# string0_best.cpp - 2. Alternative fuer eine sichere Klasse mit dynamischen C-string Member

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

class zk { // Beispiel fuer korrekte Klasse mit dynamischen C-strings
           // Hier wird s als const-Inhalt und const Zeiger zurueckgegeben. Solange
           // dieser nicht wieder ausserhalb nach (char *) gecastet wird, kann von
           // ausserhalb eines zk-Obj. nicht auf das im Objekt befindliche s zuge-
           // griffen werden. Ein (char *)-cast kann nicht verhindert werden !
      char *s;
   public:
      zk(char *z = 0):s(z ? strcpy(new char[strlen(z)+1], z) : 0){
           cout<<"Konstruktor zk, s = "<<(this->s ? this->s : "0")<<endl;
           cout<<"        Adresse s = "<<(int *)s<<endl;
      }

      // Kopierkonstruktor
      zk(zk &zkd):s(&zkd && zkd.s ? strcpy(new char[strlen(zkd.s)+1], zkd.s):0){
           cout<<"Kopierkonstruktor zk, zkd.s = "<<zkd.s<<endl;
      }

      // Zuweisungsoperator ueberladen
      zk &operator=(const zk &zkd){
          if(&zkd != this){
             delete [] s; s=0;
             s=&zkd && zkd.s ? strcpy(new char[strlen(zkd.s)+1], zkd.s) : 0;
          }
          return *this;
      }

      ~zk(){ cout<<"Destruktor zk,  s = "<<s<<endl; delete [] s; s=0; }

      const char * const get_s(){ return s; }  // s und *s sind const !!
```

```
    void set_s(char *z=0){
        delete [] s; this->s = z?strcpy(new char[strlen(z)+1], z) : 0;
    }

    //Ueberladen des += - Operators (Anketten von zkd.s an s)
    zk &operator+=(const zk &zkd){
       if(s && zkd.s){
          char *t = new char[strlen(s)+strlen(zkd.s)+1];

          strcpy(t, s); strcpy(t+strlen(s), zkd.s);
          delete [] s; s = t;
       }
       if(!s && zkd.s){ s = zkd.s?strcpy(new char[strlen(zkd.s)+1], zkd.s):0; }
       return *this;
    }
};

void main(){
   char *zt = "HTW Dresden";
   char *z = zt?strcpy(new char[strlen(zt)+1], zt):0;
   cout<<"        Adresse z = "<<(int *)z<<endl;

   zk *s1 = new zk(z);                    // Konstruktor

   zk *s2 = new zk(*s1);                  // Kopierkonstruktor
   delete [] z; z=0;

   // char *s04 = s2->get_s();            // Compile-Error

   // char *s03 = (char *)s2->get_s();    // cast nach char *
   // delete [] s03; s03 = 0;
   // eine folgende s2->get_s() - Ausgabe wuerde einen Abbruchfehler verursachen
```

```
    const char * const so5 = s2->get_s();  // OK, nichts aenderbar !!

    char *s02 = s2->get_s()?strcpy(new char[strlen(s2->get_s())+1], s2->get_s()):0;
    cout<<"s2->s = "<<(s02?s02:"0")<<endl;
    delete [] s02; s02 = 0;

    *s2 = *s1;                                  // Zuweisungsoperator
    cout<<"s2->s = "<<(s2->get_s()?s2->get_s():"0")<<endl;

    s1->set_s("TU Dresden");
    cout<<"s1->s = "<<(s1->get_s()?s1->get_s():"0")<<endl;

    *s2+=*s1;
    cout<<"s2->s = "<<(s2->get_s()?s2->get_s():"0")<<endl;

    delete s1; s1 = 0;
    delete s2; s2 = 0;

    cin.get();
}
```

```
        Adresse z = 00355F48
Konstruktor zk, s = HTW Dresden
        Adresse s = 00356068
Kopierkonstruktor zk, zkd.s = HTW Dresden
s2->s = HTW Dresden
s2->s = HTW Dresden
s1->s = TU Dresden
s2->s = HTW DresdenTU Dresden
Destruktor zk,  s = TU Dresden
Destruktor zk,  s = HTW DresdenTU Dresden
```