Bölüm 19 – Kalıtsallık(Inheritance)

İçerik

19.1 Giriş

- Kalıt
  - Varolan sınıflardan yeni sınıflar elde etme
  - Nitelikleri ve davranışları yoketme.
- Polimorfizm
  - Programları genel tarzda yazar
  - Çok değişik tipten, varolan ve (belirtilmeyen) ilişkili sınıfları kullanır
- Türev Sınıfı
  - Veri üyelerini ve üye fonksiyonlarını daha önce tanımlanmış bir baz sınıfından kalıt olarak alan sınıf

- Kalıt
  - Tek kalıt
    - Sınıf, tek bir baz sınıfından kalıtsaldır
  - Çoklu kalıt
    - Sınıf, çoklu baz sınıflarından kalıtsaldır
  - Üç tip kalıt:
    - **`public`**: Türev nesneleri baz sınıfı neslerinden erişimlidir (Bu bölümde verilecek)
    - **`private`**: Türev nesneleri baz sınıfı neslerinden erişimli değildir
    - **`protected`**: Türev sınıfları ve Arkadaşlar baz sınıfının korunumlu üyelerine erişim sağlayabilir

19.2 Baz ve Türev Sınıfları

- Türev sınıfından (alt sınıftan) bir nesne çoğu zaman baz sınıfın (üst sınıfın) bir nesnesidir

Baz sınıfı	Türev sınıfı
`Ogrenci`	`YuksekLisOgr` `LisansOgr`

`Sekil`	`Daire`
	`Ucgen`
	`Dortgen`

`Kredi`	`TasıtKredisi`
	`EvKredisi`
	`IpotekKredisi`

`Calısan`	`OgrUyesi` `IdariCal`
`Hesap`	`BankomatHesabi` `YatirimHesabi`

- **`public`** kalıtının uygulanması

```
class KomisyonIsci : public Isci{
    ...
};
```

`KomisyonIsci` sınıfı **`Isci`** sınıfından kalıttır

- **`friend`** fonksiyonlar kalıt değildir
- Baz sınıfının **`private`** üyeleri Türev sınıfından erişilemez

19.3 `protected` Üyeler

- **`protected`** kalıt
  - **`public`** ve **`private`** kalıt arasında orta düzey bir koruma
  - Türev-sınıfı üyeler, basitçe üye adlarını kullanarak baz sınıfının **`public`** ve **`protected`** üyelerini referans verebilirler
  - **`protected`** veri içermeyi (parantezi) kırabilir

19.4 Baz-Sınıfı Üyelerden Türev Sınıfı Yapma

- Türev sınıfı nesnesi
  - Baz sınıfı nesnesi olarak işlenebilir
  - Tersi doğru değil- baz sınıfı nesnesi türev sınıfı nesnesi değildir
- Pointer-ı düşürme
  - Bir baz sınıfı pointer-ı türev sınıfı pointer-a dönüştürmek için bir açık atama kullan
  - Pointer tipinin pointer-ın işaretlediği nesne tipiyle aynı olduğundan emin ol

```
turevPtr = static_cast< TurevSinifi * > bazPtr;
```

- Örnek
  - **`Daire sınıfı`** baz **`sınıfı Nokta`** dan türetilmiştir
  - Bir **`Daire`** nesnesini referans vermek için **`Nokta`** tipi bir pointer kullanıyoruz (veya tersi).

```
1  // Fig. 19.4: nokta.h
2  // Nokta sınıfı tanımı
3  #ifndef NOKTA H
4  #define NOKTA H
5
6  #include <iostream>
7
8  using std::ostream;
9
10 class Nokta {
11    friend ostream &operator<<( ostream &, const Nokta & );
12 public:
13    Nokta( int = 0, int = 0 );       // default constructor
14    void noktaKur( int, int );       // koordinatları olustur
15    int alX() const { return x; }  // x koordinatı
16    int alY() const { return y; }  // y koordinatı
17 protected:          // sürücü sınıfından ulasılabilir
18    int x, y;        // Nokta-nın x ve y koordinatları
19 };
20
21 #endif
22 // Fig. 19.4: nokta.cpp
23 // Nokta sınıfı için üye fonksiyonlar
24 #include <iostream>
25 #include "nokta.h"
26
27 // Nokta için Constructor
28 Nokta:: Nokta( int a, int b ) { noktaKur( a, b ); }
29
30 // Nokta nın x ve y koordinatlarını kur
31 void Nokta ::noktaKur( int a, int b )
32 {
33    x = a;
```

1. Nokta class tanımı

---

1. header yükle

1.1 fonksiyon tanımları

```
34    y = b;
35 }
36
37 // Nokta çıktısı  (overloaded stream insertion operatörü ile )
38 ostream &operator<<( ostream &output, const Nokta &p )
39 {
40    output << '[' << p.x << ", " << p.y << ']';
41
42    return output;
43 }
44 // Fig. 19.4: daire.h
45 // Daire sınıfı tanımı
46 #ifndef DAIRE_H
47 #define DAIRE_H
48
49 #include <iostream>
50
51 using std::ostream;
52
53 #include <iomanip>
54
55 using std::ios;
56 using std::setiosflags;
57 using std::setprecision;
58
59 #include " nokta.h"
60
61 class Daire : public Nokta {  // Daire Nokta-dan kalıttır
62    friend ostream &operator<<( ostream &, const Circle & );
63 public:
64    // default constructor
```

1.1 fonksiyon tanımları

---

1. `Daire` class tanımı

```
65    Daire( double r = 0.0, int x = 0, int y = 0 );
66
67    void kurYcap( double );    // Yarıçapı kur
68    double alYcap() const;    // yarıçapı gönder
69    double alan() const;         // alan hesapla
70 protected:
71    double ycap;
72 };
73
74 #endif
75 // Fig. 19.4: daire.cpp
76 // Daire için üye fonksiyon tanımı
77 #include "daire.h"
78
79 // Daire constructer bir üye ilkdeğerleri ile nokta constructer-ı
80 // çağırıyor
81 Daire::Daire( double r, int a, int b )
82    : Nokta( a, b )        // baz-class constructor çağırır
83 { kurYcap( r ); }
84
85 // Daire yarıçapını kur
86 void Daire::kurYcap( double r )
87    { ycap = ( r >= 0 ? r : 0 ); }
88
```

1. Daire tanımı

---

1. Header yükle

1.1 fonksiyon tanımları

```
89 // Dairenin yaricapini al
90 double Daire::alYcap() const { return ycap; }
91
92 // Dairenin alanini hesapla
93 double Daire::alan() const
94    { return 3.14159 * ycap * ycap; }
95
96 // Dairenin alani :
97 // Merkez = [x, y]; Yaricap = #.##
98 ostream &operator<<( ostream &output, const Daire &c )
99 {
100    output << "Merkez = " << static_cast< Nokta >( c )
101           << "; Yaricap = "
102           << setiosflags( ios::fixed | ios::showpoint )
103           << setprecision( 2 ) << c.ycap;
104
105    return output;
106 }
107 // Fig. 19.4: fig19_04.cpp
108 // baz-class pointeri derived-class pointer-a çevir
109 #include <iostream>
110
111 using std::cout;
112 using std::endl;
113
114 #include <iomanip>
115
116 #include "nokta.h"
117 #include "daire.h"
118
119 int main()
120 {
121    Nokta *noktaPtr = 0, p( 30, 50 );
```

1. 1 Fonksiyon tanımları

Sürücü

1. header-ları yükle

1.1 nesneleri belirle

```
122    Daire *dairePtr = 0, c( 2.7, 120, 89 );
123
124    cout << "Nokta p: " << p << "\nDaire c: " << c << '\n';
125
126    // Daireyi nokta gibi işler (baz sınıfı kısmına bak)
127    noktaPtr = &c;   // Dairenin adresini noktaPtr ye aktar
128    cout << "\nDaire c ( *noktaPtr ile): "
129         << *noktaPtr << '\n';
130
131    // Daireyi daire gibi işle
132    // baz-class pointerı derived-class pointera çevir
133    DairePtr = static_cast< Daire * >( noktaPtr );
134    cout << "\nDaire c (*dairePtr ile):\n" << *dairePtr
135         << "\n c nin alanı ( dairePtr ile): "
136         << dairePtr->alan() << '\n';
137
138    // TEHLİKELİ: Noktayi Daire gibi işle
139    noktaPtr = &p;
140
141    // baz-class pointerı derived-class pointera çevir
142    dairePtr = static_cast< Daire * >( noktaPtr );
143    cout << "\nNokta p (*dairePtr ile):\n" << *dairePtr
144         << "\n dairePtr nesnesinin alanı : "
145         << dairePtr->alan() << " ı işaret eder" << endl;
146    return 0;
147 }
```

1.1 NEsneleri belirle

1.2 Nesneleri ata

2. Fonksiyon çağır

19.4 Baz-Sınıfı Üyelerden Türev Sınıfı Yapma

```
Nokta p: [30, 50]
Daire c: Merkez = [120, 89]; Yarıçap = 2.70

Daire c ( *noktaPtr ile): [120, 89]

Daire c ( *dairePtr ile):
Merkez = [120, 89]; Yarıçap= 2.70
C nin alanı ( dairePtr ile): 22.90

Nokta p ( *dairePtr ile):
Merkez = [30, 50]; Yarıçap= 0.00
dairePtr nesnesinin alanı : 0.00 ı işaret eder
```

19.5 Üye Fonksiyon Kullanımı

- Türev Sınıfı
  - Baz sınıfın **`private`** üyelerine doğrudan erişim sağlayamaz
  - **`private`** üyeleri gizlemede, sistemleri test etme, düzenleme ve yenilemede oldukça yardımcıdır.

19.6 Baz-Sınıfı Üyeleri Türev Sınıfına Yükleme

- baz-sınıfı üye fonksiyon yüklemesi
  - Türev sınıfında, o fonksiyonun yeni versiyonunu al
    - Aynı fonksiyon adı, farklı tanım
  - Baz sınıfı versiyonuna türev sınıfından ulaşmak için hedef-çözünürlük operatörü kullanılabilir

19.6 Baz-Sınıfı Üyeleri Türev Sınıfına Yükleme

```
// Fig. 19.5: isci.h
// Isci sınıfı tanımları
#ifndef ISCI_H
#define ISCI_H

class Isci {
public:
   Isci( const char *, const char * );  //
constructor
   void yaz() const; // İlk ve soyadı yaz
   ~Isci();          // destructor
private:
   char *ilkAd;      // dinamik düzenlenen string
   char *soyAd;      // dinamik düzenlenen string
};
#endif
```

```
// Fig. 19.5: isci.cpp
// Isci için üye fonksiyonlar
#include <iostream>

using std::cout;

#include <cstring>
#include <cassert>
#include "isci.h"

// Constructor ilk ve soyad için dinamik düzenlemeli
yer ayırır
// ve ilk ve soyadı nesneye kopyalamak için strcpy
// kullanır
Isci::Isci( const char *ilk, const char *soy)
{
   ilkAd= new char[ strlen( ilk) + 1 ];
   assert( ilkAd!= 0 ); // yer yok ise dur
   strcpy( ilkAd, ilk);
```

1. `Isci` class tanımı

---

1. Header yükle

1.1 Fonksiyon tanımları

19.6 Baz-Sınıfı Üyeleri Türev Sınıfına Yükleme

```
soyAd= new char[ strlen( soy) + 1 ];
   assert( soyAd!= 0 );  // yer yok ise dur
   strcpy( soyAd, soy);
}

void Isci::yaz() const
   { cout << ilkAd << ' ' << soyAd; }

// Destructor dinamik düzenlemeyi kaldırır
Isci::~Isci()
{
   delete [] ilkAd;   // dinamik belleği düzenle
   delete [] soyAd;
}
```

```
// Fig. 19.5: saatlik.h
// SaatlikIsci classı
#ifndef SAATLIK_H
#define SAATLIK_H
#include "isci.h"

class SaatlikIsci : public Isci {
public:
   SaatlikIsci( const char*, const char*, double, double );
   double odeme() const;  // maaşı hesapla ve gönder
   void yaz() const;      // önemsiz baz-class yaz fonksiyonu
private:
    double ucret;                  // saatlik ücret
    double saat;                // haftalık çalışma
saati
};

#endif
```

1.1 Fonksiyon tanımları

---

1. `SaatlikIsci` class tanımı

19.6 Baz-Sınıfı Üyeleri Türev Sınıfına Yükleme

```
// Fig. 19.5: saatlik.cpp
// SaatlikIsci için üye fonksiyonlar
#include <iostream>

using std::cout;
using std::endl;

#include <iomanip>

using std::ios;
using std::setiosflags;
using std::setprecision;

#include "saatlik.h"

// Constructor
SaatlikIsci::SaatlikIsci( const char *ilk,
                          const char *soy,
                          double ilksaat,
double ilkUcret)
   : Isci( ilk, soy)    // baz-class
constructor-ı çağır
{
   saat = ilksaat;  // geçerli olmalı
   ucret = ilkUcret;    // geçerli olmalı
}

// Odemeyi al
double SaatlikIsci::odeme() const { return
ucret * saat; }
```

1. Header yükle

1.1 fonksiyon tanımları

19.6 Baz-Sınıfı Üyeleri Türev Sınıfına Yükleme

```
// Ad ve ödemeyi yaz
void SaatlikIsci::yaz() const
{
   cout << "SaatlikIsci::yaz() çalışıyor\n\n";
   Isci::yaz();   // baz-class yaz fonksiyonunu çağır

   cout << " saatlik işçi olup odenecek $"
        << setiosflags( ios::fixed | ios::showpoint )
        << setprecision( 2 ) << odeme() << endl;
}

// Fig. 19.5: fig.19_05.cpp
// sürücü sınıfındaki bir baz-class üye fonksiyonunu
// atlama
#include "saatlik.h"

int main()
{
   SaatlikIsci s( "Ali", "Çalışkan", 40.0, 10.00 );
   s.yaz();
   return 0;
}
```

```
SaatlikIsci::yaz() çalışıyor

Ali Çalışkan saatlik işçi olup ödenecek $400.00
```

1.1 fonksiyon tanımları

---

1. Header yükle

1.1 nesne değerlerini gir

2. Fonksiyon çağır

19.7 public, private, ve protected Kalıt

Baz sınıfı üye erişim belirteci	Kalıt tipi		
	public kalıt	**protected** kalıt	**private** kalıt
public	Türev sınıfında **public**. Herhangi **static-** olmayan üye fonksiyon, **friend** fonksiyon ve üye-olmayan fonksiyon ile doğrudan erişilebilir,	Türev sınıfında **protected** **static-**olmayan üye fonksiyonlar, ve **friend** fonksiyon ile erişilebilir	Türev sınıfında **private** **static-**olmayan üye fonksiyon ve **friend** fonksiyon ile erişilebilir
protected	Türev sınıfında **protected** **static-**olmayan üye fonksiyon ve **friend** fonksiyon ile erişilebilir	Türev sınıfında **protected** **static-**olmayan üye fonksiyon ve **friend** fonksiyon ile erişilebilir	Türev sınıfında **private** **static-**olmayan üye fonksiyon ve **friend** fonksiyon ile erişilebilir
private	Türev sınıfında saklı. Baz sınıfının **public** veya **protected** üyeleri üzerinden **static-** olmayan üye fonksiyon ve **friend** fonksiyon ile erişilebilir	Tüev sınıfında saklı. Baz sınıfının **public** veya **protected** üyeleri üzerinden **static-** olmayan üye fonksiyon ve **friend** fonksiyon ile erişilebilir	Türev sınıfında saklı. Baz sınıfının **public** veya **protected** üyeleri üzerinden **static-** olmayan üye fonksiyon ve **friend** fonksiyon ile erişilebilir.

- Doğrudan baz sınıfı
  - Türev sınıfı tanımlandığında, (**:**) notasyonu ile o türev sınıfı header- ında açık olarak listelenir
  - **`class : public Isci`**
    - **`Isci, SaatlikIsci'`** nin doğrudan baz sınıfıdır
- Dolaylı Baz sınıfı
  - Class hiyerarşisinin iki veya daha üst düzeyinden kalıttır
  - **`class DakikaIsci: public SaatlikIsci`**
    - **`Isci, DakikaIsci'`** nin dolaylı baz sınıfıdır

19.9 Türev Sınıfında Constructor ve Destructor Kullanımı

- Baz class başlatıcısı
  - Üye başlatıcı sintaksı kullanır
  - Baz-sınıfı oluşturucusunu doğrudan çağırmak için türev sınıfı oluşturucuda verilebilir
  - Aksi halde baz sınıfı' default oluşturucusu kapalı olarak çağrılır
  - Baz-sınıfı oluşturucusu ve baz-sınıfı atama operatörleri türev sınıfından kalıt değildir
    - Fakat, türev sınıfı oluşturucuları ve atama operatörleri onları çağırabilir

- Türev sınıfı oluşturucusu
  - Baz sınıfı üyelerini atamak için ilk önce baz sınıfı için oluşturucuyu çağırır
  - Eğer türev sınıfı oluşturucusu gözardı edilirse, onun default oluşturucusu baz-sınıfı' default oluşturucuyu çağırır
- Yokediciler, oluşturucuların tam zıt sırası ile çağrılır
  - Türev sınıfı yokedicisi, baz sınıfı yokedicisinden önce çağrılır

```
// Fig. 19.7: nokta2.h
// Nokta sınıfı tanımı
#ifndef NOKTA2_H
#define NOKTA2_H

class Nokta {
public:
   Nokta( int = 0, int = 0 );  // default
constructor
   ~Nokta();     // destructor
protected:       // sürücü sınıfları erişebilir
   int x, y;     // Noktanın x ve y koordinatları
};

#endif
```

```
// Fig. 19.7: nokta2.cpp
// Nokta için üye fonksiyonlar
#include <iostream>

using std::cout;
using std::endl;

#include "nokta2.h"

// Constructor
Nokta::Nokta( int a, int b )
{
   x = a;
   y = b;

   cout << "Nokta constructor: "
        << '[' << x << ", " << y << ']' << endl;
}
```

```
// Destructor
Nokta::~Nokta()
{
   cout << "Nokta destructor:  "
        << '[' << x << ", " << y << ']' << endl;
}
```

```
// Fig. 19.7: daire2.h
// Daire tanımı
#ifndef DAIRE2_H
#define DAIRE2_H

#include "nokta2.h"

class Daire: public Nokta {
public:
   // default constructor
   Daire( double r = 0.0, int x = 0, int y = 0 );

   ~Daire();
private:
   double ycap;
};

#endif
```

```
// Fig. 19.7: daire2.cpp
// Daire için üye fonksiyonlar
#include <iostream>

using std::cout;
using std::endl;

#include "daire2.h"

// Daire Constructor-ı  Nokta constructor-ını çağırır
Daire::Daire( double r, int a, int b )
   : Nokta( a, b )   // baz-class constructor-ı çağır
{
   ycap = r;
   cout << "Daire constructorı: Yarıçap = "
        << ycap << " [" << x << ", " << y << ']' << endl;
}

// Destructor
Daire::~Daire()
{
   cout << "Daire destructor:  Yarıçap = "
        << ycap << " [" << x << ", " << y << ']' << endl;
}
```

```
// Fig. 19.7: fig19_07.cpp
// Demonstrate when baz-class ve sürücü-class
// constructor-lar ve destructor-lar çağrıldığında
#include <iostream>

using std::cout;
using std::endl;

#include "nokta2.h"
#include "daire2.h"

int main()
{
   // Nokta için constructor ve destructor çağrımını
göster
   {
      Nokta p( 11, 22 );
   } // blok sonu

   cout << endl;
   Daire daire1( 4.5, 72, 29 );
   cout << endl;
   Daire daire2( 10, 5, 5 );
   cout << endl;
   return 0;
}
```

```
Daire  constructorı: [11, 22]
Nokta  destructorı:  [11, 22]

Nokta  constructorı: [72, 29]
Daire constructorı: radius is 4.5 [72, 29]

Nokta  constructorı: [5, 5]
Daire constructorı: radius is 10 [5, 5]

Daire destructorı:  radius is 10 [5, 5]
Nokta  destructorı:  [5, 5]
Daire destructorı:  radius is 4.5 [72, 29]
Nokta  destructorı:  [72, 29]
```

19.10 Kapalı Türev-Sınıfı Nesneleri Baz-Sınıfı Nesnelere Dönüştürme

- **`bazSınıfıNesne = turevSınıfıNesne;`**
  - Çalışır
    - Türev sınıfı nesnelerin baz sınıf nesnelerden daha fazla üyeye sahip olduğunu hatırlayınız
  - Baz sınıfına ilave data verilmez

`turevSınıfıNesnesi = bazSınıfıNesnesi;`

- Düzgün çalışmayabilir
  - Bir atama operatörü türev sınıfında yüklenmedikçe, türev sınıfındaki fazla veri üyeleri atanamayacaktır
- Baz sınıfı, türev sınıfından daha az veri üyesine sahiptir
  - Bazı veri üyeleri türev sınıfı nesnesinde kaybolur

- Baz ve Türev sınıfı pointer ve nesneleri birleştirmenin dört yolu
  - Bir baz-sınıfı nesneyi bir baz sınıfı pointer ile refere etme
    - İzin verilir
  - Bir türev sınıfı nesneyi bir türev sınıfı pointer ile refere etme
    - İzin verilir
  - Bir türev sınıfı nesneyi bir baz sınıfı pointer ile refere etme.
    - Muhtemelen sintaks hatası
    - Kod sadece baz sınıf üyeyi refere edebilir, veya sintaks hatası
  - Bir baz sınıfı nesneyi bir türev sınıfı pointer ile refere etme
    - Sintaks hatası
    - Türev sınıfı pointer önce bir baz sınıfı pointer-a atanmalıdır

19.11 Kalıt ile Yazılım Mühendisliği

Sınıflar çoğu zaman yakın ilişkilidir

- Ortak atama ve davranışları sadeleştir ve bunları bir baz sınıfına yerleştir
- Türev sınıflarını oluşturmak için kalıt kullan

Baz sınıfında düzenlemeler

- **`public`** ve **`protected`** arayüzler aynı oldukça türev sınıfları değişmez
- Türev sınıfları yeniden derlenmek zorunda kalabilir

9.12 Kompozisyon ve Kalıt

Kalıt "bir ilişkidir"

Kompozisyon – bir başka sınıftan bir veri üyesi olarak bir nesneye sahip sınıf

"bir ilişkiye sahiptir"

```
Isci "bir " DogumGunu;   //Yanlış!
Isci "sahiptir " DogumGunu;//Kompozisyon
```

9.13 İlişkileri “Kullan” ve “Bil”

- İlişki “kullan”
  - Bir nesne bir başka nesnenin bir üye fonksiyonuna bir fonksiyon çağrımı gerçekleştirir

- İlişki “bil”
  - Bir nesne bir başkasının farkındadır
    - Bir başka nesneye pointer veya bağlayıcı içerir
  - Bir birlik olarak da adlandırılır

9.14 Örnek: Nokta, Daire, Silindir

- **`Nokta`** sınıfı tanımla
  - **`Daire`** sürücüsü oluştur
    - **`Silindir`** sürücüsü oluştur

9.14 Örnek: Nokta, Daire, Silindir

```
// Fig. 19.8: nokta2.h
// class tanımı
#ifndef NOKTA2_H
#define NOKTA2_H

#include <iostream>

using std::ostream;

class Nokta {
   friend ostream &operator<<( ostream &, const Nokta & );
public:
   Nokta ( int = 0, int = 0 );       // default constructor
   void kurNokta( int, int );       // koordinatları kur
   int alX() const { return x; }  // x koordinatını al
   int alY() const { return y; }  // y koordinatını al
protected:         // sürücü sınıfları erişebilir
   int x, y;       // nokta koordinatları
};

#endif
```

```
// Fig. 19.8: nokta2.cpp
// Üye fonksiyonlar
#include "nokta2.h"

// Constructor
Nokta::Nokta( int a, int b ) { kurNokta( a, b ); }

void Nokta::kurNokta( int a, int b )
{
   x = a;
   y = b;
}

// Çıktı
ostream &operator<<( ostream &cikti, const Nokta &p
)
{
   cikti << '[' << p.x << ", " << p.y << ']';

   return cikti;
} // << sonu
```

```
// Fig. 19.9: daire2.h
//
#ifndef DAIRE2_H
#define DAIRE2_H

#include <iostream>

using std::ostream;

#include "nokta2.h"

class Daire: public Nokta {
   friend ostream &operator<<( ostream &, const
Daire & );
public:
   // default constructor
   Daire( double r = 0.0, int x = 0, int y = 0 );
   void kurycap( double );
   double alycap() const;
   double alan() const;
protected:
   double ycap;
};

#endif
```

```
// Fig. 19.9: daire2.cpp
// Üye fonksiyonlar
#include <iomanip>
using std::ios;
using std::setiosflags;
using std::setprecision;
```

```
#include "daire2.h"

// Daire Constructorı bir üye belirteci ile
Nokta constructorını çağırır
// ve ycap ilk değerini atar
Daire::Daire( double r, int a, int b )
   : Nokta ( a, b )        // baz-class
constructorı çağır
{ kurycap( r ); }

void Daire::kurycap( double r )
   { ycap= ( r >= 0 ? r : 0 ); }


double Daire::alycap() const { return ycap; }

// Alanı hesapla
double Daire::alan() const
   { return 3.14159 * ycap* ycap; }

// Çıktı

ostream &operator<<( ostream &cikti, const
Daire &c )
{
   cikti << "Merkez= " << static_cast< Nokta >
( c )
         << "; Yarıçap = "
         << setiosflags( ios::fixed |
ios::showpoint )
         << setprecision( 2 ) << c.ycap;

   return cikti;
} // << sonu
```

```
// Fig. 19.10: silindir2.h

#ifndef SILINDIR2_H
#define SILINDIR2_H

#include <iostream>

using std::ostream;

#include "daire2.h"

class Silindir : public Daire{
   friend ostream &operator<<( ostream &, const
Silindir & );

public:
   // default constructor
   Silindir( double h = 0.0, double r = 0.0,
             int x = 0, int y = 0 );

   void kurYuk( double );    // yüksekliği kur
   double alYuk() const;    // return yükseklik
   double alan() const;
   double hacim() const;

protected:
   double yuk;                // Silindirin yüksekliği
};

#endif
```

```
// Fig. 19.10: silindir2.cpp

#include "silindir2.h"

// Silindir constructor Daire constructorı çağırır
Silindir::Silindir( double h, double r, int x, int y )
   : Daire( r, x, y )   // baz-class constructorı çağır
{ kurYuk( h ); }

void Silindir::kurYuk( double h )
   { yuk= ( h >= 0 ? h : 0 ); }

double Silindir::alYuk() const { return yuk; }
// yüzey alanı
double Silindir::alan() const
{
   return 2 * Daire::alan() +
          2 * 3.14159 * ycap* yuk;
}

// Hacim
double Silindir::hacim() const
   { return Daire::alan() * yuk; }

// Silindir boyutlarının çıktısı
ostream &operator<<( ostream &cikti, const Silindir &c )
{
   cikti << static_cast< Daire >( c )
         << "; Yükseklik = " << c.yuk;

   return cikti;
}
```

```
// Fig. 19.10: fig19_10.cpp
// Silindir sınıfı için türev
#include <iostream>

using std::cout;
using std::endl;

#include "nokta2.h"
#include "daire2.h"
#include "silindir2.h"

int main()
{
   // Silindir nesnesi yarat
   Silindir sil( 5.7, 2.5, 12, 23 );

   cout << "X koordinatı:" << sil.alX()
        << "\nY koordinatı:" << sil.alY()
        << "\nyarıçap:" << sil.alycap()
        << "\nYükseklik: " << sil.alYuk() <<
"\n\n";

   // Değerleri değiştir
   sil.kurYuk( 10 );
   sil.kurycap( 4.25 );
   sil.kurNokta( 2, 2 );
   cout << "Silindirin yeni yüksekliği
yarıçapı ve yeri:\n"
        << sil<< '\n';
```

```
cout << "Yüzey Alanı:\n"
        << sil.alan() << '\n';

   // Silindiri nokta olarak göster
   Nokta &pRef = sil;   // pRef onun bir nokta olduğunu
"düşünür"
   cout << "\nNokta olarak yazılan silindir: "
        << pRef << "\n\n";

   // Silindiri Daire olarak göster
   Daire &daireRef = sil;  // daireRef onu bir daire gibi
görür
   cout << "Daire olarak yazılan silindir:\n" << daireRef
        << "\nAlan: " << daireRef.alan() << endl;

   return 0;
}
```

```
X koordinatı: 12
Y koordinatı: 23
yarıçap: 2.5
Yükseklik: 5.7

Silindirin yeni yüksekliği yarıçapı ve yeri:
Merkez = [2, 2]; Yarıçap = 4.25; Yükseklik = 10.00
Silindirin alanı:
380.53
Nokta olarak yazılan silindir: [2, 2]

Daire olarak yazılan silindir:
Merkez= [2, 2]; Yarıçap= 4.25
Alan: 56.74
```