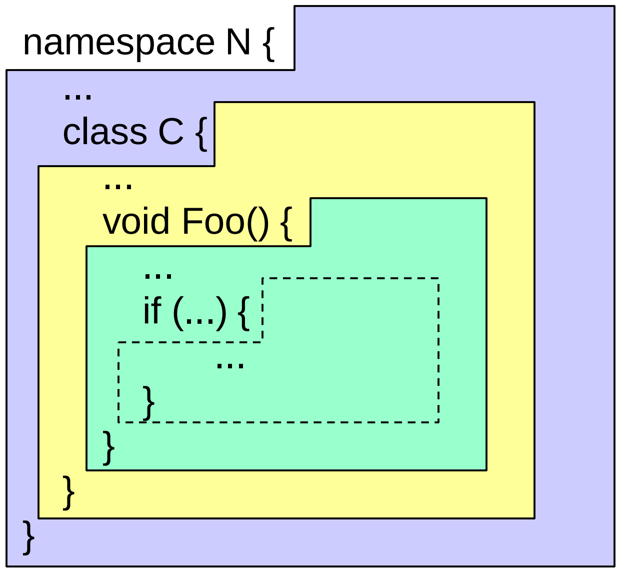
Declarations

Declaration Space

The program area to which a declaration belongs

Kinds of declaration spaces

- **namespace**:	declarations of classes, interfaces, structs, enums, delegates
- **class, interface, struct**:	declarations of fields, methods, ...
- **enumeration**:	declarations of enumeration constants
- **method block**:	declarations of local variables



Statement blocks are not declaration spaces on their own but belong to the declaration space of the enclosing method block

Rules

Scoping rules

- No name may be declared more than once in the same declaration space on the same level.
- However, it may be redeclared in an inner declaration space (except in a nested statement block)

Visibility rules

- A name is visible in its whole declaration space (local variables only from the point of their declaration onwards). This implies that the use of a name may precede its declaration (except for local variables, which must be declared before they are used)
- If a name is redeclared in an inner declaration space, it hides the same name from the outer declaration space.
- In general, no name is visible outside its declaration space.
- The visibility of names declared in namespaces, classes, structs and interfaces can be controlled by the modifiers *public*, *private*, *protected* and *internal*.
- Names of enumeration constants can only be accessed if they are qualified with their enumeration type name.

Namespaces

File: X.cs

```
namespace A {
    ... classes ...
    ... interfaces ...
    ... structs ...
    ... enumerations ...
    ... delegates ...
    namespace B {  // full name: A.B
        ...
    }
}
```

File: Y.cs

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

namespace C {...}
```

Equally named namespaces in different files constitute a single declaration space.

Nested namespaces constitute a declaration space on their own.

Using Other 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;          // without qualification (because of using Util.Figures)
    Triangle t;
    Util.Color c;    // with qualification
}
```

Foreign namespaces

- must either be imported (e.g. *using Util;)*
- or specified in a qualified name (e.g. *Util.Color*)

Most programs need the namespace System => using System;

Classes, Interfaces, Structs

```
class C {  // applies also to structs
      ... fields, constants ...
      ... methods ...
      ... constructors, destructors ...
      ... properties ...
      ... indexers ...
      ... events ...
      ... overloaded operators ...
      ... nested types (classes, interfaces, structs, enumerations, delegates) ...
}
```

```
interface IX {
      ... methods ...
      ... properties ...
      ... indexers ...
      ... events ...
}
```

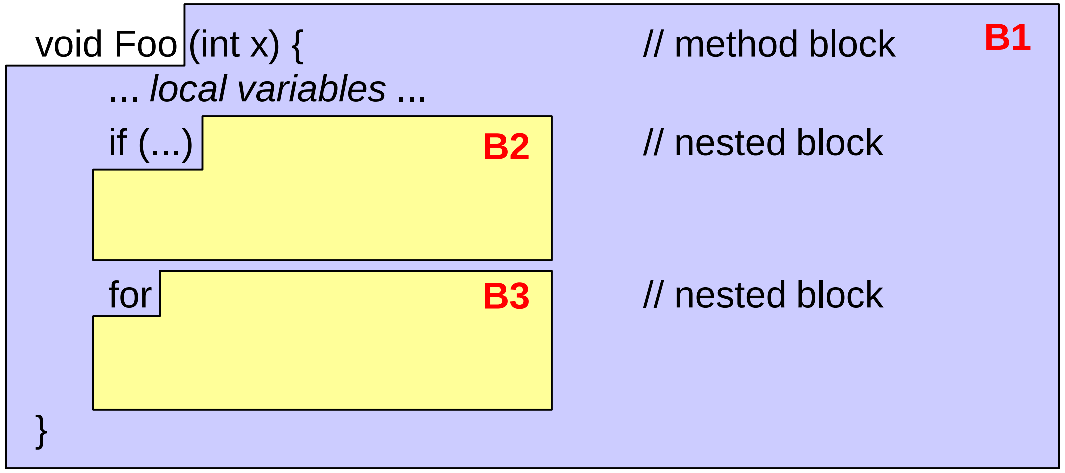
The declaration space of a subclass does not belong to the declaration space of its base class
=> it is ok to declare the same names in a base class and in its subclasses.

Enumerations

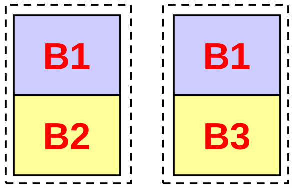
```
enum E {
	... enumeration constants ...
}
```

Statement Blocks

Kinds of blocks



The declaration space of a block includes the declaration spaces of its nested blocks.



- Formal parameters belong to the declaration space of their method block.
- The loop variable of a *for* statement belongs to the block of this *for* statement.
- The declaration of a local variable must precede the use of this variable.

Declaration of Local Variables

```
void Foo(int a) {
        int b;
        if (...) {
                int b;                  // error: b is already declared in the outer block
                int c;
                int d;
                ...
        } else {
                int a;                  // error: a is already declared in the outer block (parameter)
                int d;                  // ok: no conflict with d in the if block
        }
        for (int i = 0; ...) {...}
        for (int i = 0; ...) {...}      // ok: no conflict with i from the previous loop
        int c;                          // error: c is already declared in a nested block
}
```