→ Dokuz 12'deki örnek move vs copy !!

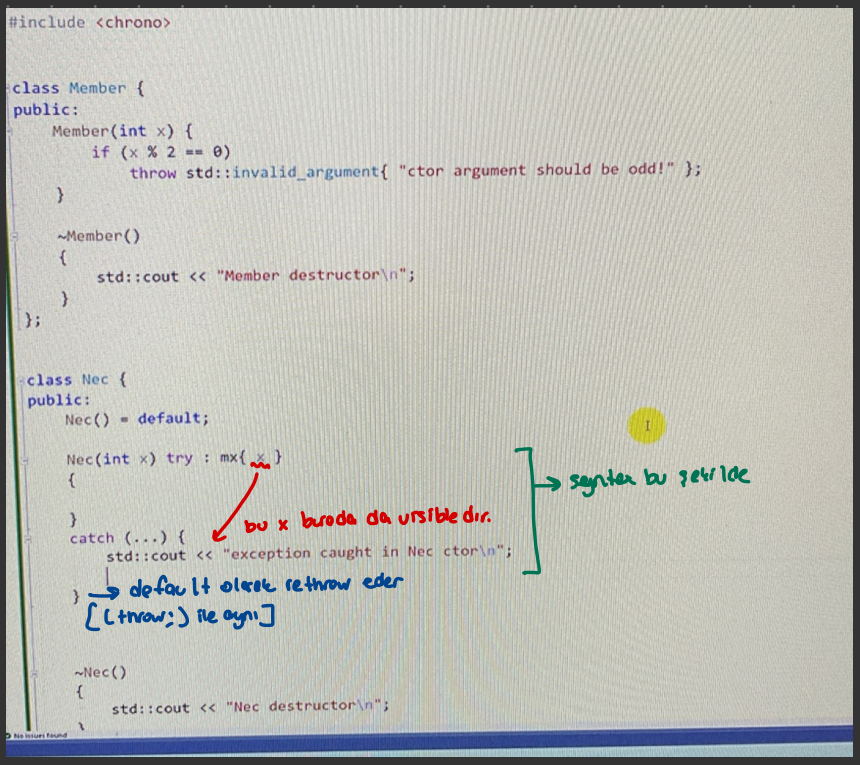
* Function Try Block:

```
void func()
try {
    // all code of func
}
catch (const std::exception& ex) {

}
```

→ Function try block constructor da kullanılır. Çünkü normalde constructorda exception olsaydı, nesne hayata gelmezdi.

→ Constructor içinde exception handle için



→ Private inheritance:

- Taban sınıfın private bölümü, türemiş sınıfa hiç zaman kapalı.

```
public kalıtımı

taban sınıf              türemiş sınıf

public      ====>        public
protected   ====>        protected
private     XXXXXXX
```

- Taban sınıfın public bölümü, türemiş sınıfın private bölümüne dahil.

```cpp
public:
    void basefunc_public();
protected:
    void basefunc_protected();

};

class Der : private Base {
public:
    void f()
    {
        basefunc_public();
        basefunc_protected();
    }
};

int main()
{
    Der myder;
    myder.basefunc_public
         _protected
}
```

Base'den gelenler benim private bölümümde, class içi erişilir ancak clientlar erişemez

myder.basefunc_public / _protected → sentaks hatası

```cpp
class Base {

};

class Der : private Base {

};

int main()
{
    Der myder;

    Base* baseptr = &myder;
    Base& baseref = myder;
}
```

"is A" relationship clientlar için yok.

private inheritance'da artık derived class base ptr, base ref kullanılan yerde kullanılamaz, Implicit conversion yok X

public olsa olurdu.

```cpp
class Base {

};

class Der : private Base {
    friend void g();
    void func()
    {
        Der myder;

        Base* baseptr = &myder;
    }
};

void g()
{
    Der myder;

    Base* baseptr = &myder;
}

int main()
{
```

Fakat "is a" relationship'i member functionlar ve friend declerationlar ile "class İÇİNDE" sağlayabiliriz.

a) Türemiş sınıf nesnesi içinde taban sınıf nesnesi vAR
B) türemiş sınıfın implementasyonu taban sınıfın public ve taban sınıfın protected işlevlerini kullanabiliyor
c) taban sın ıfın private bölümüne erişemiyortuz

taban sınmıfın public ve taban sınıfın protected öğeleri türemiş sınıfın private bölümüne ekleniyor
türemiş sınıfın client'ları taban sınıfın public öğelerini görmeyecekler

private inheritance (çok büyük çoğunlukla) containment yoluyla composition'a bir alrternatif!!!

is a relationship yok

1) private kalıtımında taban sınıfın sdanal fonksiyonlarını override edebiliyoruz!!!
2) private kalıtımında taban sınıfın protected fonksiyonlarını çağırabiliriz!!!
3) Türemiş sınıfın üye fonksiyonlarında ve türemiş sınıfın friend'lik verdiği fonksiyonlarda is-a relationship söz konusu

Containment da yok!

* Containment By Composition Örneği:

```
class A {

public:
    void f1();
    void f2();
};

class Nec1 {

private:
    A ax;
};

class Nec2 : private A {

};
```

Bu iki gösterim aynı anlama geliyor.

- Her ikisinin, public bölümü, bizim private bölümümde kendi public interface'imize, aynı isimle bile, farklı işlevler tanımlayıp çağırtabiliriz.

- Alternatif olarak "using Base::fonksiyon" ile bizim public kısmımıza enjekte edebiliriz.

- Protected Inheritance:

- taban sınıfın public bölümü, türemiş sınıfın protected bölümüne eklenir.

* Templates:

→ Derleyiciye kod yaptırmak = Template Code

```
function template  (fonksiyon şablonu)
class template  (sınıf şablonu)
variable template
alias template (tür eş isim şablonu)

variadic template
```

→ modern Cpp örneği

→ modern Cpp ile geldi.

```
template <class T>

template <typename T>
```

Aynı anlama geliyor.

a) type parameter (template tür parametresi)
b) non-type parameter (template sabit parametresi)

c) template parameter (template parametresi)