

8. Sınıflar

Bir nesne tanımlamak için kullanılan Sınıflar, Nesne yönelimli programlamanın en önemli parçasını oluştururlar. Sınıflar(Classes) veri ve fonksiyonları bir arada bulunduran bir yapıdır. Veri ve fonksiyonların bir arada bulunmasına paketleme(Encapsulation) denilmektedir.

Sınıf içinde tanımlanan veri ve fonksiyonlara sınıfın üyeleri (member) denir. Gerçek dünyada nesneleri tanımlarken nesnenin özelliklerinden ve davranışlarından bahsedilir. Örneğin insandan bahsederken adı, saç rengi, ten rengi, boyu , kilosı gibi özellikleri yanı sıra, konuşması, yürümesi, spor yapması gibi davranışlarından da bahsedilir. Nesne dayalı programlamada ise insan nesnesi tanımlanırken adı, saç rengi, ten rengi, boyu , kilosı çeşitli veri tipleri ile tanımlanabilir.

Örnek:

string adi; int tenrengi; int boy;

Bu veriler insan nesnesinin özelliklerini ifade eder. Bir başka ifade ile Nesnelere ait özellikler sınıf tanımlamaları içinde verilerle ifade edilirler.

İnsanın yürüme, konuşma ve spor yapma gibi davranışları vardır. Bu davranışları da **Yuru()**, **Konuş()**, **Sporyap()** gibi fonksiyonlar yazarak ifade edebiliriz. Sınıf tanımları içinde yapılan fonksiyonlar nesnenin davranışlarını ifade eder. Sınıf Fonksiyonlarına Sınıfın **metodları** da denir.

Bir sınıf bildirimi(declaration) şu şekilde yapılır.

Class Sınıf_ismi {

Erişim belirleyici :

Değişkenler;

Fonksiyonlar;

{

Erişim belirleyicileri **public**, **private**, **protected** terimleri ile belirtilir. Erişim belirleyicileri Sınıf içinde tanımlanan değişken ve fonksiyonlara nasıl erişilebileceğini belirler. Bir başka ifade ile değişken ve fonksiyonların kullanımını sınırlandırır, erişim denetimi sağlar.

Public: Sınıf içindeki değişkenler ve fonksiyonlar bu sınıf içinden kullanılabileceği gibi dışardan da erişilip kullanılabilir.

Private: Sınıfın üye verilerine sadece o sınıfın üye fonksiyonları ve o sınıfın arkadaşları(friend) sınıfları erişip kullanabilir. Sınıfın üyeleri dışardan erişime kapalıdır.

Protected: Sınıfın kendi üyeleri, arkadaş sınıflar ve bu sınıftan türetilen nesneler sınıfın üyelerine erişebilir.

Herhangi bir belirleyici kullanılmamış ise varsayılan olarak sınıfın üyeleri **private** dir. Aşağıda Dortgen isimli sınıf bildirimi yapılmıştır. x ve y değişkenleri **private** dir. DegerVer ve Alan isimli fonksiyonlar ise **public** dir.

```
class Dortgen {  
    int x, y;           //üye private değişkenler  
public:  
    void DegerVer(int,int); //Bu iki üye fonksiyon public dir.  
    int Alan (void);  
};
```

Bu sınıfın Fonksiyonları(**Metodları**)ise şu şekilde tanımlanır. Sınıf ismi ile sınıfa ait fonksiyon ismi arasına “::” operatörü kullanılır. Bu operatör daha önce farklı amaçlar için kullanılmıştı.

```
void Dortgen::DegerVer(int a, int b) {  
    x = a;  
    y = b;  
}
```

Sınıfların bir kez bildirimleri yapıldıktan sonra, bu sınıftan istenildiği kadar nesne tanımlanabilir.

Örnek:

Dortgen D1,D2;

İfadesinde D1 ve D2 isimli iki tane **Dortgen** sınıfından nesne tanımlanmıştır. Sınıflar kullanılarak tanımlanan her nesne için üye değişkenler için bellekte ayrı yer ayrılırken, fonksiyonlar (**Metodlar**)ortak kullanılır. Başka bir ifade ile D1 nesnesinin x ve y değişkenleri ayrı, D2 nesnesinin x ve y değişkenleri için ayrı bellek alanları kullanılır.

Aşağıdaki ifadelerde D1 ve D2 nesnelerinin x ve y değişkenlerine ayrı ayrı dışardan değer atanmaktadır.

```
D1.x=5;  
D1.y=6;  
D2.x=10;  
D2.y=4;
```

Burada dikkat edilmesi gereken nokta bu şekilde nesnelerin değişkenlerine(verilerine) dışardan değer atayabilmek için x ve y değişkenlerinin public olarak tanımlanması gerekir. X ve y private olarak tanımlı olduklarından yukarıdaki kullanımda derleyici hata mesajı verecektir.

```

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

class Dortgen {
    int x, y;
public:
    void DegerVer(int,int);
    int Alan (void);
};

void Dortgen::DegerVer(int a, int b) {
    x = a;
    y = b;
}

int Dortgen::Alan (void) {
    return x*y;
}

int main ()
{
    Dortgen D1,D2;

    //aşağıdaki iki satırı daha sonra siliniz
    D1.x=7;    //x private olduğundan hata mesajı verecektir
    D1.y=8;    //y private olduğundan hata mesajı verecektir

    D1.DegerVer(3,4); //D1 nesnesinin x ine 3 ve y sine 4 değeri atanır.
    D2.DegerVer(7,2); //D2 nesnesinin x ine 7 ve y sine 2 değeri atanır.

    cout << "Alan1: " << D1.Alan()<<'\n';
    cout << "Alan2: " << D2.Alan()<<endl;

    getch();
    return 0;
}

```

Nesnelerin Dinamik olarak kullanımları

Nesneleri sınıfları kullanarak tanımladık. Daha önceki derslerimizde istediğimizde dinamik olarak bellekten yer alma ve geri verme işlemlerini görmüştük. Nesneler için de bellekten dinamik olarak yer almak ve işi bitince de geri vermek mümkündür.

Bunun için aşağıdaki gibi bir tanımlama yapmak gerekir.

```
Dortgen *P1=new Dortgen;  
Dortgen *P2=new Dortgen;
```

Bu tanımlamalarda P1 ve P2 Dortgen türünde pointer nesneler olarak tanımlanmıştır. Dortgen sınıfının bellekte işgal edeceği yer kadar bellekte yer ayrılmakta ve başlangıç adresi P1 ve P2 pointer nesnelere atanmaktadır.

Dortgen sınıfının aşağıdaki gibi bildirimi yapıldığı var sayılsın. x ve y public olduğuna dikkat ediniz.

```
class Dortgen {  
public:  
    int x, y;  
    void DegerVer(int,int);  
    int Alan (void);  
};
```

Aşağıdaki şekilde P1 ve P2 nesneleri tanımlandığında,

```
Dortgen *P1=new Dortgen;  
Dortgen *P2=new Dortgen;
```

Bu nesnelerin üye değişkenlerine şu şekilde erişilir.

```
P1->x=5; P1->y=10;  
P2->x=4; P2->y=20;
```

Fonksiyon erişimleri(**Sınıfın metodlarına**)şu şekilde olur.

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

```
P2->DegerVer(5,8);  
P2->Alan();
```

P1 ve P2 bellekten

```
Delete P1;  
Delete P2;
```

Şeklinde atılırlar.

Aşağıdaki programı inceleyiniz

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

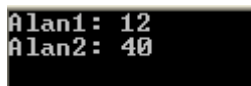
class Dortgen {
    int x, y;
public:
    void DegerVer(int,int);
    int Alan (void);
};

void Dortgen::DegerVer(int a, int b) {
    x = a;
    y = b;
}

int Dortgen::Alan (void) {
    return x*y;
}

int main ()
{
    Dortgen *P1=new Dortgen;
    Dortgen *P2=new Dortgen;

    P1->DegerVer(3,4);
    P2->DegerVer(5,8);
    cout << "Alan1: " << P1->Alan()<<'\n';
    cout << "Alan2: " << P2->Alan()<<endl;
    delete P1, P2;
    getch();
    return 0;
}
```



```
Alan1: 12
Alan2: 40
```

Nesnelerin dizi olarak kullanımları

Nesnelerde diğer türlerde olduğu gibi dizi olarak tanımlanıp kullanılabilirler. Aşağıdaki örneği inceleyiniz.

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

class Dortgen {
public:
    int x, y;
    void DegerVer(int,int);
    int Alan (void);
};

void Dortgen::DegerVer(int a, int b) {
x = a;
y = b;
}

int Dortgen::Alan (void) {
return x*y;
}

int main ()
{
    Dortgen P1[10];

    P1[0].x=6;
    P1[0].y=8;

    P1[1].DegerVer(7,2);

    cout << "Alan1: " << P1[0].Alan()<<'\n';
    cout << "Alan2: " << P1[1].Alan()<<endl;

    getch();
    return 0;
}
```