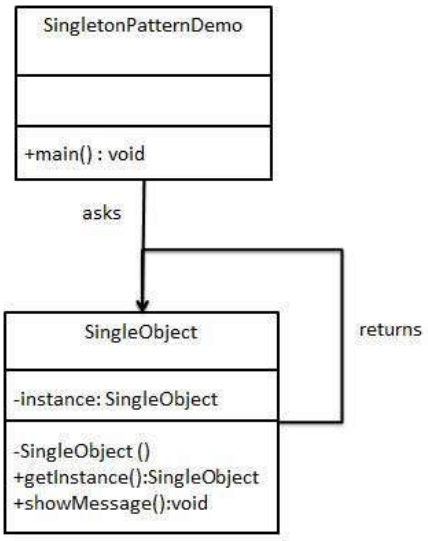
Singleton Pattern

Problemstellung

Der Zugriff auf eine Klasse soll nur über genau ein einziges Objekt möglich sein. Des weiteren wird dabei die sogenannte Lazy initialization angewendet, was bedeutet, dass das Objekt erst instantiiert wird, wenn die GetInstance-Methode zum ersten Mal aufgerufen wird.

Beschreibung

Das Singleton Design Pattern wird wie folgt implementiert:



Wie im Klassendiagramm zu sehen ist, umfasst dieses Entwurfsmuster lediglich eine Klasse. Diese hat einen privaten Konstruktor, eine Klassenvariable welche eine Instanz von sich selbst Speichert und eine statische Methode, welche die Instanz des Objekts zurück gibt. Dadurch kann immer nur eine Instanz dieser Klasse existieren. Über diese werden dann die benötigten Funktionen aufgerufen. Die GetInstance-Methode liefert das bereits erstellte Objekt der Klasse zurück, falls bereits eines vorhanden ist, anderenfalls wird eine neue Instanz erstellt. Das bringt den Vorteil mit sich, dass die Objekte der Klasse immer nur dann erstellt werden, wenn diese wirklich gebraucht werden. Dadurch wird verhindert, dass der Speicher mit unnötigen Objekten belegt wird.

Anwendung

Dieses Entwurfsmuster benutze ich selbst sehr oft. Ich arbeite auf häufig an Datenbanken. Für den Zugriff darauf haben meine Vorgänger eine Klasse geschrieben, welche die Instanzen auf alle Recordsets speichert und noch einige andere Funktionen implementiert hat. Durch die Anwendung des

Singleton Patterns wird garantiert, dass kein durcheinander mit den offenen Verbindungen mit der Datenbank entsteht, da immer nur eine aktiv ist. Des weiteren verhindert es, dass mehrere Recordsets auf der gleichen Tabelle offen sind und dadurch Inkonsistenzen entstehen.

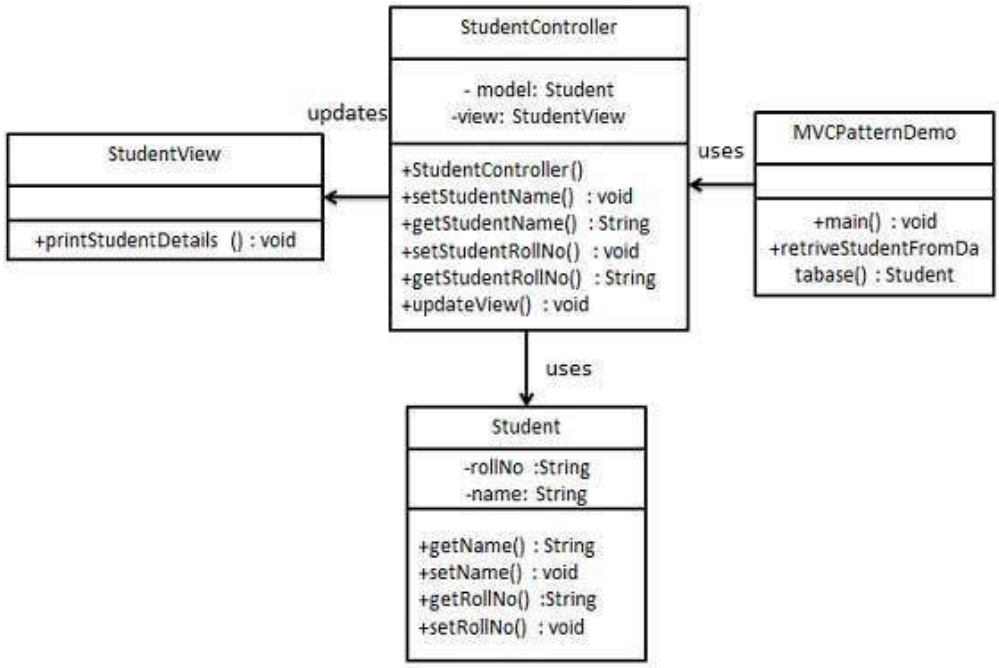
MVC Pattern

Problemstellung

Das MVC Pattern kommt immer dann zum Einsatz, wenn der Entwickler die Daten von der Logik und den Daten trennen will. Wird keine solche Trennung vorgenommen, erschwert das die Entwicklung. Schlimmstenfalls führt das zu write-only Code, welcher das Debuggen stark erschwert, da keine klare Aufgabenteilung der Klassen vorhanden ist.

Beschreibung

MVC steht für Model View Controller und trennt die Logik, die Daten und die Darstellung voneinander. Die Implementierung sieht wie folgt aus:



Die drei verschiedenen Komponenten sind für unterschiedliche Aufgaben verantwortlich:

- **Model**: Im Model sind die Daten gespeichert, welche dann über die View dargestellt werden. Es kann aber auch Logik enthalten, um den Controller über Änderungen zu informieren.
- **View**: Die View ist für die Darstellung verantwortlich. Sie bekommt vom Controller die Daten aus dem Model, um diese dann dem Benutzer zu präsentieren.

- **Controller**: Der interagiert sowohl mit der View als auch mit dem Model. Er ist für die ganze Logik, wie zum Beispiel das Eventhandling zuständig.

Anwendung

Ich persönlich habe das MVC-Pattern im Modul 151 zum ersten mal eingesetzt. Da ich der Meinung war, dass das von der gibb bereitgestellte nicht brauchbar war, habe ich mir selbst ein kleines MVC-Framework geschrieben. Hätte ich auf den Einsatz eines MVC-Frameworks verzichtet, wäre der Code der Applikation sicherlich sehr unlesbar geworden. Durch das brauchen des MVC Patterns wird auch Redundanzen im Programmcode vorgebeugt.