Method declaration

Methods is always declared in a class

Methods can be declared as

- class methods (static)
  - can be used without an object
  - can acces the class-variables (static) of the class
  - can not access the object-variable of the class
  - is often used for stateless method, where all information is delivered in the parameters – see fx the Math class.
- object methods
  - can only be used on an object (instance) of a class
  - can access the class-variables (static)
  - can access the object-variables of the class

Method declaration

Special methods

- constructors
  - have the same name as the class
  - have no return type
  - is used with the class-name when creating an new object of the class
- destructors
  - is called from the garbage collector just before object is released
  - normaly not used, while garbage collection is handled automatic
- properties
  - special declaration for a get- and set method (for accessing a variable)
  - used for encaptulating variabels
  - gives the posibility to make code like it was a variable even it is methods
  - can be used both for a class (static) and an object method

Encapsulation of object and class methods

Methods may be

- public: can be accessed direct outside the class / object
- internal: can be accessed direct outside the class / object
  but only from within the namespace
- protected: can be accessed direct from a inherited class but else not
- private: can not be accessed outside the class,
  neither from inherited class

Parameters and method-type

Methods (except constructors and destructors) have one return type

- can be a void (empty) type
- can be a basic type (int, float … char, boolean)
- can be an array
- can be a class-type (returns a object of the class)

Methods can have parameters

- value parameter
- reference parameters (ref or out)

Both can be of

- can be basic types (int, float … char, boolean)
- can be arrays
- can be objects (class type)

Parameters and method-type

In a class declaration you can declare

- more than one consructor
  - all must have difference parameter types
- more than method with the same name
  - all must have the same return type
  - all must have difference parameter types
- abstract methods to inherid from
- override methods to override iherided methods

Excample of properites

C# properties

```
class MyClass
{
        private int x;
        public int X
        {
                get { return x; }
                set { this.x = value }
        }
}
```

Use:

```
MyClass mc = new MyClass();
int v = mc.X;
mc.X = v;
```

java get-/set-methods

```
class MyClass
{
        private int x;
        public int getX () {
                return x;
         }
        public void setX (int value) {
                this.x = value }
        }
}
```

Use:

```
MyClass mc = new MyClass();
int v = mc.getX ( );
mc.setX ( v );
```