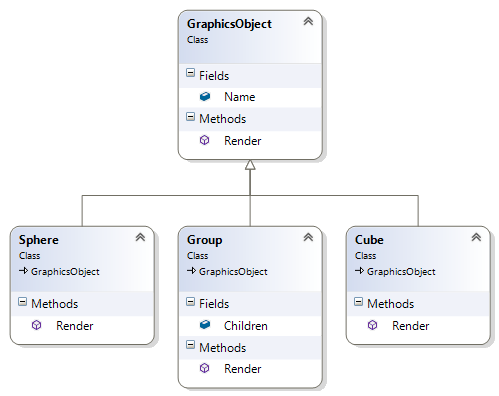
Einfacher Szenengraph:

GraphicsObject vererbt an Group, Cube und Sphere

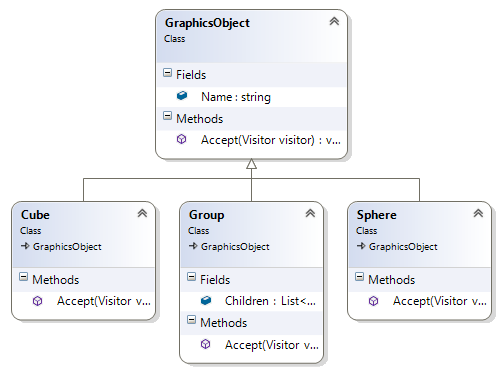


Render-Methode traversiert UND rendert

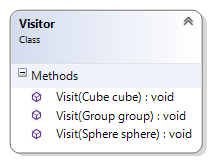
Problem: Traversierung auch für‘s Auf Platte schreiben (Picking, nach Namen Suchen…).

Lösung 1: Methode WriteToFile mit anderem Inhalt

Problem 2: Neue Traversierungsgründe erfordern Änderung an allen bestehenden Node-Typen. Klassen blähen sich auf.

Lösung 2: Visitor-Pattern. Alle GraphicsObject erhalten statt vielen funktionsbezogenen Render/SaveToFile/Pick/Find-Metoden eine einzige allgemeine Accept(Visitor visitor) Methode:

Zudem wird eine allgemeingültiger Visitor-Klasse erstellt, die für jeden GraphicsObject Typ eine eigene Vist-Methode enthält:

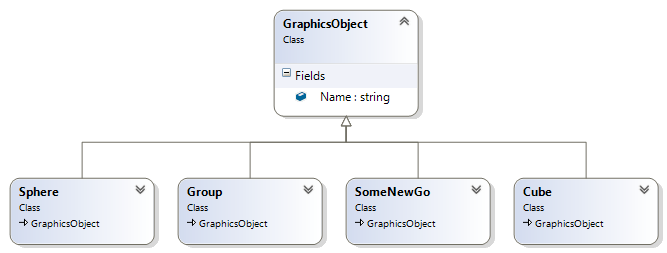


Um für eine bestimmte Funktionalität eine Traversierung durchzufühern, die polymorph für unterschiedliche Node-Typen unterschiedliche Visit-Methoden vorsieht, kann eine eigene, neue Visitor-Klasse erzeugt werden, ohne dass an den GraphicsObject-Klassen Änderungen vorgenommen werden müssen. Diese Klasse erbt von Visitor und lässt sich damit in der Accept-Methode übergeben.

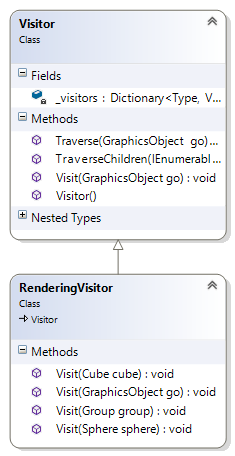
Dieses Verfahren nennt man auch „Double-Dispatch“: Mit virtuellen Methoden kann abhängig von EINEM Typ ein Methodenaufruf generiert werden. Mit dem hier gezeigten Visitor-Pattern ist die Auswahl der richtigen Visit-Methode aber von zwei Typen abhängig: Vom aktuellen Visitor UND vom aktuellen Node-Typen.

Voraussetzung ist, dass die Node-Typen das Visitor-/Double-Dispatch-Pattern unterstützen, indem sie die Accept-Methode implementieren. Dabei müssen Container-Objekte die Traversierung unterstützen.

Problem 3: Auf Objektgraphen-Klassen, über deren Source-Code man nicht verfügt, oder die anderweitig nicht geändert werden können soll ein Visitor-/Double-Dispatch-Pattern implementiert werden.



Lösung 3a: Visitor, der ohne Mitarbeit der „visitables“ auskommt. Visitor übernimmt die Traversierung selbst. Beim Visit wird per „as“ Typecast beim Traversieren der konkrete GraphicsObject-Typ überprüft und somit der GraphicsObject-Dispatch durchgeführt. Nachteil: if-Kette bei der Traversierung. Außerdem wenig flexibel, da bei neu hinzukommenden GraphicsObject-Klassen immer der Traversierungs-Grundtyp geändert werden muss.



Lösung 3b: Visitor hält sich ein Lookup der verschiedenen Visit-Methoden, die per Argument-Typ (konkreter GraphicsObject-Typ) indiziert wird und kann somit den Dispatch durchführen.