### What is Singleton Pattern?

The Single Pattern is the simplest design pattern. Many times it’s important to have only one instance for a particular class, for example we need to provide configuration manager for all the configurations of our application. But, we need to restrict instantiation of configuration manager class to only one. So only one instance of configuration manager will serve all the requests for configuration of our application. To better understand the scenario following is pseudo code.

|  |
| --- |
| if(configurationManagerInstance == null)      configurationManagerInstance = new ConfigurationManager();  return configurationManagerInstance; |

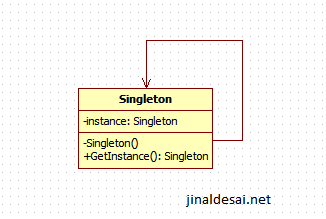
Whenever any request arrives that demands configuration settings of our application, one instance of configuration manager will serve the request. If this is the first request it will going to instantiate the configuration manager instance otherwise it will return already instantiated configuration manager instance.

### What Singleton Pattern Ensures?

Single Pattern ensures following things.

* It ensures that instance is created only once.
* It ensures that instance will never be null.
* It ensures that instance will be thread safe.
* Instance uses little memory.
* Provides lazy instantiation.
* Class is freezes for sub-classing and instantiation.

### Implementation of a Singleton Pattern

The implementation of a singleton pattern involves a static member inside “sealed” public singleton class with a private constructor and a static public method GetInstance() which returns a reference to the static member (which holds only single instance of the class).  
  
  
  
UML Diagram of Singleton Pattern