Deklarationen

Deklarationsraum

Programmbereich, zu dem eine Deklaration gehört

Arten von Deklarationsräumen

- **Namespace**: Deklarationen von Klassen, Interfaces, Structs, Enums, Delegates
- **Klasse**, **Interface, Struct**: Deklarationen von Feldern, Methoden, ...
- **Enum**: Deklarationen von Enumerationskonstanten
- **Block**: Deklarationen lokaler Variablen

Regeln

Deklarationsregeln

- Kein Name darf in einem Deklarationsraum auf gleicher Ebene mehrfach deklariert werden.
- Er darf aber in inneren Deklarationsbereichen neu deklariert werden (außer in inneren Anweisungsblöcken).

Sichtbarkeitsregeln

- Ein Name ist in seinem ganzen Deklarationsraum sichtbar, lokale Variablen allerdings erst ab ihrer Deklaration.
- Deklarationen in inneren Deklarationsräumen verdecken gleichnamige Deklarationen aus äußeren Deklarationsräumen.
- Kein Name ist außerhalb seines Deklarationsraums sichtbar.
- Sichtbarkeit kann mit Attributen eingeschränkt werden (private, protected, internal, ...)

Deklarationsraum Namespace

File: XXX.cs

```
namespace A {
    ... Classes ...
    ... Interfaces ...
    ... Structs ...
    ... Enums ...
    ... Delegates ...
    namespace B {  // voller Name: A.B
        ...
    }
}
```

File: YYY.cs

```
namespace A {
    ...
    namespace B {...}
}
```

```
namespace C {...}
```

Gleichnamige Namespaces in verschiedenen Dateien bilden gemeinsamen Deklarationsraum

Eingeschachtelte Namespaces bilden eigenen Deklarationsraum

Benutzung fremder Namespaces

Color.cs

```
namespace Util {
    public enum Color {...}
}
```

Figures.cs

```
namespace Util.Figures {
    public class Rect {...}
    public class Circle {...}
}
```

Triangle.cs

```
namespace Util.Figures {
    public class Triangle {...}
}
```

```
using Util.Figures;

class Test {
    Rect r;          // Benutzung ohne Qualifikation (weil using Util.Figures)
    Triangle t;
    Util.Color c;    // Benutzung mit Qualifikation
}
```

Fremde Namespaces müssen entweder

- mit *using* importiert oder
- als Qualifikation vor verwendeten Namen geschrieben werden

Fast jedes Programm benötigt Namespace System => using System;

Dekl.raum Klasse, Interface, Struct


```
class C {  // gilt auch für Structs
      ... Felder, Konstanten ...
      ... Methoden ...
      ... Konstruktoren, Destruktoren ...
      ... Properties ...
      ... Indexers ...
      ... Events ...
      ... überladene Operatoren ...
      ... geschachtelte Typen (Klassen, Interfaces, Structs, Enums, Delegates) ...
}
```

```
interface IX {
      ... Methoden ...
      ... Properties ...
      ... Indexers ...
      ... Events ...
}
```

Deklarationsraum der Basisklasse gehört nicht zum Deklarationsraum der Unterklasse
=> gleichnamige Deklarationen in der Unterklasse erlaubt.

Deklarationsraum Enum


```
enum E {
      ... Enumerationskonstanten ...
}
```

Deklarationsraum Block

Verschiedene Arten von Blöcken

```
void foo (int x) {                    // Methodenblock
      ... lokale Variablen ...
      if (...) {                      // geschachtelter Block
            ... lokale Variablen ...
      }
      for (int i = 0; ...) {          // for-Block
            ... lokale Variablen ...
      }
}
```

Deklarationsraum eines Blocks schließt Deklarationsräume geschachtelter Blöcke ein.

Formale Paramater gehören zum Deklarationsraum des Methodenblocks.

Deklaration einer Laufvariable gehört zum Deklarationsraum des for-Blocks.

Deklaration einer lokalen Variablen muß ihrer Verwendung vorausgehen.

Deklaration lokaler Variablen

```
void foo(int a) {
      int b;
      if (...) {
            int b;          // Fehler: b bereits in äußerem Block deklariert
            int c;
            int d;
            ...
      } else {
            int a;          // Fehler: a bereits im äußeren Block deklariert (Parameter)
            int d;          // ok: keine Überschneidung mit d im then-Block
      }
      for (int i = 0; ...) {...}
      for (int i = 0; ...) {...}    // ok: keine Überschneidung mit i aus letzter Schleife
      int c;                        // Fehler: c bereits in einem inneren Block deklariert
}
```