BTEP243 – Ders 4

Statik Veri Üye ve Metotlar

Genel olarak bir sınıfa ait nesnelerin verileri bellekte farklı bölgelerde yer alır. Ancak bazı durumlarda, belirli bir üyenin ortak bir alanda tek bir kopyasının bulunmasını isteyebiliriz. Bu durumda static anahtar sözcüğünden yararlanıyoruz. Static tanımlı üyeler public yada private tanımlı olabilirler. Static tanımlı üyeler o sınıftan hiçbir nesne var olmasa bile bellekte yer alırlar. Statik üyeler, sınıfın herhangi bir nesnesine ait değildirler. Statik üyelerinin tipik kullanımı bir sınıfın tüm nesneleri için ortak veri kayıdı tutmak içindir.

Örneğin, belirli nesnelerin sayısını saklamak için bir sayaç olarak statik veri üyesi kullanabilirsiniz. Yeni bir nesne oluşturulduğunda her zaman bu statik veri üyesi toplam nesne sayısını tutmak için arttırılmalıdır. Erişim Operatörü **(::)** ve sınıf adı kullanılarak statik üyelere erişilebilir.

Örnek:

```
class x {
private:
static int x;
};

int x :: i = 0;
void main( )
{
...
cout << x :: i;
...
}
```

Statik veri üyeleri sınıf dosyası içerisinde, ancak sınıf bildirimi dışında başlatılmalıdır.

Statik üye fonksiyonları

Bir fonksiyon elemanı statik olarak tanımlanmışsa, sınıfın herhangi bir nesnesinden bağımsız olarak hareket eder. Bir statik üye fonksiyonu, sınıfın herhangi bir nesnesi bulunmuyorsa dahi erişim operatörü kullanılarak erişilme özelliğine sahiptir. Bir statik veri yalnızca statik fonksiyon tarafından erişilebilir.

Örnek 1:

//kutu.h

```
class kutu {
private:

        static int kutusayac;
        double uzunluk, genislik, yukseklik;

public:

kutu (double  u = 2.0, double g = 2.0, double y = 2.0)
{
        cout << << endl "Varsayılan yapıcı çalışıyor.";
        uzunluk = u;
        genislik = g;
```

```
        yukseklik = y;
        kutusayac ++;
}
double hacim ()
{
        return uzunluk * genislik * yukseklik;
}
static int Getkutusayac ()
{
        return kutusayac;
}
};
```
//statik üyelere başlangıç değeri atama işlemi mutlaka sınıf sonlandırıldıktan sonra yapılmalıdır.
```
int kutu :: kutusayac = 0;
```

//kutu.cpp
```
#include <iostream>
using namespace std;
#include "kutu.h"
void main()
{
        cout << "İlk aşamadaki nesne sayısı:" << kutu :: Getkutusayac () << endl;
        kutu kutu1 (3.3,1.2,1.5);
        kutu kutu2 (8.5,6.0,2.0);
        cout << "Kutu1 nesnesinin hacmi:" << kutu1.hacim () << endl;
        cout << "Kutu2 nesnesinin hacmi:" << kutu2.hacim () << endl;
        cout << "Son aşamadaki nesne sayısı:" << kutu :: Getkutusayac () << endl;
        system ( "pause");
}
```

Getkutusayac () static int üye işlevi kullanılmadığı taktirde, sınıfın kapsamı dışından erişmek için (public) statik veri üyesi oluşturmanız gerekir.

public:
static int kutusayac;
....
main()
{
...
cout << " Son aşamadaki nesne sayısı:" << kutu :: kutusayac << endl;

....
}

Örnek 2:

Aşağıdaki örnekte değişken *"toplamsayi"* statik değişken olarak tanımlanır. Bu değişken, sınıfa ait tüm nesneler için geçerli olacaktır. Diğer bir deyişle, sınıftaki tüm nesneler için genel bir kullanım alanına sahiptir. Eğer sıradan sayı tamsayı (statik değil) olarak tanımlanırsa her nesne için ayrı ayrı kullanılabilme özelliğine sahip olabilir. Ancak örneğimizde *"toplamsayi"* statik üyesi, oluşturulacak her nesne için ortak olarak kullanılacaktır.

Çözüm 1

//arabagalerisi.h

```
class arabagalerisi {
int sayac;
string marka;
static int toplamaraba; // statik veri üyesi
public:
arabagalerisi (string m)
{
        marka=m;
        cout << marka<< “ marka arabaların sayısını giriniz:";
        cin >> sayac;
        toplamaraba + = sayac;
}
void arabasayigöster ()
{
        cout<< marka << " marka arabaların sayısı =" << sayac << endl;
}
static int gettoplamsayi ()
{
        return toplamaraba;
}
void arabaal (int n)
{
        sayac+ =n;
        toplamaraba + = n;
        cout << "\ n" << n << "" << marka << " marka araba satın alındı.";
}
void arabasat (int n)
{
        if (n> sayac)
                cout <<”Galeride yeterince “<< marka << " marka araba bulunmamaktadır...."<<endl;
        else
        {
                sayac- = n;
                toplamaraba - = n;
                cout << "\ n" << n << "" << marka << " marka araba satılmıştır.";
        }
}
};
```

int arabagalerisi :: toplamaraba; **//varsayılan değer 0 (sıfır) olacaktır.**

// int arabagalerisi :: toplamaraba = 10; toplam araba sayısı 10’dan başlayacaktır.

//arabagalerisi.cpp

```
#include <iostream>
#include <string>
using namespace std;
#include "arabagalerisi.h"
void main( )
{
        arabagalerisi mersedes( "Mercedes"), toyota( "Toyota"), bmw( "BMW");
        mercedes.arabaal(10);
        toyota.arabasat(7);
        bmw.arabaal(15);
        cout << "\ n **********************************************”<<endl;
        mercedes.arabasayigoster ();
        toyota.arabasayigoster ();
        bmw.arabasayigoster ();
        cout << "Galerideki toplam araba sayısı:" << arabagaleri::gettoplamsayi ( ) << endl;
        system ( "pause");
}
```

HARİCİ (EXTERNAL) VE STATİK NESNELER

Otomatik Nesneler: Ana fonksiyon kapsamı içerisinde oluşturulan nesnelerdir. Sadece yaratıldıkları fonksiyon içerisinde geçerlidirler, fonksiyon kapsamı dışında yaşayamazlar.

Statik Nesneler: Yerel fonksiyon kapsamında oluşturulurlar ancak program sonuna kadar geçerliliklerini sürdürürler.

Harici Nesneler: Herhangi bir fonksiyon kapsamının dışında oluşturulan nesnelerdir.

Örnek

```
//ornek.h
class abc {
int sayi;
public:
abc ()
{value = 0; }
void sayiekle (int s)
{
sayi + = s;
}
int getsayi ()
{
return sayi;
}
};
//harici nesne
abc haricinesne;
```

```
#include <iostream>
using namespace std;
#include"ornek.h"
void main( )
{
haricinesne. sayiekle (5);
for(int i; i <3 i ++)
```

```
{
	// OTOMATİK NESNE
	abc otonesne;
	otonesne.sayiekle (10);
	// STATİK NESNE
	static abc statnesne;
	statobj. sayiekle (15);
	cout << "\n \nOtomatik nesne değeri =" << otonesne.getsayi ();
	cout << "\n \nStatik nesne değeri=  "<< statnesne.getsayi () ;
}
cout << "\n\nHarici nesne değeri="<< haricinesne.getsayi ();
system( "pause");

}
```

Örnek2 - Çözüm 2

//arabagalerisi2.h

```
class arabagalerisi {
int sayac;
string marka;
static int toplamaraba; // statik veri üyesi
public:
arabagalerisi (string m)
{
	marka=m;
	cout << marka<< " marka arabaların sayısını giriniz:";
	cin >> sayac;
	toplamaraba + = sayac;
}
void arabasayigöster ()
{
	cout<< marka << " marka arabaların sayısı =" << sayac << endl;
}
static int gettoplamsayi ()
{
	return toplamaraba;
}
void arabaal (int n)
{
	sayac+ =n;
	toplamaraba + = n;
	cout << "\ n" << n << "" << marka << " marka araba satın alındı.";
}
void arabasat (int n)
{
	if (n> sayac)
		cout <<"Galeride yeterince "<< marka << " marka araba bulunmamaktadır...."<<endl;
	else
	{
		sayac- = n;
		toplamaraba - = n;
		cout << "\ n" << n << "" << marka << " marka araba satılmıştır.";
	}
```

```
}
};

int arabagalerisi :: toplamaraba;
// arabagalerisi için harici nesneler
arabagalerisi mercedes ( "Mercedes"), toyota ( "Toyota"), bmw ( "BMW");
```

//cargallery2.cpp

```
#include <iostream>
#include <string>
using namespace std;
#include "arabagalerisi2.h"
void stok( );
void main( )
{
        mercedes.arabaal (10);
        toyota.arabasat (7);
        bmw.arabaal (15);
        stok();
        system( "pause");
}
void stok( )
{
        cout << "\ n ********************************** \nOtomobillerin Sayisi: ";
        mercedes. arabasayigöster ();
        toyota. arabasayigöster ();
        bmw. arabasayigöster ();
        cout << " \ nToplam araba sayısı:" << arabagalerisi :: gettoplamsayi () << endl;
}
```

// Çözüm 3 - Parametreli yapıcıyı çalıştıracak bir dizi nesnesi oluşturmak

```
#include <iostream>
#include <string>
using namespace std;
#include "arabagalerisi.h"
void main
{
        arabagalerisi galeri [3] = { "Honda", "Hyundai", "Fiat"};
        galeri[0].arabaal (5);
        galeri[2].arabasat(4);
        cout << "\ n ********************************** \nOtomobillerin Sayisi: ";
        for (int i = 0; i<3 ; i ++)
                galeri [i].arabasayigöster ();
        cout << " \ nToplam araba sayısı:" << arabagalerisi :: gettoplamsayi () << endl;
        system( "pause");
}
```

***"this"* İşaretçisi (this pointer)**

this işaretçisi, statik eleman fonksiyonlar hariç bütün eleman fonksiyonların erişebildiği özel bir işaretçidir. **this** işaretçisi, eleman fonksiyonu çağıran nesneyi işaret eder.
Yani, bir eleman fonksiyonunu bir nesne için çağırdığımız zaman, derleyicimiz önce oluşturduğumuz nesnenin adresini this göstergesine aktarır ve ardından fonksiyonu çağırır.

this işaretçisinin diğer bir kullanımı ise, eleman fonksiyonu çağıran nesne ile bu fonksiyona parametre olarak gönderilen nesnenin aynı olup olmadığını kontrol eder.

Örnek:

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

class Film
{
	string film_adi, oyuncu_adi;
public:
	Film(string f, string o)
	{
		this->film_adi=f;
		this->oyuncu_adi=o;
	}
	string yaz()
	{
		return this->film_adi+this->oyuncu_adi;
	}
	void goster()
	{
		string a;
		a=this->yaz(); //a=yaz();
		cout<<a;
	}
};
void main()
{
	Film nesne("Titanic"," Leonardo Di Caprio");
	nesne.goster();
	system("pause");
}
```

SABİT *(const)* ÜYE FONKSİYONLARI

const, değiştirilemez nesneler tanımlamak için kullanılır. **const** ile tanımlı bir nesnenin herhangi bir üyesini değiştirmeye çalıştırmak hata oluşturacaktır. Bazen bir sınıfın belirli bazı üyelerinin değiştirilmesini önlemek isteriz. Bu durumda ilgili üye fonksiyonun sonuna **const** belirteci konur.

Örnek 1:

```
class constOrnek
{
        int x;
public:
        constOrnek (int a)
        {
            x = a;
        }
        void setx (int);
        const int getx ();
};
int constOrnek:: getx () const
{
        return x;
}
void constOrnek::setx (int x)
{
        this->x=x;
}

#include <iostream>
using namespace std;
void main()
{
        const constOrnek test1;
        constOrnek test2;
        test1.setx (5); // nesne const, ancak fonksiyon const olmaığından HATA oluşur!!!
        cout << test1. getx () << endl; // nesne ve fonksiyon const olduğundan herhangi bir hata oluşmaz.
        test2. setx (5); // nesne ve fonksiyon const olmadığından herhangi bir hata oluşmaz
        cout << test2. getx () << endl; //const fonksiyonlar const olmayan nesneler tarafından ulaşılabilir.
        system ( "pause");
}
```