

13 Çokbiçimlilik (Polymorphism)

Nesne yönelimli programlamada üç önemli kavram bulunmaktadır. Bunlar sırası ile Sınıflar, Kalıtım ve Çokbiçimlilik dir. Çokbiçimlilik sanal(virtual) fonksiyonlarla gerçekleştirilir. Sanal fonksiyonların da temel sınıfa işaret eden işaretçilerle yakın ilişkisi vardır.

İşaretçiler ve türemiş sınıflar

Temel sınıf türünden bir işaretçi, türemiş sınıf türünden bir nesnenin adresini tutabilir. Aşağıdaki örnekte P1, P2 işaretçilerinin Cokgen türünden olduğuna ve sırası ile Dortgen ve Uçgen türünden nesnelerin adreslerini tuttuklarına dikkat ediniz. Ayrıca Hem temel sınıfta ve hem de türemiş sınıfta tanımlanmış **int Alan()** fonksiyonu yer almaktadır. Bu fonksiyon çağrıldığında hangi Alan fonksiyonu çağrılmış olacaktır. Eğer işaretçi kullanılmamış ise çağırılan nesne hangi sınıftan ise o sınıfın Alan fonksiyonu işlem görecektir. Aşağıdaki örnekte

D1.Alan(); ifadesinde

D1 dortgen türünden olduğu için D1 nesnesinin Alan fonksiyonu çağrılacaktır. Yani Türemiş sınıfın Alan isimli fonksiyonu temel sınıfın aynı isimli fonksiyonunu örtecektir(Gizleyecektir).

Aşağıda verilen ifadelerde P1 ve P2 Cokgen (temel sınıf) türünden işaretçiler olup, türemiş sınıflar türünden tanımlanmış D1 ve U1 nesnelerinin adresini tutmaktadır.

Dortgen D1;

Ucgen U1;

Cokgen *P1=&D1;

Cokgen *P2=&U1;

P1->Alan(); P2->Alan() ; ifadeleri de temel sınıfın Alan fonksiyonunu çağıracaktır. Bunun sebebi P1 ve P2 nin temel sınıf türünden olmasıdır ve Çalışma anında (run time) sadece P1 ve P2 işaretçilerinin türüne bakılması içerdiği yani nasıl bir nesnenin adresinin tutulduğuna bakılmamasıdır. Amacımız türemiş sınıfın kendi Alan fonksiyonlarını çağırması olduğuna göre bu nasıl sağlanacaktır?

```
#include <iostream>
```

```
#include <conio.h>
```

```
using namespace std;
```

```
class Cokgen {
```

```
public:
```

```
int genislik, yukseklik;
```

```
public:
```

```
void DegerVer(int a,int b) {
```

```
    genislik=a; yukseklik=b;
```

```
}
```

```
int Alan() {return 0;}
```

```
};
```

```
class Dortgen : public Cokgen{ //Dortgen sınıfı Cokgen sınıfından turetiliyor
```

```
public:
```

```
int Alan () {return genislik*yukseklik;}
```

```
};
```

```
class Ucgen : public Cokgen{ //Ucgen sınıfı Cokgen sınıfından turetiliyor
public:
    int Alan () {return (genislik*yukseklik)/2;}
};
```

```
int main ()
{
    Dortgen D1;
    Ucgen U1;

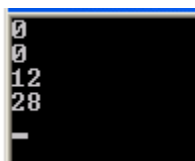
    //P1, P2 işaretçilerinin Cokgen türünden olduğuna ve
    // sırası ile Dortgen, Uçgen türünden nesnelerin
    //adreslerini tutuklarına dikkat ediniz.
    Cokgen *P1=&D1;
    Cokgen *P2=&U1;

    P1->DegerVer(3,4);
    P2->DegerVer(7,8);

    cout << P1->Alan()<<endl; //temel sınıfın Alan fonk. çağrılır
    cout << P2->Alan()<<endl; //temel sınıfın Alan fonk. çağrılır

    cout << D1.Alan()<<endl; //türemiş sınıfın Alan fonk. Çağrılıyor.
    cout << U1.Alan()<<endl; //Alan fonk. Temel sınıfı örtüyor.

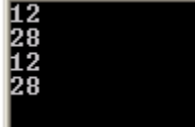
    getch();
    return 0;
}
```



Türemiş sınıfın kendi Alan fonksiyonunu çağırmasını sağlamak için temel sınıfta yer alan Alan fonksiyonunun başına aşağıda görüldüğü gibi **virtual(sanal)** kelimesi getirilmesi gerekmektedir.

```
class Cokgen {
public:
    int genislik, yukseklik;
public:
    void DegerVer(int a,int b) {
        genislik=a; yukseklik=b;
    }
    virtual int Alan() {return 0;}
};
```

Bu takdirde yukardaki programın ekran çıktısı şu şekildedir.



```
12
28
12
28
```

Başka bir örnek

```
#include <iostream>
#include <conio.h>
using namespace std;

class Cokgen {
public:
    int genislik, yukseklik;
public:
    void DegerVer(int a,int b) {
        genislik=a; yukseklik=b;
    }

    virtual int Alan() {return genislik*yukseklik/3;}
};

class Dortgen : public Cokgen{ //Dortgen sınıfı Cokgen sınıfından turetiliyor
public:
    int Alan () {return genislik*yukseklik;}
};

class Ucgen : public Cokgen{ //Ucgen sınıfı Cokgen sınıfından turetiliyor
public:
    int Alan () {return (genislik*yukseklik)/2;}
};

int main ()
{
    Dortgen D1;
    Ucgen U1;
    Cokgen C1;

    Dortgen *P1=&D1;
    Ucgen *P2=&U1;
    Cokgen *P3=&C1;

    Cokgen *PP1=&D1;
    Cokgen *PP2=&U1;
    Cokgen *PP3=&C1;

    D1.DegerVer(3,4);
    U1.DegerVer(7,8);
```

```
C1.DegerVer(2,6);
```

```
P1->DegerVer(3,4);
```

```
P2->DegerVer(7,8);
```

```
P3->DegerVer(2,6);
```

```
PP1->DegerVer(3,4);
```

```
PP2->DegerVer(7,8);
```

```
PP3->DegerVer(2,6);
```

```
cout << D1.Alan()<<endl;
```

```
cout << U1.Alan()<<endl;
```

```
cout << C1.Alan()<<endl;
```

```
cout << P1->Alan()<<endl;
```

```
cout << P2->Alan()<<endl;
```

```
cout << P3->Alan()<<endl;
```

```
cout << PP1->Alan()<<endl;
```

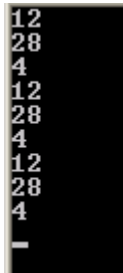
```
cout << PP2->Alan()<<endl;
```

```
cout << PP3->Alan()<<endl;
```

```
getch();
```

```
return 0;
```

```
}
```



```
12
28
4
12
28
4
12
28
4
_
```