Lösungen für das 7. Praktikum

1

```
#include <iostream>
using namespace std;

class Laserscanner {
    private:
        double  mAbstand;

    public:
        Laserscanner() {
            mAbstand = 0.0;
        }

        double  getAbstand() { return mAbstand; }

        void    scan() {
            double  zufall = rand()/double(RAND_MAX);
            mAbstand = zufall * 25.0;
        }
};

class Roboter {
    protected:
        int     mId;
        bool    mFehler;
        double  mPos;

    public:
        Roboter(int id) {
            mId     = id;
            mFehler = false;
            mPos    = 0.0;
        }

        int     getId()     { return mId; }
        double  getPos()    { return mPos; }
        bool    getFehler() { return mFehler; }

        void status() {
            cout    << "Roboter #" << mId << ": ";
            if( mFehler )
                cout    << "Roboter angehalten!" << endl;
            else
                cout    << "Alles OK, Roboter arbeitet normal" << endl;

            cout    << "Position: " << mPos << " m" << endl;
        }
};

class Husky : public Roboter {
    private:
        Laserscanner    mScanner;

    public:
        Husky(int id) : Roboter(id), scanner() {
        }
```

```
        void    fahren(double strecke) {
            mScanner.scan();

            if( mScanner.getAbstand() < strecke )
                mFehler = true;
            else
                mPos += strecke;
        }
};

int main()
{
    Husky   husky(12);
    srand(time(NULL));

    while( !husky.getFehler() )
    {
        husky.fahren(0.5);
        husky.status();
    }

    return 0;
}
```

2

```
//helm.h
#pragma once
#include <iostream>
#include <string>
using namespace std;
class Helm
{
       string farbe;
       int festigkeit;
public:
       Helm(string _farbe, int _festigkeit) :
              farbe(_farbe), festigkeit(_festigkeit) {}
       string getFarbe() { return farbe; }
int getFestigkeit(){return festigkeit;}
};

//tiger.h
#pragma once
class Tiger
{
       int registriernummer;
       int schaedelfestigket;
public:
       Tiger(int _registriernummer, int _schaedelfestigket) :
              registriernummer(_registriernummer), schaedelfestigket(_schaedelfestigket)
{}

       int getSchaedelfestigket() { return schaedelfestigket; }
       void ausgabe();
};

//tiger.cpp
#include <iostream>
```

```
using namespace std;
#include "Tiger.h"

void Tiger::ausgabe(void)
{
	cout <<endl<< registriernummer << " " << schaedelfestigket << endl;
}

//helmtiger.h
#pragma once
#include "tiger.h"
#include "Helm.h"
class Helmtiger :
	public Tiger
{
	Helm* helm;
public:
	Helmtiger(int _registriernummer, int _schaedelfestigket, Helm* _helm);
	void ausgabe();
	void setHelm(Helm* _helm);
	int gesamtFestigkeit();
};

//helmtiger.cpp
#include "Helmtiger.h"

Helmtiger::Helmtiger(int _registriernummer, int _schaedelfestigket, Helm* _helm) :
	Tiger(_registriernummer, _schaedelfestigket), helm(NULL)
{
	setHelm(_helm);
}
void Helmtiger::setHelm(Helm* _helm) {
		helm = _helm;
}
void Helmtiger::ausgabe(void)
{
	Tiger::ausgabe();
	if (helm != NULL) cout << helm->getFarbe() << " "<< helm->getFestigkeit() <<
endl;
}
int Helmtiger::gesamtFestigkeit()
{
	if (helm != NULL)
		return Tiger::getSchaedelfestigket() + helm->getFestigkeit();
	else
		return Tiger::getSchaedelfestigket();
}

//main
#include "Helmtiger.h"
int main()
{
	Tiger *t= new Tiger(1, 5);
	t->ausgabe();

	Helm h1("lila", 3), h2("rosa", 3);
	Helmtiger* ht1 = new Helmtiger(2, 6, &h1);
	ht1->ausgabe();
	cout << ht1->gesamtFestigkeit();
```

```
        Helmtiger* ht2 = new Helmtiger(3, 5, &h2);
        ht2->ausgabe();
        cout << ht2->gesamtFestigkeit();

        delete t;
        delete ht1;
        delete ht2;
}
```