→ String anlatmaya devam etti. 1.23`te inheritance`a geçiyor
→ STL kitabına bak. String gelirse

*Inheritance (kalıtım):

- Cpp kalıtım araçları ⊃ Nesne yönelimdeki kalıtım.

→ Nesne yönelimli programlamada kalıtım: • Elimizdeki sınıfın, bir public interface`i var Bu public interface`i kendisine katan yeni bir sınıf oluşturma

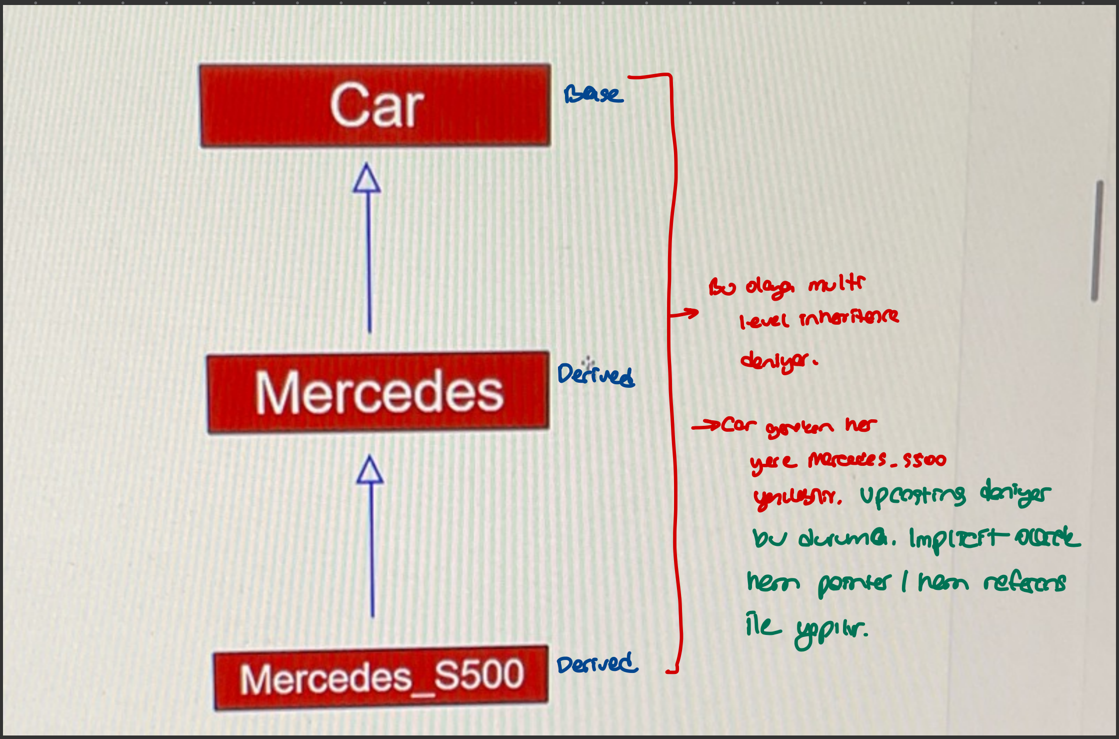
$$\underbrace{\text{• Has a relationship}}_{\downarrow}$$ ve Is a relationship

composition

- Code-reuse ve abstraction sağlar

- Kaynak olarak kullanacağımız ve zaten var olan, kendisinden türetilen sınıf = parent class / super class (işçiden → yazılma)

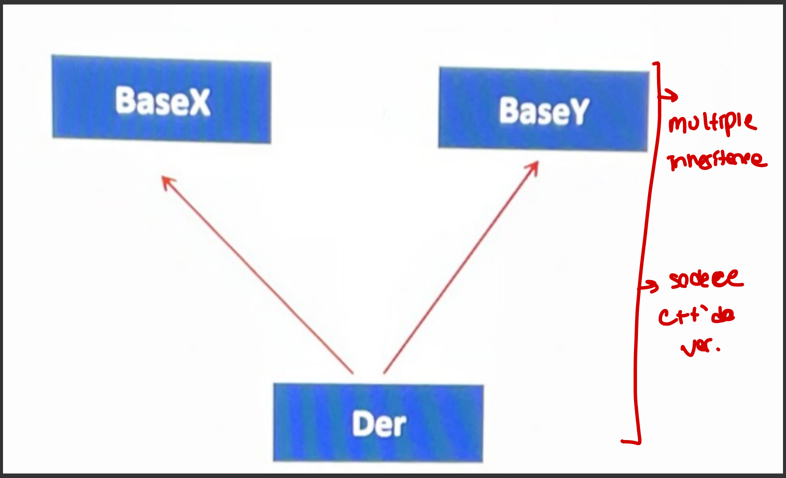
- Oluşturulan class`a child class / sub class



→ Bu olaya multi level inheritance deniyor.

→ Car gereken her yere Mercedes_S500 yollanır. Upcasting deniyor bu duruma. Implicit olarak hem pointer / hem referans ile yapılır.

- Base'den der ürettiğinde bu der de ayrı bir base olabilir.



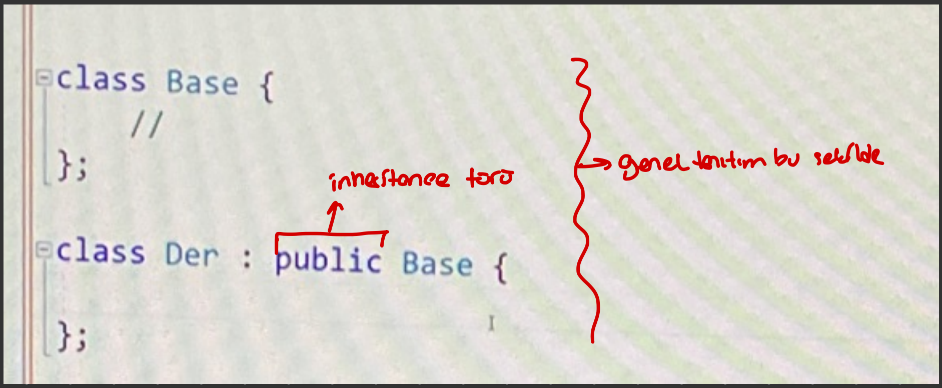
→ multiple inheritance

→ sadece C++`da var.

- Der, kendi interface`ine, hem base x hem base y alır.

• Base class incomplete type olamaz!
↳ bildirim yapıp definition yapılmayan tür.

• Cpp'de 3 adet inheritance vardır. → public inh → object oriented'ta bu
→ private inh
→ protected inh

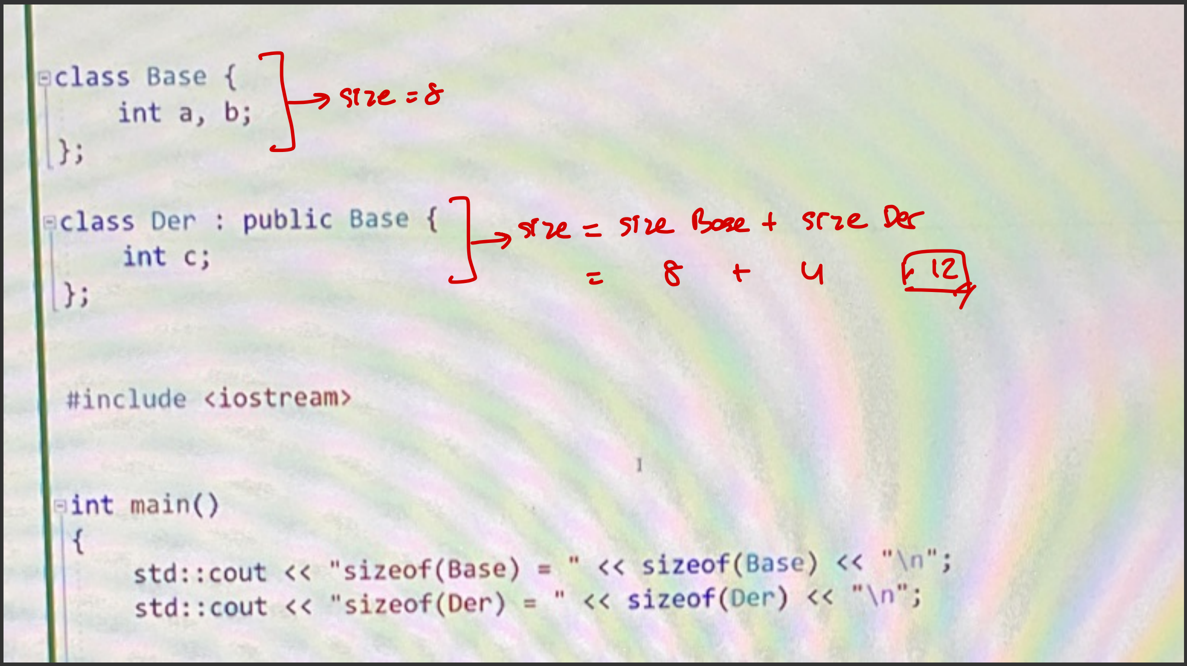


```
class Base {
    //
};

class Der : public Base {

};
```

• Der sınıfı, base sınıfından public kalıtımla elde edilmiş

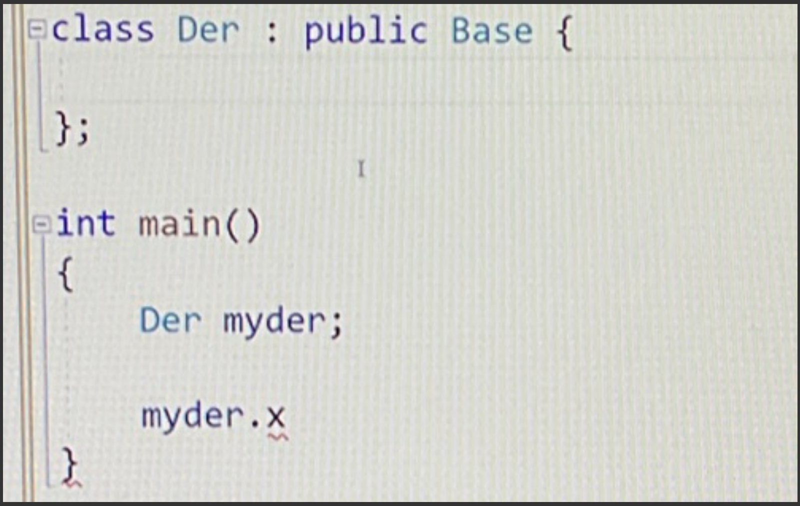
• class Der : Base = class Der : private

• Fakat struct'ta inheritance türü default public



```
class Base {
    int a, b;
};

class Der : public Base {
    int c;
};


#include <iostream>


int main()
{
    std::cout << "sizeof(Base) = " << sizeof(Base) << "\n";
    std::cout << "sizeof(Der) = " << sizeof(Der) << "\n";
```

* Name look-up in Inheritance:

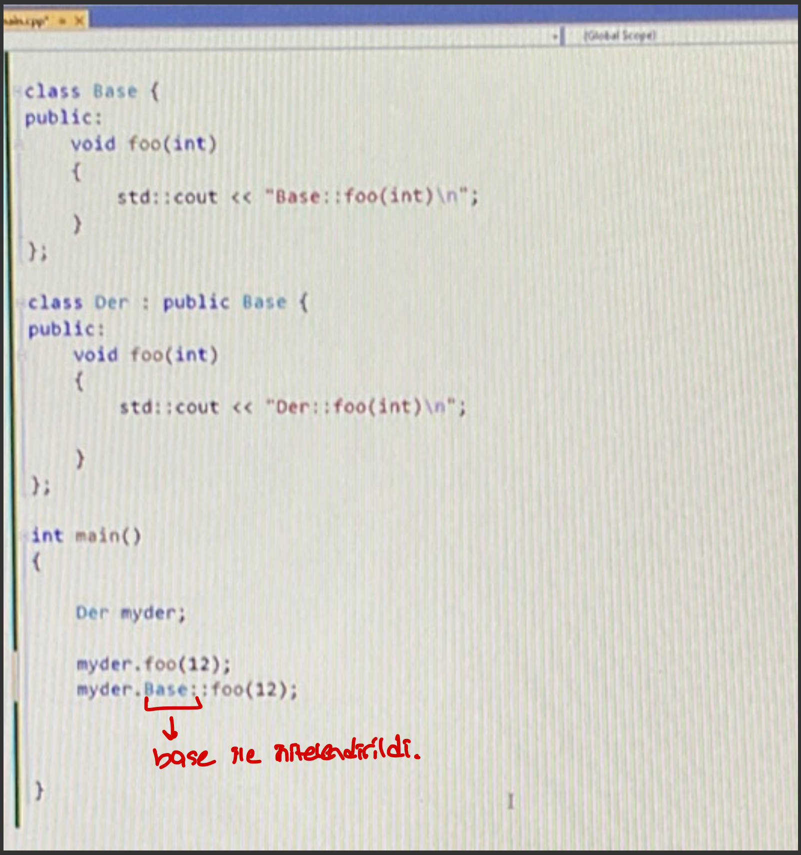


```
class Der : public Base {

};

int main()
{
    Der myder;

    myder.x
}
```

• Name look up önce derived class, eğer orada yoksa sonra base class'a bakar. Eğer ikisinde de varsa derived class, information hiding

• Scope farklı → function overloading'e girmez

↳ information hiding'ten kaçınmak için:



```cpp
class Base {
public:
    void foo(int)
    {
        std::cout << "Base::foo(int)\n";
    }
};

class Der : public Base {
public:
    void foo(int)
    {
        std::cout << "Der::foo(int)\n";

    }
};

int main()
{

    Der myder;

    myder.foo(12);
    myder.Base::foo(12);


}
```

↓ base ile nitelendirildi.

- Base sınıfın private'ı herkese kapalı. Kalıtımla erişilemez!
  Clientlar erişemez!