MODELLIERUNG

Singleton

Autor: Johannes Theiner
Semester: Sommersemester 2018
letzte Änderung: 20. Januar 2022

Inhaltsverzeichnis

1 Beschreibung

Das Singleton-Muster (Einzelstück) gehört zur Gruppe der Erzeugungsmuster und stellt sicher das ein Objekt nur einmal im Speicher vorhanden ist. Dazu wird der Zugriff auf den Konstruktor so eingestellt das nur die eigene Klasse Objekte erstellen kann. Die Klasse selbst enthält nur eine Klassenmethode die das Objekt erstellt und in einer Klassenvariable speichert um bei späteren Aufrufen dieses zurückzugeben.

2 Anwendungsfälle

2.1 Zugriff auf zentrale Ressourcen

In vielen System stehen einige Ressourcen nur begrenzt zur Verfügung (Drucker, Scanner, Plotter). Das Singleton-Muster ermöglicht es den Zugriff auf diese Ressourcen zu verwalten und zu garantieren das keine Anfragen gleichzeitig bearbeitet werden.

2.2 Warteschlangen

Während die Abarbeitung einzelner Aufgaben auf mehrere Threads oder Computer aufgeteilt werden kann ist es nötig die Verteilung dieser Aufgaben nur von einer Instanz durchführen zu lassen.

2.3 fachliche Serialisierung

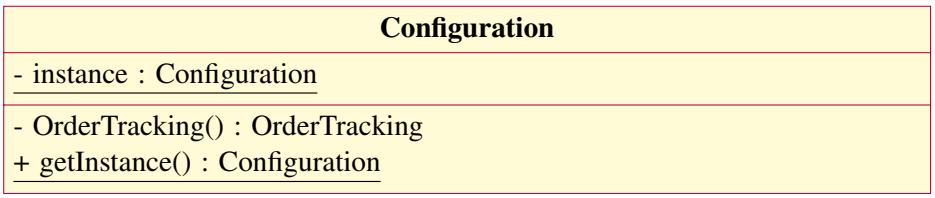
Bei einem System welches mehrere eindeutige Identifizierer gleichzeitig erstellen soll, etwa ein System zur Rechnungserstellung, ist es sinnvoll dies als Singleton zu implementieren um sicherzustellen das zwei gleichzeitig anfragende Clients nicht den selben Identifizierer erhalten.

2.4 Klassen mit großem Instanzierungs Aufwand

Auch bei einer Klasse welche viel Zeit benötigt um initialisiert zu werden macht es Sinn diese erst zu initialisieren wenn die benötigt wird und danach nur noch wiederzuverwenden.

3 Implementation

3.1 als UML Klassendiagramm



3.2 in Java

4 Alternativen

4.1 Globale Variablen

Häufig werden anstatt Singletons der Einfachheit halber globale Variablen verwendet, was mit einigen Nachteilen verbunden ist.
Während auf globale Variablen von überall zugegriffen werden kann, kann der Zugriff auf Singletons kontrolliert werden.
Globale Variablen werden außerdem bereits beim Start des Programms initialisiert was unter Umständen zu langen Startzeiten führen kann, während es bei Singletons möglich ist die Initialisierung erst auszuführen wenn die Klasse benötigt wird.

4.2 Statische Klassen

Manchmal wird auch die gesamte Klasse zu einer statischen Klasse, was allerdings bei der Implementierung einschränkt. So kann eine Singleton-Klasse ihren eigenen Zustand einfacher in privaten Variablen speichern und nichtstatische Methoden anbieten.

4.3 Vererbung

Es ist auch möglich das Singleton Muster mit Vererbung zu verbinden, wenn der Konstruktor nicht auf *private* sondern auf *protected* gesetzt wird. im Regelfall erwartet der Anwender der Singleton-Klasse dies aber nicht.