INH8.CPP - Aufweichen der private-Ableitung

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

class base {
 private:   void privB(){ cout<<"base privB()"<<endl; }
            int i;

 protected: void protB(){ cout<<" base protB()"<<endl; }
            int p;

 public:    base(int k=0):j(k),i(k),p(k){};
            ~base(){};
            void pubB(){ cout<<"base pubB()"<<endl; }
           void pubB(int){ cout<<"base pubB(i), i="<<i<<endl; }
            int j;
};

class derived : private base {
 private: void privD(){ cout<<"derived privD()"<<endl; }

 public:  derived(int d):base(d){};
          ~derived(){};
          base::pubB; //hier als public !!
          base::j;    //hier als public !!
          base::p;
       // base::i;    //nicht moeglich !!
          void pubD(){ cout<<"derived pubD()"<<endl; }
};

void out(base &b){ cout<<"out"<<endl;
                   b.pubB();
                   b.pubB(5);
                //  b.privB(); // kein Zugriff
                //  b.protB(); // kein Zugriff
}

void main(){
        derived d1(1);
        d1.pubB();
        d1.pubB(2);
        cout<<"d1.j = "<<d1.j<<endl;
        cout<<"d1.p = "<<d1.p<<endl;

     // base b5(d1);               // Error, kein Zugriff !
     // base b6=(base)d1;          // Error, kein Zugriff !
     // base   *b1=&d1;            // Error, kein Zugriff !
     // base   &b2=d1;             // Error, kein Zugriff !

        base   *b3=(base *)&d1;    // OK wegen cast (base *)

     // base   &b4=(base)d1;       // Error, kein Zugriff !
     // out(d1);                   // Error, kein Zugriff !

        out((base &)d1);
        cin.get();
}
```