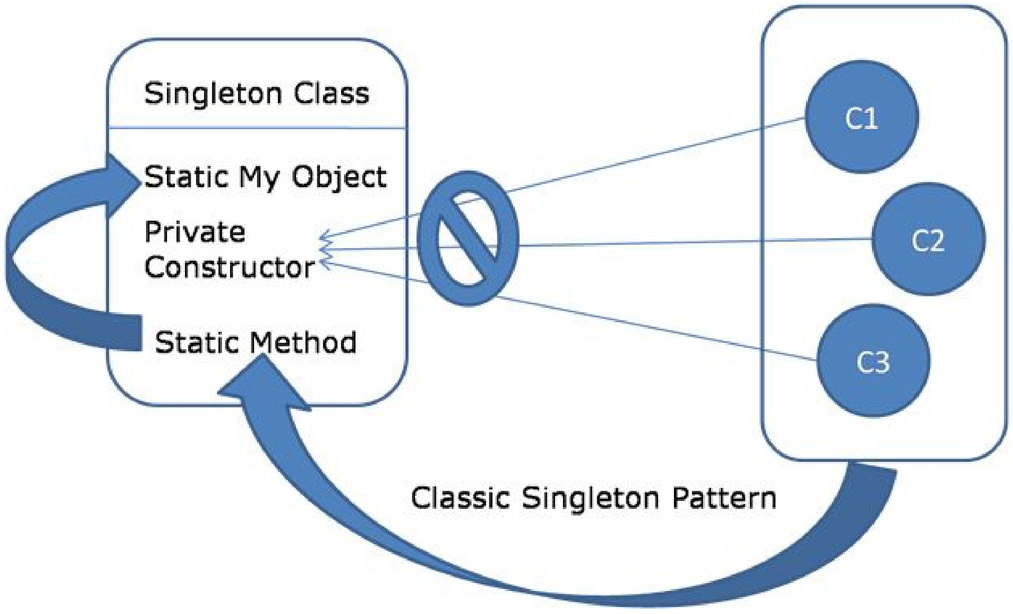
Singleton Design Pattern | Introduction

Singleton is a part of **Gang of Four design pattern** and it is categorized under **creational** design patterns. In this article, we are going to take a deeper look into the usage of the Singleton pattern. It is one of the most simple design pattern in terms of the modelling but on the other hand this is one of the most controversial pattern in terms of complexity of usage.

Singleton pattern is a design pattern which restricts a class to instantiate its multiple objects. It is nothing but a way of defining a class. Class is defined in such a way that only one instance of class is created in the complete execution of program or project. It is used where only a single instance of class is required to control the action throughout the execution. A singleton class shouldn't have multiple instances in any case and at any cost. Singleton classes are used for logging, driver objects, caching and thread pool, database connections.



Implementation of Singleton class

An implementation of singleton class should have following properties:

1. **It should have only one instance :** This is done by providing instance of class from within the class. Outer classes or subclasses should be prevented to create the instance. This is done by making the constructor private in java so that no class can access the constructor and hence cannot instantiate it.
2. **Instance should be globally accessible :** Instance of singleton class should be globally accessible so that each class can use it. In java it is done by making the access-specifier of instance public.

```
//A singleton class should have public visiblity
//so that complete application can use
public class GFG {

  //static instance of class globally accessible
  public static GFG instance = new GFG();
  private GFG() {
    // private constructor so that class
    //cannot be instantiated from outside
    //this class
  }
}
```

Run on IDE

Detailed Article: Implementation of Singleton Design Pattern in Java

Initialization Types of Singleton

Singleton class can be instantiated by two methods:

1. **Early initialization :** In this method, class is initialized whether it is to be used or not. Main advantage of this method is its simplicity. You initiate the class at the time of class loading. Its drawback is that class is always initialized whether it is being used or not.
2. **Lazy initialization :** In this method, class in initialized only when it is required. It can save you from instantiating the class when you don’t need it. Generally lazy initialization is used when we create a singleton class.

Examples of Singleton class

1. **java.lang.Runtime :** Java provides a class Runtime in its lang package which is singleton in nature. Every Java application has a single instance of class Runtime that allows the application to interface with the environment in which the application is running. The current runtime can be obtained from the getRuntime() method.
   An application cannot instantiate this class so multiple objects can’t be created for this class. Hence Runtime is a singleton class.
2. **java.awt.Desktop :** The Desktop class allows a Java application to launch associated applications registered on the native desktop to handle a URI or a file.
   Supported operations include:

- launching the user-default browser to show a specified URI;
  launching the user-default mail client with an optional mailto URI;
- launching a registered application to open, edit or print a specified file.
- This class provides methods corresponding to these operations. The methods look for the associated application registered on the current platform, and launch it to handle a URI or file. If there is no associated application or the associated application fails to be launched, an exception is thrown.
- Each operation is an action type represented by the Desktop.Action class.

This class also cannot be instantiated from application. Hence it is also a singleton class.

Applications of Singleton classes

There is a lot of applications of singleton pattern like cache-memory, database connection, drivers, logging. Some major of them are :-

1. **Hardware interface access:** The use of singleton depends on the requirements. Singleton classes are also used to prevent concurrent access of class. Practically singleton can be used in case external hardware resource usage limitation required e.g. Hardware printers where the print spooler can be made a singleton to avoid multiple concurrent accesses and creating deadlock.
2. **Logger :** Singleton classes are used in log file generations. Log files are created by logger class object. Suppose an application where the logging utility has to produce one log file based on the messages received from the users. If there is multiple client application using this logging utility class they might create multiple instances of this class and it can potentially cause issues during concurrent access to the same logger file. We can use the logger utility class as a singleton and provide a global point of reference, so that each user can use this utility and no 2 users access it at same time.
3. **Configuration File:** This is another potential candidate for Singleton pattern because this has a performance benefit as it prevents multiple users to repeatedly access and read the configuration file or properties file. It creates a single instance of the configuration file which can be accessed by multiple calls concurrently as it will provide static config data loaded into in-memory objects. The application only reads from the configuration file at the first time and there after from second call onwards the client applications read the data from in-memory objects.
4. **Cache:** We can use the cache as a singleton object as it can have a global point of reference and for all future calls to the cache object the client application will use the in-memory object.

Important points

- Singleton classes can have only one instance and that instance should be globally accessible.
- java.lang.Runtime and java.awt.Desktop are 2 singleton classes provided by JVM.
- Singleton Design pattern is a type of creational design pattern.
- Outer classes should be prevented to create instance of singleton class.