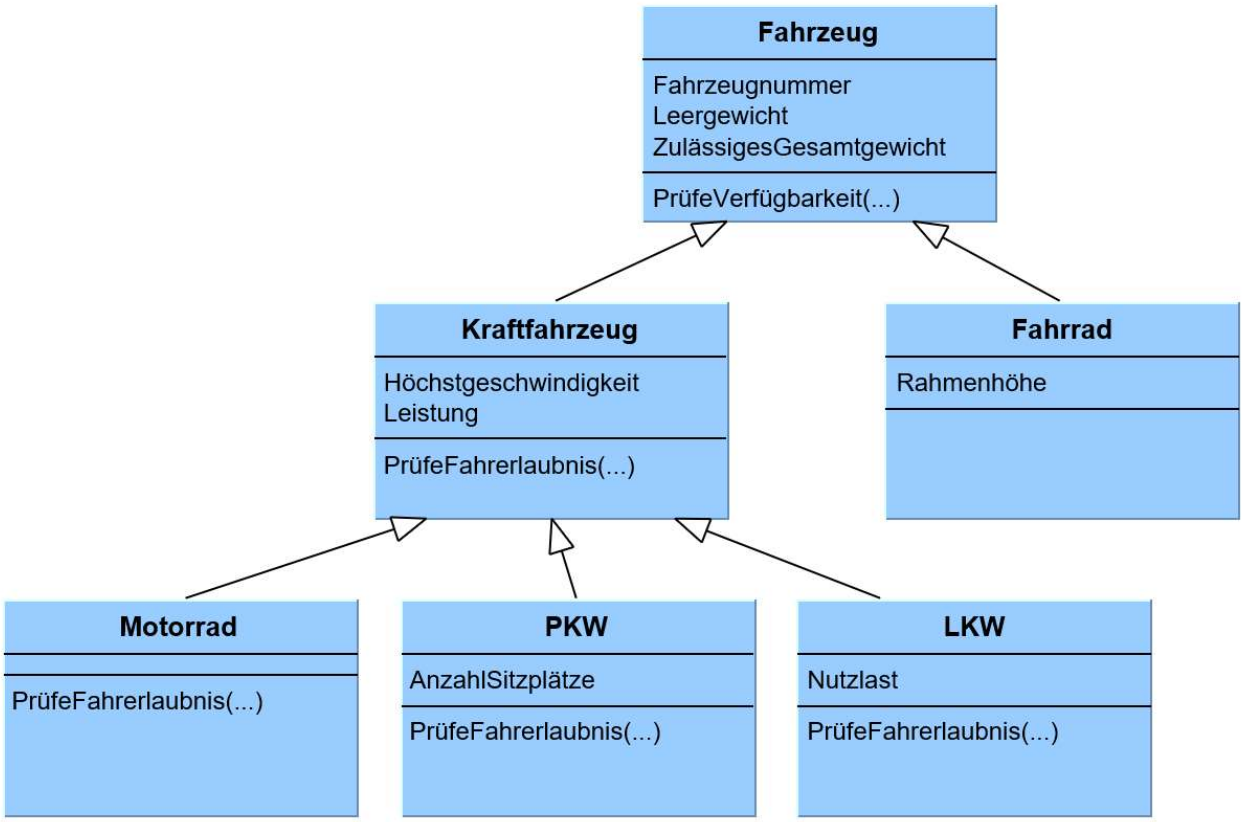
Objekt-orientierung in C#

Dienstag, 13. Dezember 2016 14:55

Vererbung: alle Member der Basisklasse werden quasi vom Compiler in die Subklasse kopiert



- In C# nur einfach Vererbung (genau eine Mutterklasse)
- dafür beliebige Anzahl von Schnittstellen
- **base**-Schlüsselwort zum Aufrufen der überschriebenen Basisklassenmember von abgeleiteten Klassen
- **sealed** Schlüsselwort verhindert Vererbung von dieser Klasse

Der Konstruktor

- spezielle Methode die automatisch bei Speicherallokation mit **new** aufgerufen wird
- kann nicht explizit aufgerufen werden
- wenn kein Konstruktor angegeben wird ein Standard-Konstruktor zur Verfügung gestellt
- spezielle Syntax: kein Rückgabewert, Name = Klassenname

- sind mehrere Konstruktoren vorhanden kann man diese nutzen mit **this**
- auf Konstruktoren der Mutterklasse kann mit **base** zugegriffen werden

Beispiel: Was tun die 3 Konstruktoren von Employee ?

```
class Person
{
   public string name;
   public Person(string name) {
      // perform some checks here
      this.name = name;
   }
}
class Employee : Person
{
   public int id;
   public Employee() : this("unbekannt", 0) { }
   public Employee(int id) : this("unbekannt", id) { }
   public Employee(string name, int id) : base(name)
   {
      // perform some checks here
      this.id = id;
   }
}
```

Polymorphie: auf einer Referenz der Basisklasse kann die spezifische Methode der Sub-klasse aufgerufen werden
Funktioniert in C# durch die Schlüsselwörter **`virtual`** und **`override`**

```
class Basisklasse {
    public virtual void test() {
        Console.WriteLine("Basis");
    }
}
class Subklasse : Basisklasse {
    public override void test() {
        Console.WriteLine("Sub");
    }
}
```

Aufruf:

```
    Basisklasse b = new Subklasse;
    b.test(); // Ausgabe: Sub
```

Schlüsselwort-Optionen für Klassenmethoden

- **virtual**: die Methode kann Inhalt enthalten und kann in einer Abgeleiteten Klasse polymorph überschrieben werden
- **override**: muss bei einer überschriebenen Methode angegeben werden um eine virtual Methode der Basisklasse zu überschreiben
- **sealed** eine überschriebene virtuelle Methode darf in Tochterklassen nicht weiter überschrieben werden
- **abstract**: die Methode ist leer und muss überschrieben werden
- **new**: wenn die zu überschreibende Methode nicht virtual in der Basisklasse ist kann override nicht verwendet werden -> new nötig, Polymorphismus funktioniert dann nicht
- **static:** Methode muss von der Klasse aus aufgerufen werden, nicht von einer Instanz
- **const**: Der Rückgabewert kann nicht verändert werden (Aufruf function()=1 geht nicht)

Optionen für Klassendeklaration

- **static:** Klasse darf nur statische Member enthalten und kann nicht instanziiert werden
- **abstract:** wenn eine abstracte Methode enthalten ist muss die Klasse als abstract markiert werden, verhindert instanziierung

Optionen für Variablen:

- **static**
  die Variable oder Klasse existiert einmalig für die Funktion oder Klasse und nicht für eine Instanz)

- **const** (die Variable ändert sich nicht)
  - const vor Parametern: Die Parameter können nicht verändert werden
  - const vor Membervariable: Variablenwert muss sofort zugewiesen werden, Variable ist dann automatisch auch static, beide Einschränkungen sind aufgehoben wenn **readonly** statt **const** verwendet wird

Zugriffsmodifizierer für Sichtbarkeit

	Überall	Bibliothek	abgeleitete Klassen	
private				-> Standard in einem Typ (z.B. in einer Klasse)
protected				
internal				
public				

Casting von Klassen

implicit cast in eine Basisklasse:

explicit cast in Subklasse:

as keyword:

is keyword:

Properties (Eigenschaft)

Property = Member-Variable die in einem Objekt definiert ist und auf die von außen zugegriffen werden soll

Zugriff auf Eigenschaften:

C++ Style **C#**

```
private int myVariable;

public int GetMyVariable()
{
    return myVariable;
}
private void SetMyVariable(int value)
{
    myVariable = value;
}

public void AccessVariables(int a)
{
    int b = GetMyVariable();
    SetMyVariable(a);
}
```

- Schreibaufwand reduziert
- Direkter Zugriff im Aufrufer

Auto-generated properties

Will man nur den schreibenden Zugriff auf eine Variable einschränken, kann man noch kürzer schreiben:

```
public int MyVariable {get; private set; }
```

- Dabei wird die zugrundeliegende Variable (vorher myVariable) nicht mehr benutzt
- Intern wird eine eigene Variable angelegt auf die nun kein direkter Zugriff mehr möglich ist

Object initialization Syntax

Objekte / Felder einer Klasseninstanz können direkt zugewiesen werden, ohne benutzerdefinierten Konstruktor

```
class Person {
    public string Nachname { get; set; }
    public string Vorname { get; set; }
    public Person() {}
}
…
Person p1 = new Person() {Nachname = "Hook"};
Person p2 = new Person() {Vorname = "Wendy"};
Person p3 = new Person() {Vorname = "Peter",
                          Nachname =  "Pan"};
```

Anonyme Objekte

- Klasse ohne Bezeichner-Namen
- zur schnellen Gruppierung von Elementen
- Der Typ dieser Objekte ist kompliziert -> var keyword benutzen