Advanced C#

H.Mössenböck

University of Linz, Austria

moessenboeck@ssw.uni-linz.ac.at

Contents
- Inheritance
- Interfaces
- Delegates
- Exceptions
- Namespaces and Assemblies
- Attributes
- Threads
- XML Comments

Inheritance

Syntax

```
class A {                        // base class
    int a;
    public A() {...}
    public void F() {...}
}
```

```
class B : A {                    // subclass (inherits from A, extends A)
    int b;
    public B() {...}
    public void G() {...}
}
```

- B inherits *a* and *F*(), it adds *b* and *G*()
  - constructors are not inherited
  - inherited methods can be overridden (see later)
- Single inheritance: a class can only inherit from one base class, but it can implement multiple interfaces.
- A class can only inherit from a class, not from a struct.
- Structs cannot inherit from another type, but they can implement multiple interfaces.
- A class without explicit base class inherits from *object*.

Asignments and Type Checks

```
class A {...}
class B : A {...}
class C: B {...}
```

Assignments

```
A a = new A();      // static type of a: the type specified in the declaration (here A)
                    // dynamic type of a: the type of the object in a (here also A)
a = new B();        // dynamic type of a is B
a = new C();        // dynamic type of a is C

B b = a;            // forbidden; compilation error
```

Run time type checks

```
a = new C();
if (a is C) ...     // true, if dynamic type of a is C or a subclass; otherwise false
if (a is B) ...     // true
if (a is A) ...     // true, but warning because it makes no sense

a = null;
if (a is C) ...     // false: if a == null, a is T always returns false
```

Checked Type Casts

Cast

```
A a = new C();
B b = (B) a;          // if (a is B) stat.type(a) is B in this expression; else exception
C c = (C) a;

a = null;
c = (C) a;            // ok ➔ null can be casted to any reference type
```

as

```
A a = new C();
B b = a as B;         // if (a is B) b = (B)a; else b = null;
C c = a as C;

a = null;
c = a as C;           // c == null
```