KOENIG & BAUER

Polymorphie

ein Konzept der objektorientierten Programmierung

we're on it.

Object oriented programming

Father: Son, go and get Red Label
Son: 750ml or 1ltr?
*

Whisky

Mother: Son, go and get Red Label
Son: 500gms or 1kg?

Chees

Quelle: https://www.reddit.com/r/ProgrammerHumor/comments/8pysag/object_oriented_programming/

Die vier Säulen der OOP

Objektorientierte Programmierung

Abstraktion

class

Kapselung

private
protected
public

Vererbung

ParentClass
ChildClass

Polymorphismus

Polymorphismus

schnelle Fakten

- griechisch für **Vielgestaltigkeit**
- Konzept der OOP
- ermöglicht, dass Bezeichner abhängig von seiner Verwendung Objekte unterschiedlichen Datentyps annehmen können.
- tritt immer im Zusammenhang mit Vererbung und Interfaces auf

Methodenaufbau

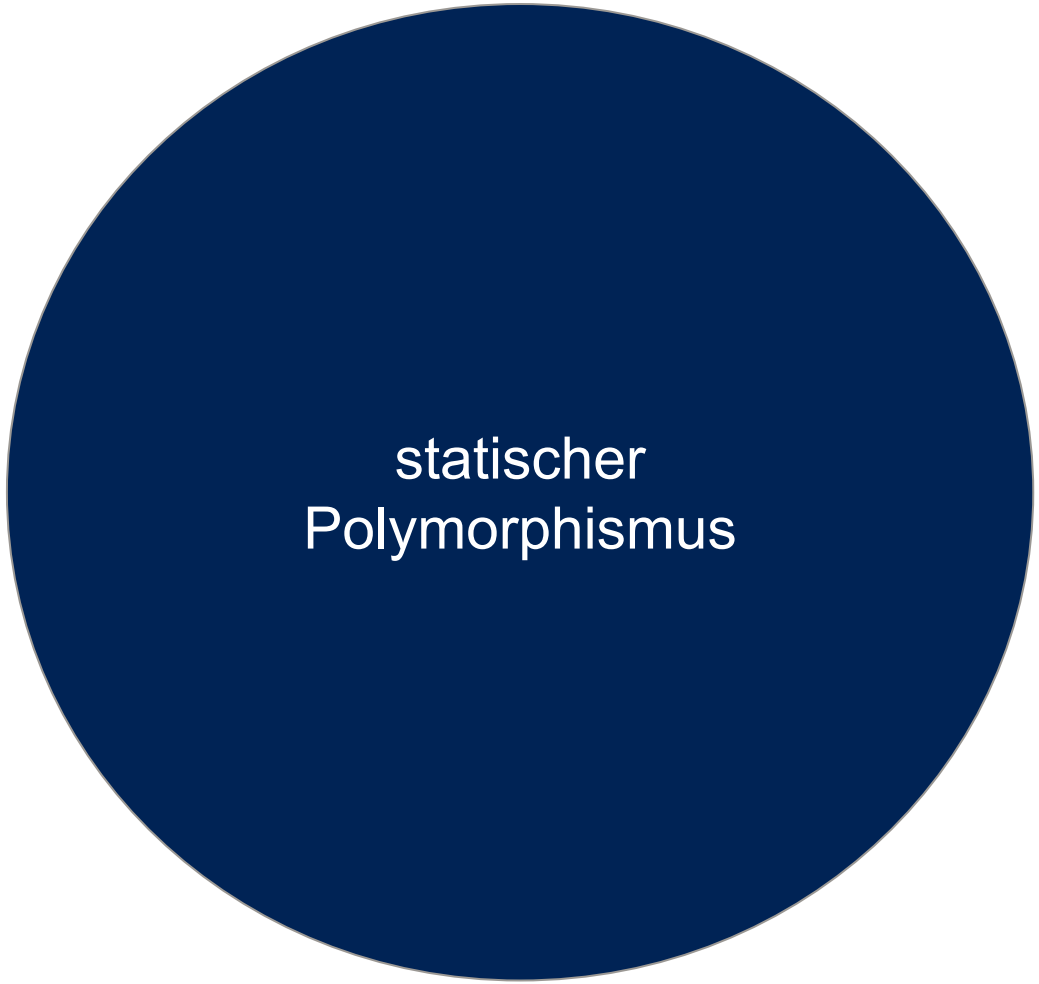
allgemein

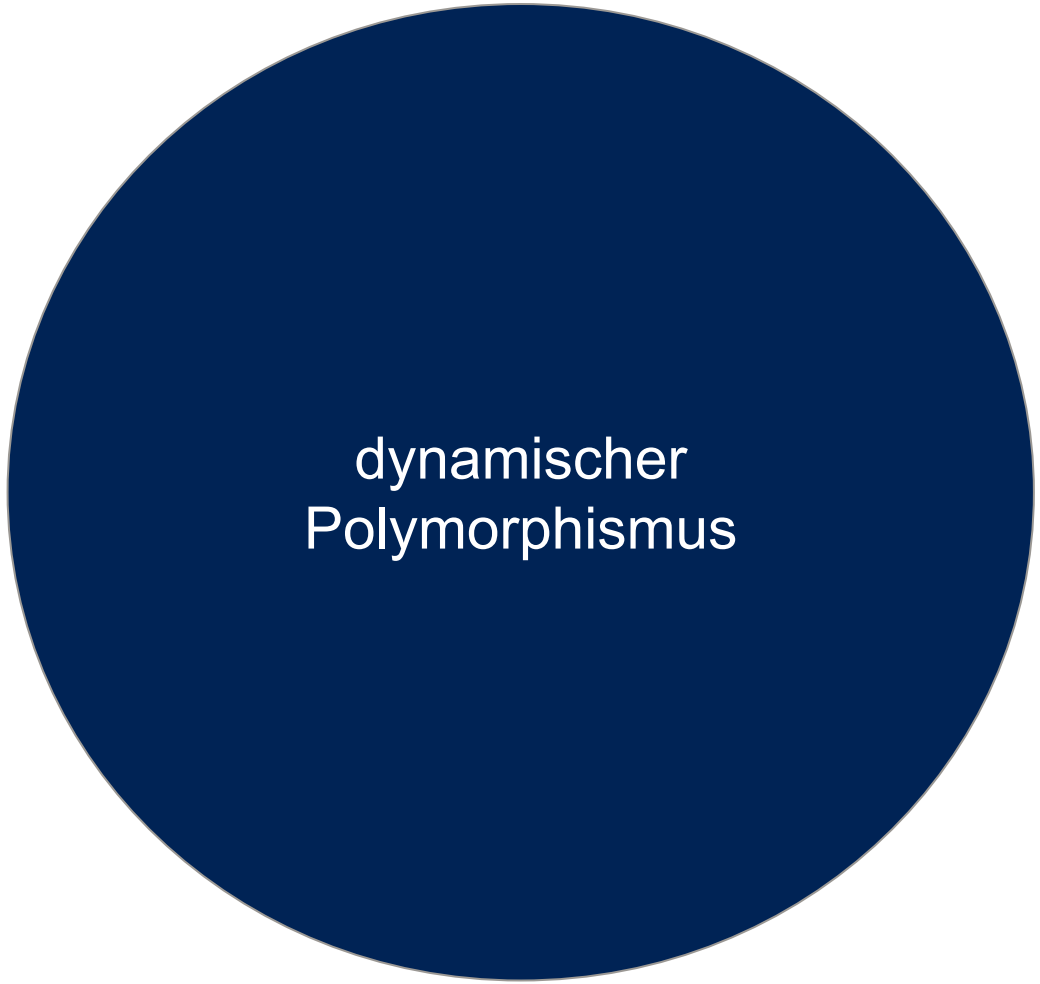
```
<Modifizierer> <Rückgabetyp> <Bezeichner> (<Parameterliste>)
{
    <Anweisung>
}
```

```
public void schmeckt()
{
    std::cout << “Das Bier schmeckt lecker” << std::endl;
}
```

Polymorphismus

Arten





Polymorphismus

statischer Polymorphismus - **Überladen** von Methoden

```
public void schmeckt()
{
    std::cout << “Das Bier schmeckt lecker” << std::endl;
}

public void schmeckt(string geschmack)
{
    std::cout << “Das Bier schmeckt nach”<< geschmack << std::endl;
}
```

Merke: Überladen

Beide Methoden haben den gleichen Namen
ABER Parameterlisten unterscheiden sich!

Polymorphismus

statischer Polymorphismus - **Überladen** von Methoden

- Methodensignatur unterscheidet sich **nur** in den Parameterlisten:
  - Typ: (int zahl) vs (string wort)
  - Anzahl: (int zahl) vs (int zahl, int zahl)
- Methoden sind in der **gleichen** Klasse oder Subklasse
- Beim kompilieren weiß der Compiler welche Methode er wählt

Polymorphismus

dynamischer Polymorphismus - **Überschreiben** von Methoden

Merke: Überschreiben

Rumpf ist unterschiedlich

[Class A]

```
public void schmeckt()
{
    return genuss;
}
```

[Class B :: public A]

```
public void schmeckt()
{
    return genuss * 2;
}
```

Quelle: https://images-na.ssl-images-amazon.com/images/I/91Yr-on9hQL._AC_SL1500_.jpg

Polymorphismus

dynamischer Polymorphismus - **Überschreiben** von Methoden

- Subklasse erbt Methode der Basisklasse
- Methode aus Basisklasse wird in Subklasse überschrieben
- Methodensignatur ist gleich
- **ABER**: Subklasse blendet geerbtes Verhalten aus!
    - Methodenrumpf (Anweisung) wird geändert

Überladen vs. Überschreiben

Quelle: https://www.reddit.com/r/ProgrammerHumor/comments/6e1zaa/overloading_vs_overriding/

späte/dynamische Bindung

Wir haben ein Objekt und Rufen eine Methode des Objektes auf

1. Prüfung, ob Methode i.d. Klasse vorhanden ist
   a. **JA**: Ausführen
   b. **NEIN**: Gehe zur Basisklasse der aktuellen Klasse

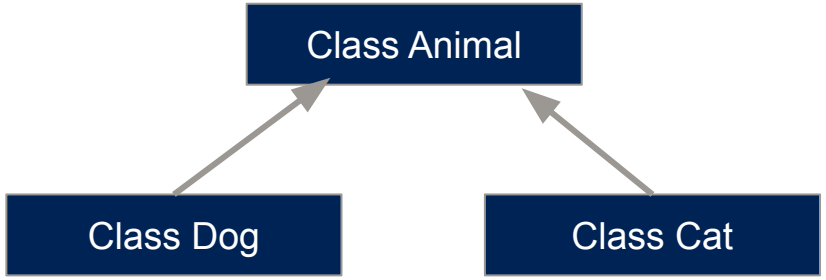
2. Ist Methode i.d. Basisklasse vorhanden?
   a. **JA**: Ausführen
   b. **NEIN**: Gehe zur Basisklasse der aktuellen Klasse

3. Wiederhole bis Methode gefunden wurde. Andernfalls wird Compiler Error ausgegeben

Substitutionsprinzip

“Das (...) Substitutionsprinzip (...) ist ein Kriterium in der objektorientierten Programmierung, das die Bedingungen zur Modellierung eines Datentyps für seinen Untertyp angibt.

Es besagt, dass ein Programm, das Objekte einer Basisklasse T verwendet, auch mit Objekten der davon abgeleiteten Klasse S korrekt funktionieren muss, ohne dabei das Programm zu verändern.” *(Wikipedia, Liskovsches Substitutionsprinzip)*




```
Animal* a = new Animal();
.
.
a -> makeNoise();          // *Animal Noise*
.
.
a = new Dog();
.
.
a -> makeNoise();              // *BARK BARK*
```

Zusammenfassung

- **Statischer Polymorphismus**
  - Methode wird **überladen (Compilezeit)**
  - Methodenname ist gleich
  - Parameterlisten sind verschieden

- **Dynamischer Polymorphismus**
  - Methode wird **überschrieben (Laufzeit)**
  - Methode wurde von Basisklasse geerbt
  - gleiche Methodensignatur
  - Rumpf wurde geändert
  - Zur Laufzeit wird entschieden, welche Methode aufgerufen wird

Codebeispiele (C++)

Github

git clone https://github.com/Nutzernam3/Polymorphie-Tutorial.git

Quelle: https://rapidapi.com/blog/wp-content/uploads/2017/01/octocat-768x576.gif

Ergänzendes Material

- https://de.wikipedia.org/wiki/Polymorphie_(Programmierung)
- https://de.wikipedia.org/wiki/Liskovsches_Substitutionsprinzip
- OpenBook(deutsch): Objektorientierte Programmierung
  http://openbook.rheinwerk-verlag.de/oop/oop_kapitel_05_002.htm
- OpenBook(deutsch): C++ in 21 Tagen
  http://mediainformatik.de/ftp/homes/guest/Prog3/C++%20%20in%2021%20Tagen.pdf
- Playlists zu C verwandten Sprachen:
  https://www.youtube.com/c/TheMorpheus407/playlists?view=50&sort=dd&shelf_id=5
- Video: Was ist Objektorientierte Programmierung?:
  https://www.youtube.com/watch?v=2le2YYr3N7s
- Video: Vererbung, Polymorphie und Liskovsches Substitutionsprinzip:
  https://www.youtube.com/watch?v=a_K3WFwxvjA

KOENIG & BAUER

Kris Myslowski
kris.myslowski@koenig-bauer.com

Koenig & Bauer AG

Vielen Dank für Ihre Aufmerksamkeit!

koenig-bauer.com
@koenigandbauer