

→ String anlatmaya devam etti. 1.23'te inheritance'a geçiyor.

→ STL kütüphanesine bak. String gördükse

* Inheritance (Kalıtım):

• Cpp kalıtım araçları > Nesne yönelimli kalıtım.

→ Nesne yönelimli programlamada kalıtım: • Bizimdeki sınıfın, bir public interface'i var. Bu public interface'i kendisine katan yeni bir sınıf oluşturuyor.

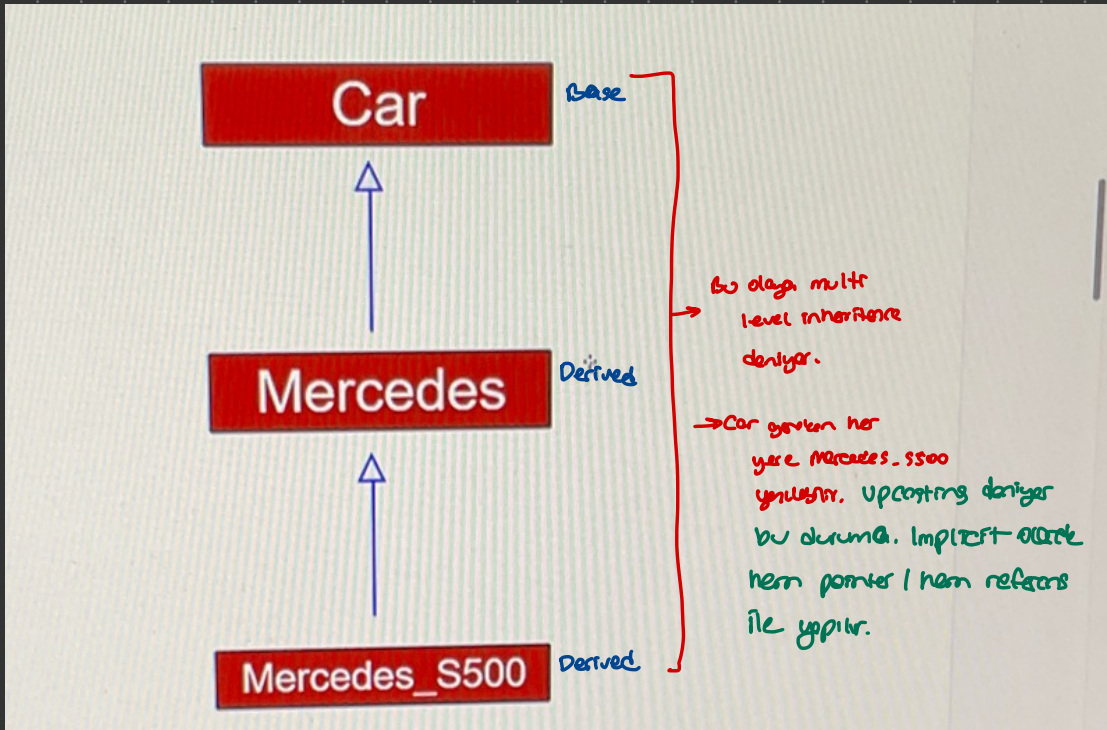
• Has a relationship ve Is a relationship.

↓
Composition

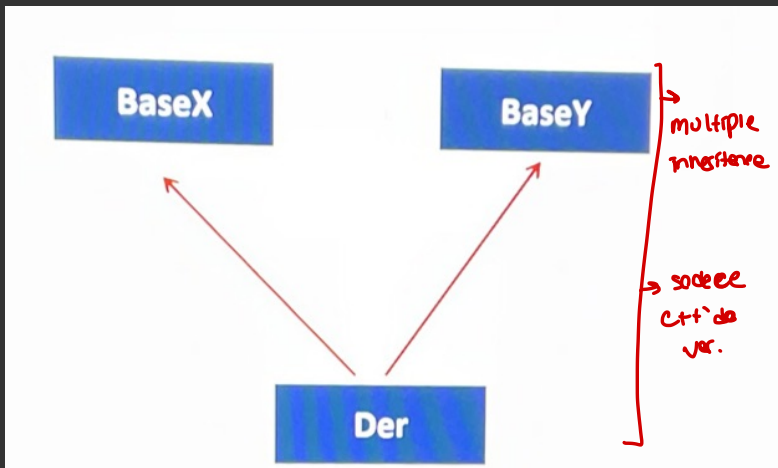
• Code-reuse ve abstraction sağlar

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

• Oluşturulan class'a child class / sub class.



• Base'den der ettiğimizde bu der de aynı bir base olabilir.



• Der, kendi interface'ine, hem base x hem base y alır.

- Base class incomplete type olamaz!

↳ bildirim yapıp
definition yapilmagan tür.

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

```
class Base {
    //
};

class Der : public Base {
    //
};
```

inheritance türü

generalization bu şekilde

- Der sınıfı, base sınıfından, public kalıtımla elde ediliyor

• 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";
}
```

size = 8

size = size Base + size Der
= 8 + 4 = 12

* Name look-up in Inheritance:

```
class Der : public Base {
    //
};

int main()
{
    Der myder;

    myder.x
}
```

- Name look up once derived class, eğer orada varsa sonra base class'a bakar. Eğer ilerinde de varsa derived class, information hiding

- Scope farklı → func overloading'e girmez

↳ methoden hidingten vermeiden:

```
main.cpp - x [Global Scope]

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 nie mehr aufrufen!
}

1
```

- Base sind in **private**, herkömml. Kopie, Konstruktoren erstellen! Clienten erreichen!