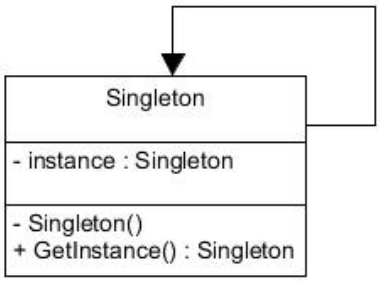
Singleton Design Pattern

Kurzbeschreibung

Ein Singleton Pattern wird dann gebraucht, wenn von einer Klasse nur eine Instanz erstellt werden sollte und diese einzige Instanziierung global über das ganze Programm benötigt wird.

UML-Diagramm



Ausführliche Beschreibung

Ein Konstruktor einer Singleton Klasse muss immer privat sein, denn wenn der Konstruktor privat ist kann man ausserhalb der Klasse keine Instanzen erstellen. Eine solche Klasse darf nur einmal instanziiert werden, dies geschieht beim ersten Aufruf der getInstance Funktion/Property. Nachdem man Sie einmal benutzt hat, kann man davon keine weiteren Instanzen erstellen. Wie man im unterem Codebeispiel erkennen kann, kann man keine weitere Instanz erstellen, wenn es schon eine gibt. Dafür muss jedoch der Verweis auf das instanziierte Objekt und die «getInstance» Funktion/Property statisch sein. Das grösste Problem von diesem Design Pattern, ist das Löschen der Instanz, dazu gibt es noch ein weiteres Problem, denn wenn man «multithreading» nutzt kann es vorkommen, dass von der Klasse weitere Instanzen erstellt werden, dafür gibt es jedoch schon eine einfache Lösung. Ein Vorteil von so einem Pattern ist, dass man in einem Singleton Pattern Interfaces oder Abstrakte Klassen implementieren kann, dies gilt als grosser Vorteil gegenüber einer statischen Klasse, zudem kann man ein Singleton wie ein Parameter einer Methode übergeben.

Codebeispiel – Java

```java
public final class Singleton {
    private static volatile Singleton instance;

    private Singleton() {}

    public static Singleton getInstance() {
        if (instance == null) {
            instance = new Singleton();
        }
        return instance;
    }
}
```