

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] Theoretisch funktioniert sie nicht - in der Praxis taugt sie aber als Singleton (siehe dazu folgende Diskussion)

[2] Man darf sich hier bei Zugriff nicht auf ein korrekt initialisiertes Singleton verlassen, da im konkreten Fall auf Grund des Memory Modells 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 Instance(){
        if( singleton == null){ //erster Test
            synchronized (Resource.class){
                if( singleton == null){ //zweiter Test
                    singleton=new Resource();
                } //endif
            } // end sync
        } // endif
        return singleton;
    }
}
```

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



By **Huluhu424242**
(FunThomas424242)

Not published yet.
Last updated 11th May, 2016.
Page 1 of 2.

Sponsored by **Readable.com**
Measure your website readability!
<https://readable.com>

cheatography.com/funthomas424242/
github.com/Huluhu424242

Singleton - korrekte Initialisierung

```
private static volatile Resource resource;  
private static volatile Resource singleton;  
public static Resource create Instance(){  
    if (resource == null){  
        synchronized (Resource.class){  
            if (singleton == null){  
                singleton = new Resource();  
            }  
        }  
        resource = singleton;  
    }  
    return resource;  
}
```

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



By **Huluvu424242**
(FunThomas424242)

Not published yet.
Last updated 11th May, 2016.
Page 2 of 2.

Sponsored by **Readable.com**
Measure your website readability!
<https://readable.com>

cheatography.com/funthomas424242/
github.com/Huluvu424242