Referenztypen und der Umgang mit NULL in C

Wir definieren für unsere Beispiele 2 Klassen:

```
class Person
{
    public int age = 0;
}
// Entspricht Pupil extends Person
class Pupil : Person
{
    public string klasse = "";
}
```

Umgang mit NULL

Oftmals führt ein null Wert zu einer *NullReferenceException*. In C# gibt es 2 Operatoren, die den Umgang mit null erleichtern: ?. und ??. Folgendes Beispiel zeigt den Einsatz

```
string myStr = null;

if (myStr == null)
{
    Console.WriteLine("myStr kann NULL sein, da es ein Referenztyp ist.");
}

// Ermittelt die Länge des Strings. Dieser Code
// liefert eine NullReferenceException
int len = 0;
try
{
    len = myStr.Length;
}
catch (NullReferenceException)
{
    Console.Error.WriteLine("OOPS.")
}
```

In Java müssen wir umständlich mit *if* prüfen, ob der Wert *null* ist:

```
if (myStr == null)
{
    len = 0;
}
else
{
    len = myStr.Length;
}
```

In C# gibt es eine elegantere Möglichkeit. *?.* ist der "ternary conditional operator". Er liefert NULL, wenn myStr NULL ist und keine Exception. *??* ist

der NULL coalescing Operator. Er liefert 0, wenn der 1. Operand NULL ist.

```
len = myStr?.Length ?? 0;    // In len steht 0, wenn myStr null ist.
```

Bei der Prüfung von String ist die Methode *IsNullOrEmpty()* auch nützlich. Hier in Verbindung mit einer bedingten Zuweisung:

```
len = string.IsNullOrEmpty(myStr) ? 0 : myStr.Length;
```

Eigene Typen

Viele Eigenschaften aus Java lassen sich 1:1 übertragen.

```
Person p;             // p ist null.
p = new Person();     // Eine Person wird am Heap erstellt und
                      // die Referenzadresse in p geschrieben.
Person p2 = p;        // Nun habe ich eine Instanz, auf die 2
                      // Referenzvariablen zeigen.
p2.age = 18;          // p.age liefert natürlich auch 18.

Pupil pu = new Pupil();
Person p3;
```

In Zusammenhang mit der Vererbung ergeben sich folgende Besonderheiten:

```
p3 = (Person)pu;         // "Hinaufcasten" ist möglich, da die
                         // Vererbung ja eine "is-a" beziehung ist.
p3.klasse = "3BHIF";     // Geht natürlich nicht mehr.
object obj1 = pu;        // Alles ist von object abgeleitet.
obj1.age = 12;           // Natürlich nicht mehr möglich.
((Pupil)obj1).age = 12; // Das würde gehen, aber nur wenn obj1
                         // ein Pupil war.
```

is und as

In C# erleichtern die Schlüsselwörter *is* und *as* den Umgang mit Typencasts.

```
// Wenn pu nicht in Pupil umgewandelt werden kann, wird NULL
// geliefert. In diesem Fall wird eine neue Instanz von
// Pupil erstellt.
p3 = pu as Pupil ?? new Pupil();
// true, da is angibt, ob ein Typencast durchgeführt werden kann.
if (pu is Person)
{
    Console.WriteLine("pu is Person.");
}
// In object gibt es die Methode GetType() und ToString().
// Liefert "ReferenceTypesApp.Pupil".
string type = pu.GetType().ToString();
```