Singlton Design Pattern in Deep

Traditional Methods of Making Singletons

There are several popular methods for making singletons.

3. and make it thread safe with synchronized and voletile (read from main memory not cpu cache ) :

ایراد روش دوم اینکه اگه همزمان دو تا thread همزمان بیان داخل و بخوان ابجکت ازشون ساخته بشه حتی با وجود کسری از ثانیه ، ازشون دوتا ابجکت مستقل با هش کد های متفاوت از نخ ها ایجاد میکنه چراکه اصل دیزاین پترن سینگلتون را زیر سوال میبره و دیگه thread های مختلف با هش کدهای یکسان نداریم در نتیجه این ThreadSafe نیست پس این روش قابل قبول نیس

برای حل این مشکل در حالت سوم میایم و از مکانیسم و متدهایی که جاوا در اختیار مون گذاشته استفاده میکنیم یکی از این مکانیزم ها استفاده از voletile است **که میاد چیکارمیکنه؟؟ ببینید هرمتدی که ساخته میشه میاد روی سی پی یو یک کش براش ایجاد میشه در این حالت** voletile میگه که نیا از cpu cache متد را بخون بیا و از main memory مقدار متد ایجاد شده را بخون (منظور متد getInstance())

public class Singleton {  
  
 private static voltile Singleton INSTANCE = null;  
  
 private Singleton() {}  
  
 public static Singleton getInstance() {  
 if (INSTANCE == null) {  
 synchronized (Singleton.class) {  
 if (INSTANCE == null) {  
 INSTANCE = new Singleton();  
 }  
 }  
 }  
 return INSTANCE;  
 }  
  
}

**تا همیشه مقداری را که میخونی تازه باشه و در ادامه با مکانیزم synchronized بیا و اونو همزمان کن و نخ ها را با هش کد مشابهه ایجادش کن تا مشکل مرحله دو حل بشه ؛**

**بعبارتی میاد با ورود نخ اول به متد ، یک ابجکت ازش میسازه و اجازه نمیده همزمان با اون نخ شماره دو هم ورود کنه اولی که کارش تموم شد دومی بعد اولی ورود پیدا میکنه حالا دراین حالت چک میکنه اگه نمونه ای از آبجکت ساخته شده بود دیگه برای بار دوم مقدار جدید نمیسازه new نمیکنه بلکه برای نخ دومی هم همون ابجکت نخ دوم درنظر میگیره پس تا اینجا مشکل حل میشه.و اصطلاحا اومد و** ThreadSafe اش کرد اما بازم ایراد داره ...

**اما باز هم یه مشکل وجود داره اینکه میگه تو اومدی و فقط جلو یه مشکل را که new object کردنه را گرفتی اما ما میتونیم به روشهای مختلف یک کلاس را داشته باشیم و ازش استفاده کنیم بعبارتی از حالت های دیگه کلاس استفاده کنیم مثل موارد زیر که دانستن اون خیلی مهمه اینا مارو به برنامه نویس حرفه ای تبدیل میکنه::**

1. Serialization and deserialization
2. Reflection
3. Clone

**این سه خاصیت و حالت ممکن که حتی اگه جلو new کردن این کلاس هم بگیریم بازم ما امکان داشتن ابجکت های جدید از کلاس مدنظرمون را داریم و باز در این شرایط شرط داشتن دیزاین پترن سینگلتون و داشتن ابجکت های مختلف با یک هش کد ثابت در کل برنامه بهم میخوره برای حل این مسئله به سراغ حالت چهارم میریم**

Static fields are initialized at class loading time. Therefore, in Method 1 , singleton instances are created even in a case we don’t use them at runtime. This is not a problem as long as the singleton object is not too big and creating the instance is not too expensive. Method 2 avoids this problem with lazy initialization. In Method 3, the instance is created when we access the singleton object for the first time. Fine-grained synchronization is used to ensure that no more than one object is created with multiple concurrent threads .

All of the above methods work fine until you are not doing serialization and deserialization with a singleton class. Let’s think again: How did we achieve the singleton behavior in above methods? It was done by making the constructor private and making the constructor inaccessible for creating new instances of the class. But isn’t there any other ways to create an instance of a class other than the constructor? The answer isn’t no. There are some other advanced methods.

1. Serialization and deserialization
2. Reflection
3. Clone