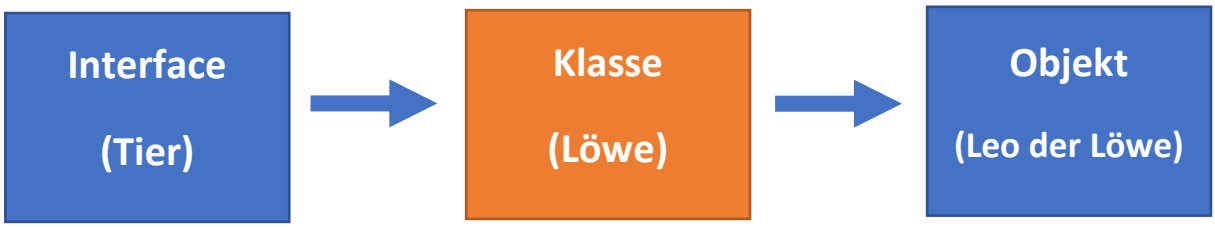
INTERFACES

WAS SIND INTERFACES

In den letzten Videos haben wir uns mit den sogenannten Interfaces (zu Deutsch: Schnittstellen) beschäftigt und in diesem Dokument fassen wir das ganze Thema mal zusammen. Stellen wir uns also zuerst einmal die Frage: Was sind Interfaces?

Ein Interface ist wie ein Vertrag den eine Klasse unterschreibt, der ihr vorschreibt was für Member diese zu beinhalten hat. Während eine Klasse beschreibt, wie ein Objekt aussieht, beschreibt ein Interface wie eine Klasse auszusehen hat.



Man kann mit Interfaces dafür sorgen, dass eine bestimmte Gruppe von Klassen (z.B. Klassen, die jeweils unterschiedliche Tiere beschreiben), immer denselben grundlegenden Aufbau haben. Das Interface „Tier“ könnte somit zum Beispiel vorgeben, dass jede Klasse die ein Tier beschreiben soll eine „Essen()“- bzw. „Trinken()“-Methode implementiert. Das sorgt für einen weitaus konsequenteren Code, da Fehler bzw. unterschiede beim Aufbau der Klassen nicht toleriert werden.

WIE DEFINIERT MAN EIN INTERFACE?

Ein Interface zu definieren ist sehr einfach. Im Grunde definiert man ein Interface genauso wie eine Klasse auch: **Rechtsklick auf den Haupt-Namespace im Projektmappenexplorer → „Hinzufügen“ → „Klasse“ → „Schnittstelle“ auswählen.** Jetzt musst du dem Interface nur noch einen Namen geben und schon du hast ein Interface definiert. WICHTIG: Bei Interfaces schreibt man zugunsten der Übersichtlichkeit immer ein „I“ (großes i) an den Anfang des Namens. Beispiel: „ITier“.

```
interface ITier
{

}
```

Abbildung 1. Ein leeres Interface

Wie du siehst erinnert ein Interface alleine vom Aussehen schon stark an Klassen. Der einzige Unterschied ist das Schlüsselwort „interface“ statt „class“.

Jetzt können wir den Inhalt des Interfaces, wie bei Klassen auch, in den dazugehörigen Codeblock hineinschreiben. Der Inhalt von Interfaces ist dabei allerdings nur eine Vorlage für erbende bzw. implementierende Klassen und dementsprechend wird im Interfaces selber nichts wirklich codiert.

Sprich: Keine Zugriffsmodifizierer, keine Methodenkörper usw… Wir geben nur an, was für Namen und Typen die Member letztendlich in einer Klasse haben sollen.

```
interface ITier
{
    //Eigenschaften
    string Geschlecht { get; set; }

    //Methoden
    void Essen();
    void Trinken();
}
```

Abbildung 2. Ein fertiges Interface

Wie du siehst hat die Eigenschaft „Geschlecht“ keinen Zugriffsmodifizierer, da ein Interface nur als eine abstrakte Vorlage dienen soll. Außerdem haben die Methoden nur einen Namen, eine Signatur und die Art von Rückgabewert. Die eigentliche Codierung der Funktion findet dann innerhalb von einer Klasse statt, die das Interface implementiert (also davon erbt).

WIE IMPLEMENTIERT MAN EIN INTERFACE?

Wenn eine Klasse von einem Interface erbt, dann spricht man von der sogenannten Implementierung. Beim erben des Interfaces erklärt sich die Klasse damit einverstanden, dass sie jeden Member des Interfaces selber beinhalten muss (diesmal aber in auscodierter fertiger Form). Wir implementieren also alle vorgegebenen Member der Schnittstelle in unserer Klasse.

Wie man von einer Schnittstelle erben kann siehst du im folgenden Codeschnipsel:

```
class Löwe : ITier
{
    public string Geschlecht { get; set; }

    public void Essen()
    {
        Console.WriteLine("Der Löwe isst...");
    }

    public void Trinken()
    {
        Console.WriteLine("Der Löwe trinkt...");
    }
}
```

Wie du siehst erbt die Klasse „Löwe“ von der Schnittstelle „ITier“ und sie implementiert jeden sich darin befindlichen Member. Dabei sind die Methoden dieses Mal auch richtig auscodiert.