Volatile Variablen

1. Vor jeder Auswertung der Variable (Laden) wird ihr aktueller Wert aus dem Hauptspeicher in den lokalen Thread Speicher geholt (gelesen).
2. Nach jeder Zuweisung (Speichern) wird ihr aktueller Wert aus dem lokalen Thread Speicher in den Hauptspeicher geschrieben.
3. Volatile Variablen werden in genau der Reihenfolge geschrieben oder gelesen wie vom Thread angefordert.
4. Die Lese- und Schreibzugriffe sind immer atomar (auch bei double und long -64 bit Werten- bei denen das sonst nicht so ist).

Singleton - unsichere Initialisierung

```
Diese Implementierung hat folgende Besonderheiten:
[1] Theore tisch funkti oniert sie nicht - in der Praxis taugt sie aber als Signleton (siehe dazu folgende Diskussion)
[2] Man darf sich hier bei Zugriff nicht auf ein korrekt initia lis iertes Signleton verlassen, da im konkreten Fall auf Grund des Memory Modelles keine Annahmen darüber zugelassen sind wann die Felder des erzeugten Singleton wirklich alle korrekt befüllt und für jeden Thread konsistent lesbar sind.
public class Resource{
  private static Resource singleton;
public static Resource create Ins tan ce(){
      if( sin gle ton ==n ull){ //erster Test
        syn chr oni zed (Re sou rce.cl ass){
            if( sin gle ton ==n ull){ //zweiter Test
                  sin gle ton=new Resour ce();
           } //endif
         } // end sync
     }// endif
  return singleton;
 }
}
```

Details zum Problem unter: http://www.cs.umd.edu/~pugh/java/memoryModel/DoubleCheckedLocking.html

By **Huluvu424242**
(FunThomas424242)

cheatography.com/funthomas424242/
github.com/Huluvu424242

Not published yet.
Last updated 11th May, 2016.

Singleton - korrekte Intialisierung

```
private static volatile Resource resource;
private static volatile Resource singleton;
public static Resource create Ins tan ce(){
  if( res our ce= =null){
    syn chr oni zed (Re sou rce.cl ass){
      if( sin gle ton ==n ull){
              sin gle ton=new Resour ce();
     }
   }
    res our ce= sin gleton;
 }
  return resource;
}
```

Ähnliche Lösungen werden auch hier diskutiert: http://www.angelikalanger.com/Articles/EffectiveJava/41.JMM-DoubleCheck/41.JMM-DoubleCheck.html

By **Huluvu424242**
(FunThomas424242)

cheatography.com/funthomas424242/
github.com/Huluvu424242

Not published yet.
Last updated 11th May, 2016.