Aufgabenblatt 5

**Aufgabe 5.5**

**Erklären Sie den Effekt des Schlüsselworts override im Kontext der Vererbung!**

Das Wort „override“ signalisiert dem Compiler, dass eine Methode einer abgeleiteten Klasse, die Methode der Basisklasse mit gleicher Signatur überschreiben soll, d.h. dass ihre Funktionsweise an die Eigenschaften der abgeleiteten Klasse angepasst werden und in der abgeleiteten Klasse neu definiert werden müssen. Gleichzeitig überprüft das Wort aber auch, ob die entsprechende Methode der Basisklasse mit dem Schlüsselwort „virtual“ deklariert wurde und es dementsprechend überhaupt eine Methode zum Überschreiben gibt.

**Was passiert, wenn Sie das Schlüsselwort weglassen?**

Wäre die entsprechende Methode in der Basisklasse eine rein virtuelle Methode, könnte kein Objekt der abgeleiteten Klasse instanziiert werden, da es nur ein

Würde das Wort weggelassen werden und wäre die entsprechende Methode in der Basisklasse mit dem Schlüsselwort „virtual“ als virtuelle Methode deklariert, würde stattdessen in der abgeleiteten Klasse eine neue Methode mit gleichem Namen angelegt werden. Gleiches gilt für den Fall, dass die Methode in der Basisklasse nicht als virtual deklariert ist.

**Aufgabe 5.7**

**Sehen Sie sich folgenden Beispielcode an:**

Color red {255 , 0, 0};

glm :: vec3 position {0.0f, 0.0f, 0.0f};

std :: shared\_ptr <Sphere > s1 =

std :: make\_shared <Sphere >( position , 1.2f, red , " sphere0 ");

std :: shared\_ptr <Shape > s2 =

std :: make\_shared <Sphere >( position , 1.2f, red , " sphere1 ");

s1 -> print (std :: cout );

s2 -> print (std :: cout );

**Erklären Sie anhand des Beispiels die Begriffe „Statischer Typ einer Variablen“ und „Dynamischer Typ einer Variablen“.**

Der Statische Typ einer Variablen ist der Typ des Werts der Variable, der zur Compilierzeit festgelegt wird. Der dynamische Typ einer Variablen ist der Typ, den der Wert der Variablen während der Laufzeit tatsächlich annimmt. Für gewöhnlich sind statischer und dynamischer Typ in C++ identisch, nur im Kontext der Vererbung können sie sich unterscheiden, wenn bspw. ein Pointer des Typs der Basisklasse auf ein Objekt des Typs der abgeleiteten Klasse zeigt. In diesem Fall ist statische Typ der Pointervariablen der Typ der Basisklasse, während ihr dynamischer Typ der Typ der abgeleiteten Klasse ist.

**Wann wird welche Art des Typs überprüft? Was sind die dynamischen und die statischen Typen der Variablen s1 bzw s2?**

Der statische Typ einer Variablen wird zur Compilierzeit ermittelt, der genaue dynamische Typ erst während der Laufzeit.

S1: statischer Typ: Sphere // dynamischer Typ: Sphere

S2: statischer Typ: Shape // dynamischer Typ: Shape

**Aufgabe 5.8**

**Geben Sie im Funktionsrumpf der Kon- und Destruktoren der Klassenhierarchie deren Aufruf auf der Konsole aus. Kompilieren Sie den gegebenen Programmcode. In welcher Reihenfolge werden Konstruktoren und Destruktoren aufgerufen?**

Bei der Konstruktion der Objekte wird zunächst der Konstruktor der Basisklasse und anschließend der Konstruktor der abgeleiteten Klasse aufgerufen. Bei der Destruktion ist es genau umgekehrt: Zunächst wird der Destruktor der abgeleiteten Klasse aufgerufen, dann der Destruktor der Basisklasse.

**Entfernen Sie nun das Schlüsselwort virtual vom Destruktor der Basisklasse, testen Sie erneut und erklären Sie den Unterschied.**

Da der Destruktor nun nicht mehr überschrieben wird, wird für das Objekt s2 nur der Destruktor von Shape aufgerufen, statt wie zuvor wie bei s1, erst der Destruktor von Sphere und innerhalb von diesem der Destruktor von Shape.

Dies geschieht, weil der Destruktor durch das Fehlen des Schlüsselwortes virtual nun nicht mehr die Funktionsweise für die einzelnen abgeleiteten Klassen konkretisiert, sondern weil er nun überladen ist. Die Überladung wird zur Übersetzungszeit aufgelöst. Dafür wird von jeder Variable der statische Typ als Referenzpunkt genommen. Da die Klasse Shape eine eigene Überladung der Methode „Destruktor“ hat, wird diese für das Objekt s2 verwendet, da sein statischer Typ „Shape-Pointer“ ist. Wäre das Schlüsselwort virtual noch gegeben, könnte die Konkretisierung während der Laufzeit zu tragen kommen und für s2 würde der Destruktor seines dynamischen Typs aufgerufen werden, nämlich Sphere.

**Aufgabe 5.9**

**Erklären Sie die Unterschiede zwischen Klassenhierarchie vs. Objekthierarchie - Klassendiagramm vs. Objektdiagramm.**

Die Klassenhierarchie beschreibt den Aufbau verschiedener Klassen, die aufeinander aufbauen. Dabei gibt es abgeleitete Klasse(n) von mehreren oder einer Basisklasse(n), die die Verhaltensweisen und Eigenschaften der Basisklasse konkretisieren und ausbauen können und so für speziellere Anwendungsfälle verwendet werden können. Die Klassenhierarchie beschreibe also die verschiedenen Klassen und ihre Beziehung untereinander.

Die Objekthierarchie dagegen beschreibt die Beziehung der zu einem bestimmten Zeitpunkt mithilfe von Klassen erzeugten Objekte untereinander. Diese können sich von den Beziehungen ihrer Ausgangsklassen unterscheiden, so können bspw. Objekte miteinander interagieren, deren Klassen in keinerlei Beziehung zueinander stehen.

Ein Klassendiagram visualisiert den Aufbau der Klassenhierarchie und zeigt bspw. in UML welche Klassen miteinander in Beziehung stehen und welche Klassen abgeleitete Klassen anderer Klassen sind. Sie können bspw. aber keine Beziehung zweier Instanzen der gleichen Klasse untereinander modellieren.

In diesem Fall wäre dem ein Objektdiagramm überlegen, welches eine Beziehung zwischen diesen beiden Objekten darstellen könnte. Es modelliert ein System zu einem konkreten Zeitpunkt und stellt so die Objekthierarchie dar.