



# Klassen, Namensräume (namespaces)

Christoph Bichlmeier

09 November 2006

chair for computer aided medical procedures

department of computer science | technische universität münchen



# Namensraum (namespace)

- Bei größeren Projekten wird die Namensgebung von Klassen und Funktionen unübersichtlich.
- Deshalb wurden in C++ Namensräume eingeführt.

```
namespace myNamespace{  
    class MyClass{  
        MyClass();  
        void myNonsenseFunction(){ return; }  
    };  
}
```

myClass.h

```
include "myClass.h"  
int main(){  
    myNamespace::MyClass someObject;  
    someObject = new myNamespace::MyClass;  
    return 0;  
}
```

main.cpp

```
include "myClass.h"  
using namespace myNamespace;  
int main(){  
    MyClass someObject;  
    someObject = new MyClass;  
    return 0;  
}
```

main.cpp



## Beispiel: Die Klasse string in std

- Die Klasse string bietet einen wesentlich komfortableren Zugriff auf Zeichenketten als `char*` .

```
include <string>
int main(){
std::string s = "Was ist die Losung";
std::cout << s << std::endl;
std::cin >> t;
if( t.compare("per aspera ad astra")==0) {
    std::cout << "Richtig!"<< std::endl;
}
return 0;
}
```



# Klassen und Objekte

- Kurz zur Wiederholung
  - Klasse ist ein Objekttyp --- Objekt ist eine Instanz einer Klasse
  - Objekte bestehen aus Daten (Membervariablen) und zugehörigen Funktionen (Memberfunktionen)
  - statische Variablen gibt es von einer Klasse nur *einmal* pro Prozess
    - eine Änderung einer statischen Variable wirkt sich auf *alle* Instanzen aus
  - statische Funktionen können ohne Instanz aufgerufen werden (dürfen aber natürlich nicht auf Member zugreifen)
  - nur virtuelle Funktionen in vererbten Klassen können überschrieben werden und werden zur Laufzeit bestimmt
  - abstrakte Klassen sind Klassen, von denen man kein Objekt erzeugen kann, weil nicht alle virtuellen Funktionen implementiert sind



# Klassen und Objekte: Der Header

```
class MyClass:MyParentClass
{ // die Klasse MyClass erbt von MyParentClass
public:
    MyClass(); //Konstruktor
    MyClass(std::string text); //zweiter Konstruktor
    virtual ~MyClass; // Destruktor

    void fu(); // eine Funktion
    virtual int fun(); // eine virtuelle Funktion
    virtual int func()=0; // eine rein virtuelle (=abstrakte) Funktion
    static double funct(); // eine statische Funktion

    std::string m_someString; // eine Membervariable
    static int m_someNumber; // eine statische Membervariable
}
```

myClass.h



# Klassen und Objekte: Die Implementierung

```
include "myClass.h"

MyClass::m_someNumber = 5;

MyClass::MyClass() {
    m_someString = "Eine Intialisierung vom Text";
}; //Konstruktor

MyClass::MyClass(std::string text) {
    m_someString = text;
}; //zweiter Konstruktor

MyClass::~MyClass(){}; // Destruktor

void MyClass::fu(){}; // eine Funktion
int MyClass::fun(){return 4;}; // eine virtuelle Funktion
double MyClass::funct(){return 2;}; // eine statische Funktion
```

myClass.cpp



# Gegenüberstellung Syntax

## C++

- Vererbung :
- Mehrfachvererbung  
:Klasse1, Klasse2
- Zugriff auf Member  
`objekt.funktion()` bei  
Referenzen und  
`objekt->funktion()` bei  
Pointern
- globale Variablen und  
Funktionen
- Templates <>

## Java

- Vererbung `extends`
- Mehrfachvererbung  
`implements Klasse1, Klasse2`  
(nur für abstrakte Klassen  
ohne Membervariablen)
- Zugriff auf Member  
`objekt.funktion()`
- keine globale Variablen
- Generics (ab JAVA 1.5)