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


class A
{
public:
    A(string const & name)
        : _ss(name)
    {
    }

    string Get_string() const
    {
        return _ss;
    }

//private:              // No access from outside this class
protected:              // Allows access to members by derived classes.
    string _ss;

};

class B : public A          // IS-A  [derived class]; 'public' allows access to public/protected members/methods of class A
{
public:
    B(string const & name, int b)  // Has-A  [additional property wrt. A]
        : A(name), _b(b)
    {
    }

    //string Get_string() const {return _ss;}  // overload, not needed here

    int fkt_B() const
    {
        return _b*_b;   // Has-A  [additional property wrt. A]
    }

```

Konstruktor A

Getter

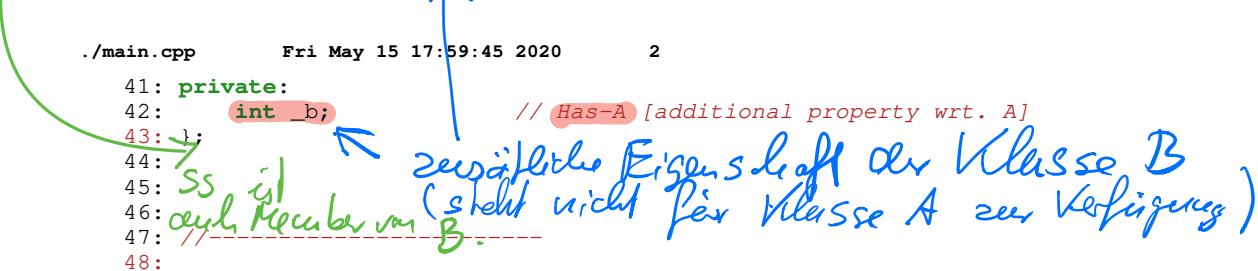
private: ⇒ ss nicht sichtbar für B

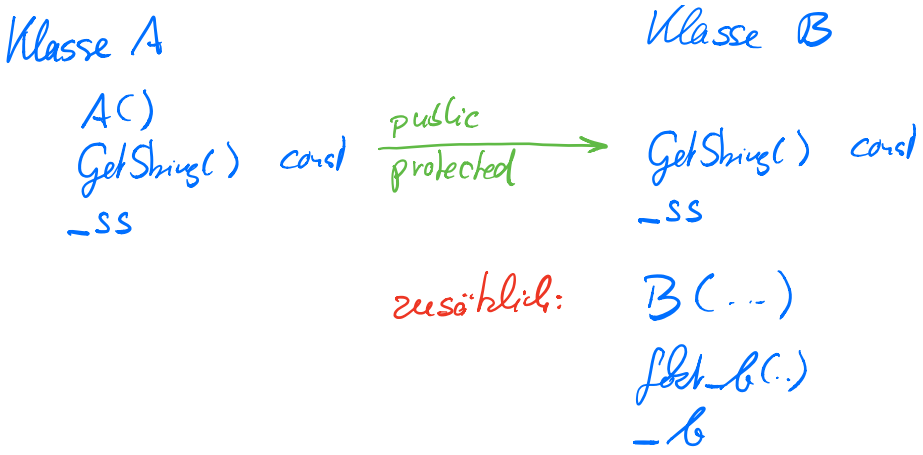
Klasse B ist auch eine Klasse A

Konstruktor Basis Klasse!

auch Methode der Klasse B

```
41: private:
42:     int _b;             // Has-A [additional property wrt. A]
43: };
44:
45:
46:
47: //-----------------------
48:
49: int main()
50: {
51:     A v("Freitag");
52:     cout << v.Get_string() <<  endl;  // A::Get_string
53:
54:     B g("Samstag",7);
55:
56:     cout << g.Get_string() << " ";    // B::Get_string --> A::Get_string
57:     cout << g.fkt_B() << endl;
58:
59:
60:     return 0;
61: }
```




Klasse A
A()
GetString() const
_ss
public
protected
Klasse B
GetString() const
_ss
zusätzlich:
B(...)
fkt_b(..)
_b