unit testing is more difficult (because it introduces a global state into an application).
2. This pattern reduces the potential for parallelism within a program, because to access the singleton in a multi-threaded system, an object must be serialized (by locking).

**Singleton class vs. Static methods**

The following conpares Singleton class vs. Static methods:

1. A Static Class cannot be extended whereas a singleton class can be extended.
2. A Static Class can still have instances (unwanted instances) whereas a singleton class prevents it.
3. A Static Class cannot be initialized with a STATE (parameter), whereas a singleton class can be.
4. A Static class is loaded automatically by the CLR when the program or namespace containing the class is loaded.

**How to Implement Singleton Pattern in your code:**   
There are many way to implement a Singleton Pattern in C#.

1. No Thread Safe Singleton.
2. Thread-Safety Singleton.
3. Thread-Safety Singleton using Double-Check Locking.
4. Thread-Safe Singleton without using locks and no lazy instantiation.
5. Fully lazy instantiation.
6. Using .NET 4's Lazy<T> type.

**1. No Thread Safe Singleton:**

Explanation of the following code:

1. The following code is not thread-safe.
2. Two different threads could both have evaluated the test (if instance == null) and found it to be true, then both creates instances, which violates the singleton pattern.
3. Note that in fact the instance may already have been created before the expression is evaluated, but the memory model doesn't guarantee that the new value of instance will be seen by other threads unless suitable memory barriers have been passed.

Diagram:  
