5.5

- override bedeuted, dass eine virtuelle Methode der Basisklasse in einer von dieser abgeleiteten Klasse mit der gleichen Signatur überschrieben wird und daraufhin der compiler die methode der abgeleiteten klasse aufruft

- comiler übeprüft dadurch außerdem, ob überhaupt eine virtuelle methode in der basisklasse existiert (selbstkontrolle) die überschrieben werden kann

- vergisst man override + virtual und instanziiert ein abgeleitetes objekt der basisklasse, dann wird die methode der basisklasse und nicht die methode des abgeleiteten objekts aufgerufen

5.7

Die statische Klasse einer Variablen ist die der Deklaration – bei s1 Sphere, bei s2 Shape

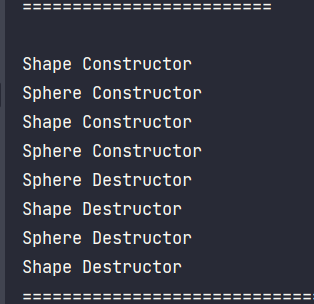
die dynamische Klasse ist die des aktuellen Wertes – bei s1/s2 Sphere

Statischer Typ wird überprüft zur Übersetzungszeit

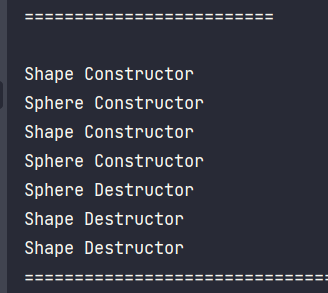
Dynamischer Typ wird überprüft zur Laufzeit

5.8

- mit virtual

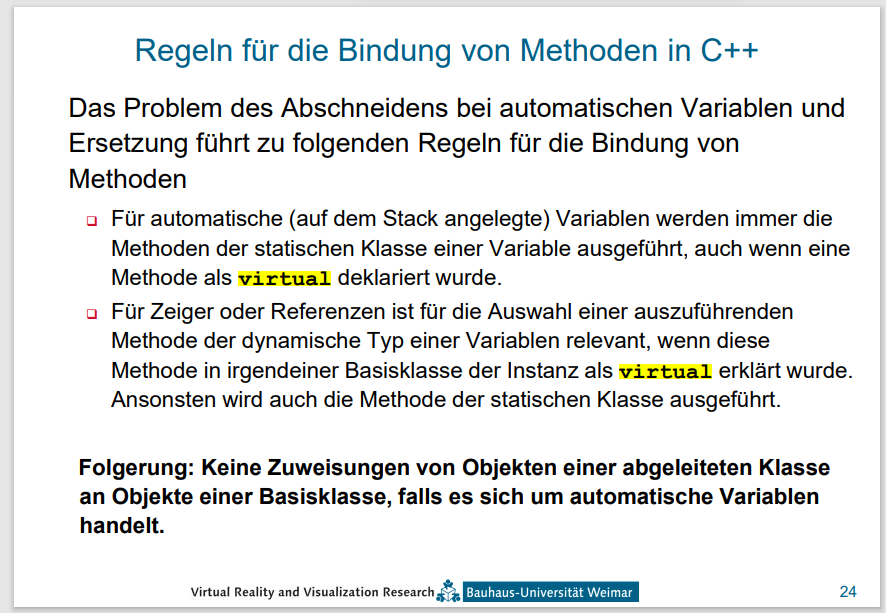


* Ohne virtual



Hier sorgt die Variable s2 für den Unterschied. Bei s1 werden in beiden fällen die Kon-/Destruktoren in der gleichen Reihenfolge aufgerufenn, bei s2 wird im Falle ohne des Schlüsselwortes virtual der Sphere Destrucotr nicht aufgerufen. Das Problem des Slicing (bei automatischen Variablen) wurde hier durch die dynamische erzeugunng umgangen. Ursache des Unterschieds ist, dass bei der Variablen s2 der statische Typ ein anderer ist (nämlich Shape) als der dynamische (hier Sphere). Für Zeiger oder Referenzen wird eine Methode des dynamischen Typs nur ausgeführt, wenn diese in irgendeiner Basisklasse der Instanz als virtual erklärt wurde (weil der Compiler zuerst auf die Methode des statischen Typs, also der Basisklasse, schaut)

Ansonsten wird auch die Methode der statischen Klasse ausgeführt -> genau das ist hier passiert, als virtual weggelassen wurde.



5.9

Klassenhierarchie vs. Objekthierarchie

Klassenhierarchie:

* Basisklasse -> Unterklasse(n) -> ggf. UnterUnterKlasse(n)
* Festgelegt
* Basiert auf der implementierten Vererbung

Objekthierarchie:

* Oberstes Objekt in der Hierarchie muss nicht aus der Basisklasse instanziiert sein, siehe Composite
* Beliebig konstruierbar
* Basiert auf ist-ein Beziehung

Klassendiagramme:

* Stellen den definierten (implementierten), allgemeinen Zustand der Klassenhierarchie dar
* Sind statisch

Objektdiagramme;

* Stellen Beziehungen konkreter Objekte zu einem bestimmten Zeitpunkt dar
* -sind dynamisch