2023 08 16

Constructor

```
  Constructor
    1) sınıf isimiyle aynı olamak zorunda
    2) non-static member function olmak zorunda
    3) ctor free function olamaz
    4) ctor static member function olamaz
    5) const member function olamaz
    6) geri dönüş değeri kavramına sahip değildir
    7) overload edilebilir.
    8) public, private ve protected olabilir ama diğerleri gibi erişim sıkıntı
olabilir
    9) nokta ya da ok operatörüyle çağrılamıyor

    default ctor: parametresi olmaya ya da tüm parametreleri varsılan arguman alan
ctor

    class MyClass{
        MyClass(int x = 6) // default ctor
    };
```

Destructor

```
    Destructor
    Bir sınıf nesnesinin lifespan bitmesini sağlan hayatını sonlandıran fonks
    1) non-static member function olmak zorunda
    2) dtor free function olamaz
    3) dtor static member function olamaz
    4) const member function olamaz
    5) geri dönüş değeri kavramına sahip değildir
    6) parametre değişkeni olmıcak
```

Special Member Functions:

```
     special member functions:
        default ctor
        destructor
        copy ctor
        move ctor (C++11)
        copy assignment
        move assignment (C++11)

special member denmesinin nedeni bu fonksiyonların kodları (belirli koşullar
altında)
derleyici tarafından bizim için yazılabilmesi
```

Global Sınıf Nesneleri

```
class Myclass;
// aynı kaynak dosyasında olduğu için ctor çağrılma sırası tanımlanma sırasına
göre
Myclass g;
Nec g_nec;
// main fonksiyonu çağrılmadan g nesnesinin ctor çağrılır ve main bittikten sonra
dtor çağrılır
int main()
{

}
```

Aynı programın farklı kaynak dosyalarında tanımlanan global sınıf nesnelerinin tanımlanan global sınıf nesnelerin ctor'larının çağrılma sırası dil tarafından belirlenmiş değildir.

```
class Myclass;
void foo()
{
    static Myclass m; //static storage class
}
int main()
{
        foo(); foo(); foo(); // foo function 3 defa çağrılmasına rağmen m objesi
        //bi kere oluşacak ve main tamlandıktan sonra yokolacak.
        //1
        {
            //2
            Myclass m;  // önce 1 olacak sonra 2  sonra m objesinin ctor oluşacak
                        //sonra 3 ve m objesnin dtor olacak ve 4 olacak
            //3
        }
        //4
}
```

Ctor Initializer List

```
//önce data memberlar( tx,ux, mx) meydanda gelir daha sonra ctor çağırılır.
class Myclass{
    public:
        Myclass() : my(10) , mx(my /3) // burda unbehaviour oluyor çünkü
                    //mx ilk init edilir my bu sırada garabed value oluyor
        {

        }
    private:
        T tx;
        U ux;
        W mx;

        int mx,my;
};
```

```cpp
class Myclass{

public:
    Myclass(int &r) : mx{20}, mr(r)
    {


    }
private:
    // referans ve const'lar default init edilemez
    const int mx; // eğer value init yapmazsak yukarıdaki gibi

    int &mr; // syntax hata dönerdi
}
```

```cpp
class Person {
    public:
        Person(const char* p)
        {
            /*
                böyle yaparsak önce default ctor çağırılır
                daha sonra copy assignment çağrılır
            */
            m_address = p;
        }
    private:
        std::string m_address;
}
```

```cpp
class myclass{
myclass() : mx(46) {};
private:
    int mx = 10;
    int my = 20;

    /*
        in class initializer cpp11
        default member initializer 10 ve 20' yi derleyici default init edicek
        biz ne yapması gerektiğii söylüyoruz init eden derleyici

    */
};
```