Klassen

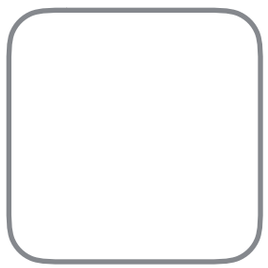
C++ Übung am 02. Juni 2016

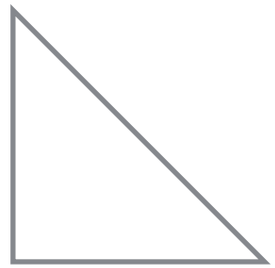
Klassen

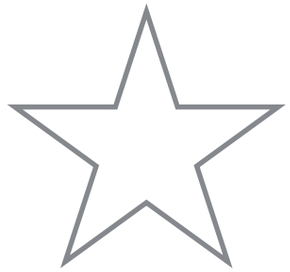
- der Kern objektorientierter Programmierung
- Festlegung der Daten und Funktionen einer Klasse
- Deklaration als public oder private
- Konstruktoren erzeugen ein Objekt der Klasse

Klassen

Klassen







Attribute:

Größe

Form

Methoden:

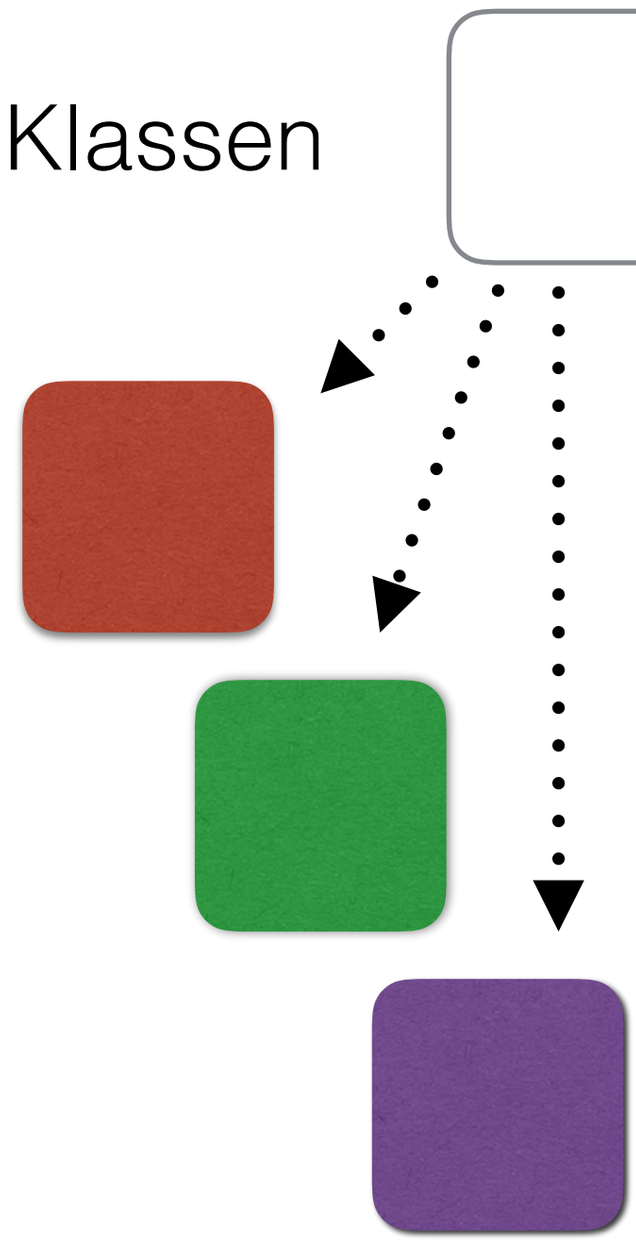
vergrößere()

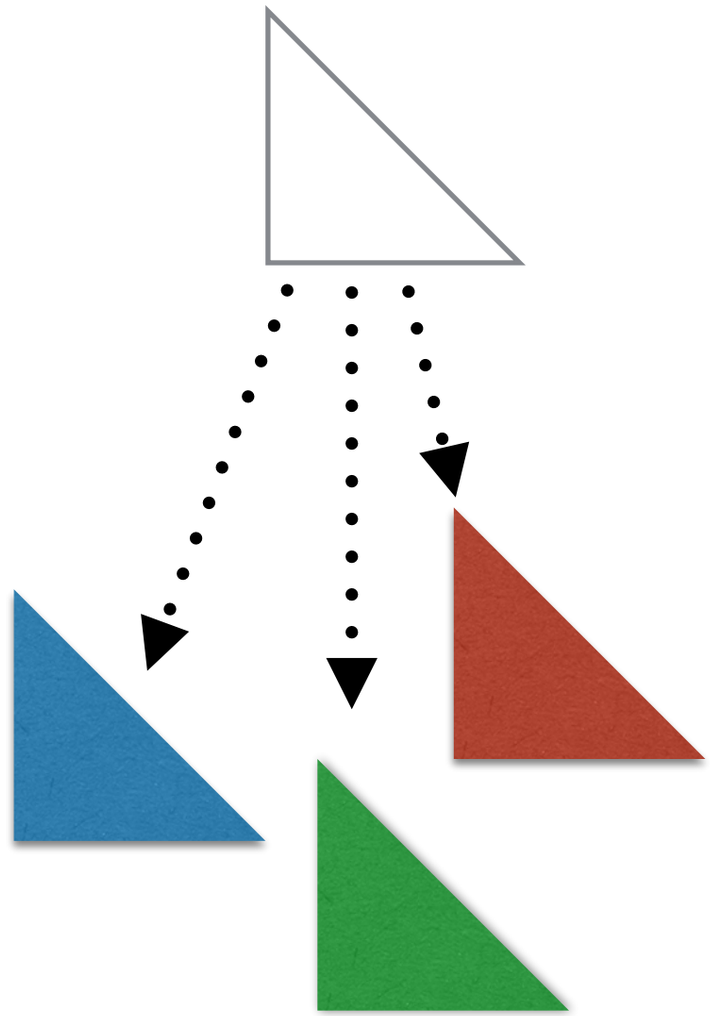
färbe()

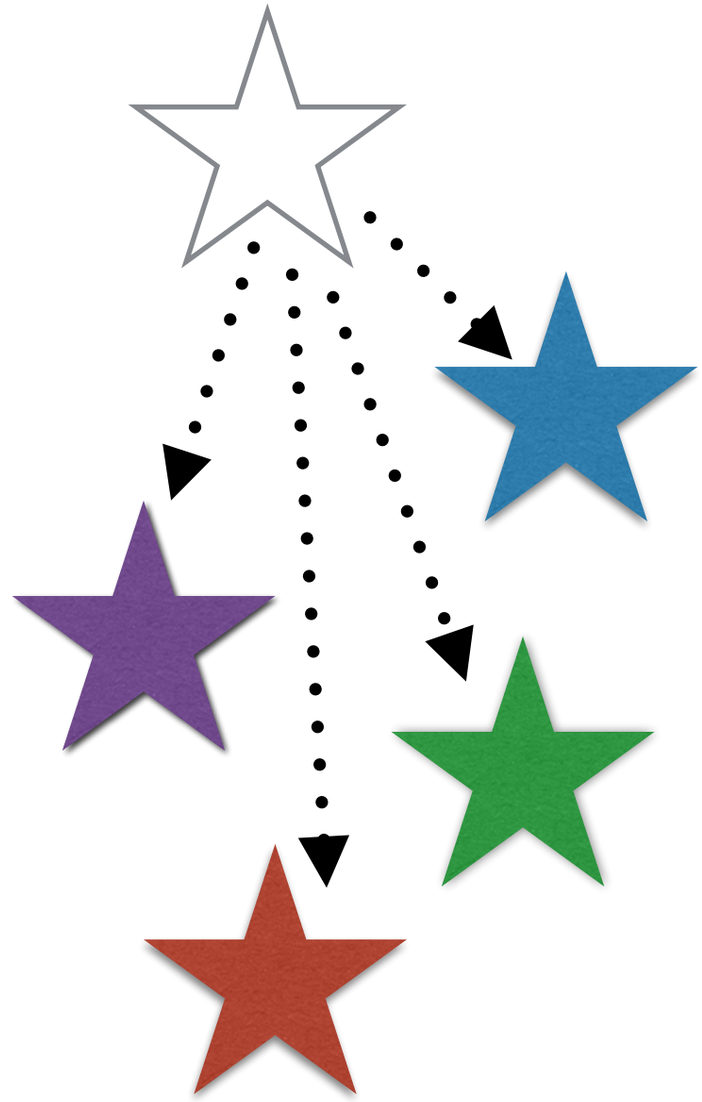
rotiere()

Klassen

Klassen







Objekte = Instanzen

Objekte

Attribute:

Größe = 3 cm

Farbe = grün

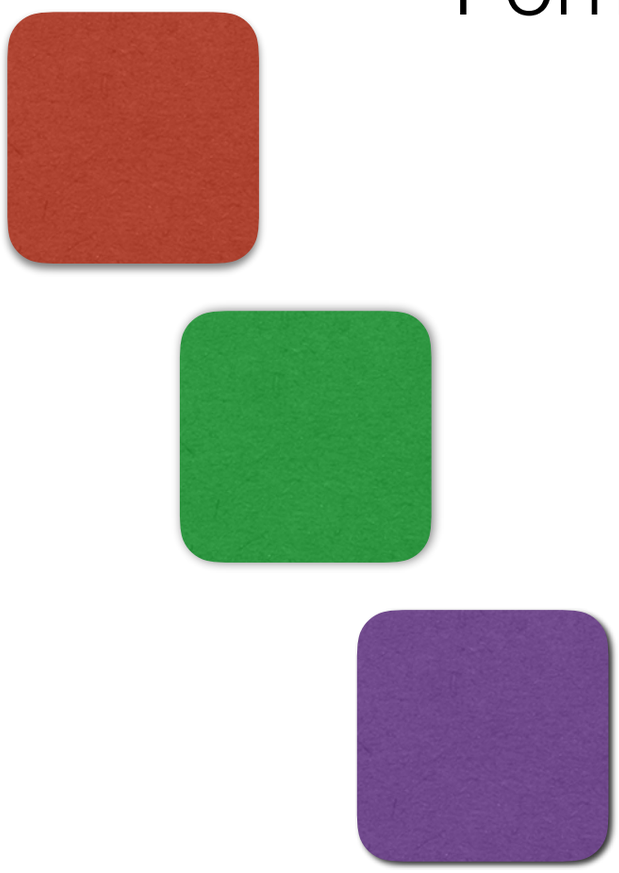
Form = Dreieck

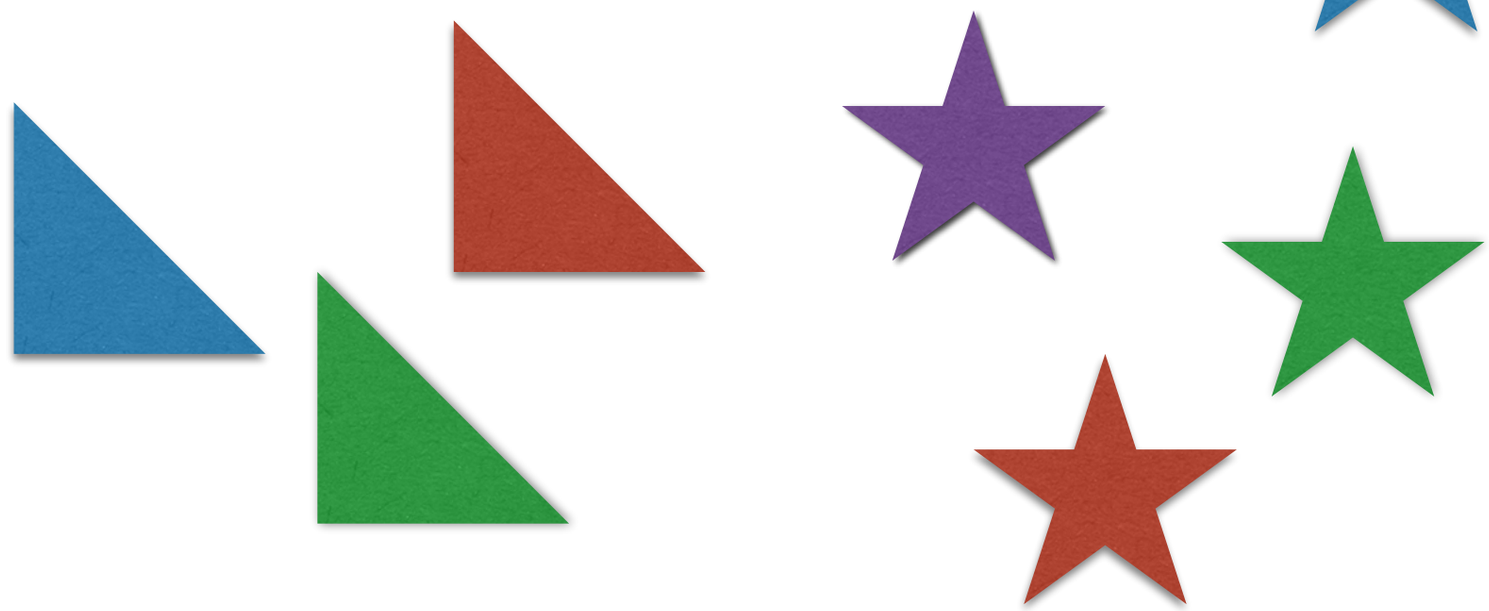
Methoden:

vergrößere(4cm)

färbe(blau)

rotiere(90°)





Objekte = Instanzen

Mensch.hpp

```
#ifndef MENSCH_HPP
#define MENSCH_HPP

#include <iostream>
using namespace std;

class Mensch {
    public:
        wstring nachname;
        wstring vorname;
        wstring wohnort;
        void zieheUm(wstring neuerWohnort);
        bool altGenug(int benoetigtesAlter);
        Mensch(wstring nachname, wstring vorname, wstring
               wohnort, int alter);
    private:
        int alter;
        int wieAlt();
};

...
```

```
Mensch::Mensch(wstring nachname, wstring vorname, wstring
                wohnort, int alter) {
    this->nachname = nachname;
    this->vorname = vorname;
    this->wohnort = wohnort;
    this->alter = alter;
};

void Mensch::zieheUm(wstring neuerWohnort) {
    wohnort = neuerWohnort;
};
```

Mensch.hpp

```
bool Mensch::altGenug(int benoetigtesAlter) {
    if (wieAlt() >= benoetigtesAlter) {
        return true;
    } else {
        return false;
    }
};

int Mensch::wieAlt() {
    return alter;
};

#endif
```

MainMensch.cpp

```
#include <iostream>
#include "Mensch.hpp"
using namespace std;

int main() {
    Mensch susi(L"Maier", L"Susi", L"Berlin", 25);
    Mensch franz(L"Maier", L"Franz", L"Berlin", 11);

    wcout << L"Susi wohnt in " << susi.wohnort << endl;
    wcout << L"Franz wohnt in " << franz.wohnort << endl;
    wcout << L"Susi ist so alt: " << susi.alter << endl; // FEHLER

    if (susi.altGenug(18)){
        susi.zieheUm(L"Muenchen");
        wcout << L"Susi wohnt jetzt in " << susi.wohnort << endl;
    } else {
        wcout << L"Susi ist nicht alt genug um umzuziehen"<< endl;
    }
    if (franz.altGenug(18)){
        franz.zieheUm(L"Muenchen");
        wcout << L"Franz wohnt jetzt in " << franz.wohnort << endl;
    } else {
        wcout << L"Franz ist nicht alt genug um umzuziehen" << endl;
    }
}
```

Kapselung

private:

Funktionen und Daten sind nur von innerhalb der Klasse aus erreichbar

```
wcout << L"Susi ist so alt: " << susi.alter << endl;
```

-> FEHLER

public:

Funktionen und Daten der Klasse sind auch von anderen Programmteilen aus erreichbar

```
wcout << L"Franz wohnt in " << franz.wohnort << endl;
```

Konstruktor

Default-Konstruktor:

- wenn kein Konstruktor implementiert wurde
- reserviert Speicherplatz
- erstellt eine Instanz der Klasse
- weist den Variablen aber keinen Wert zu

Konstruktor

Default-Konstruktor:

- wenn kein Konstruktor implementiert wurde
- reserviert Speicherplatz
- erstellt eine Instanz der Klasse
- weist den Variablen aber keinen Wert zu

Konstruktor:

- muss implementiert werden
- reserviert Speicherplatz
- erstellt eine Instanz der Klasse
- ihm werden zur Initialisierung der Daten des neuen Objekts Argumente übergeben
- Anzahl und Art der Argumente entscheidet, welche Konstruktor-Methode verwendet wird
  -> also können mehrere Konstruktoren existieren

Dateien mit C++

MyClass.hpp: Klassendeklaration mit Implementationsteil

MyClassMain.cpp: Hauptprogramm (#include "*MyClass*.hpp")

oder:

MyClass.h: Klassendeklaration

MyClass.cpp: Klassenimplementation (#include "*MyClass*.h")

MyClassMain.cpp: Hauptprogramm (#include "*MyClass*.h")

Hinweise

```
bool yes_i_am(std::wstring word){
   if (iswlower(word[0])){
      return false;
   } else if(iswupper(word[-1])){
      return false;
   } else if(word.size()>4){
      return false;
   } else return true;
}
```

Hinweise

```
bool yes_i_am(wstring word) {
    int max=5;
    if(word.length()==max) {
        if(iswupper(word.at(0))){
            if(iswlower(word.at(4))){
                return true;
            } else {
                return false;
            }
        } else {
            return false;
        }
    } else {
        return false;
    }
}
```

Programming under Pressure

Programming under Pressure

Schreibt ein Programm, das die UTF-8-Datei "der_Idiot_100.txt" öffnet und zeilenweise einliest. Die einzelnen Wörter sollen dann mit einem Stringstream gesplittet und alle Buchstaben zu Kleinbuchstaben umgewandelt werden, Punktuationszeichen sollen gelöscht werden. Auf der Konsole soll schließlich eine Wortliste mit einem Wort pro Zeile ausgegeben werden.

zB.: Eine kleine Breze!
-> eine
    kleine
    breze

Für Schnelle:
Änderung: Der oder die mittleren Buchstaben sollen großgeschrieben sein. kleine -> klEIne