# Bölüm 20- Virtual(Sanal) Fonksiyonlar ve Polimorfizm

## İçerik

# 20.1 Giriş

- **`virtual`** fonksiyonlar ve Polimorfizm
  - Kolaylıkla genişletilebilir sistemler tasarla ve uygula
  - Programlar, bir hiyerarşi içindeki tüm varolan sınıfları genel anlamda işlemek için yazılabilir

# 20.2 Tip Alanları ve `switch` Deyimleri

- **`switch`** deyimi
  - Nesnenin tipine göre hareket et
  - Bir **`switch`** yapısı, , **`sekiller`** hiyerarşisindeki tipe göre hangi **`yaz`** fonksiyonunu kullanacağını belirleyeyebilir

- **`switch`** ile ilgili sorunlar
  - Programcı bir switch deki tüm olası durumları test etmeyi unutabilir
    - Bunu takip etmek zaman alıcı ve hata yapmaya açıktır
    - **`virtual`** fonksiyonlar ve polimorfik programlama **`switch`** gereksinimini ortadan kaldırabillir

# 20.3 Sanal(Virtual) Fonksiyonlar

- **`virtual`** fonksiyonlar
  - **`switch`** deyimleri yerine kullan
  - Deklarasyon:
    - Baz sınıfındaki prototip fonksiyonun önünde **`virtual`** yaz

```
virtual void ciz() const;
```

  - Bir sürücü sınıfı nesneye baz sınıfı pointer doğru **`çiz`** fonksiyonunu çağıracaktır
  - Eğer sürücü sınıfında bir **`virtual`** fonksiyon tanımlamamışsa baz sınıfından kalıt alınır

- **`SekilPtr->Ciz();`**
  - Derleyici dinamik birleştirme uygular
  - Fonksiyon, çalışma (execution) anında belirlenir
- **`SekilNesne.Ciz();`**
  - Derleyici statik birleştirme uygular
  - Fonksiyon, derleme(compiling) anında belirlenir

# 20.4 Soyut ve Somut Sınıflar

- Soyut sınıflar
  - Temel amacı, diğer sınıflar için bir baz sınıfı sağlamaktır
  - Soyut baz sınıfının hiç bir nesnesi sabitlenemez
    - Real nesne tanımlamak için çok genel; örneğin, **`IkiBoyutluSekil`**
    - Pointer ve referansa sahip olabilir
- Somut sınıflar

  Nesneleri sabitlenebilir
  - Reel nesneler için özel tanım sağlar; örneğin,

    **`Kare`**, **`Daire`**
- Soyut sınıf yapma
  - Sınıfın bir veya daha çok **`virtual`** fonksiyonunu "soyut" olarak (başlangış değerini sıfır alarak) tanımla

```
virtual double kazanclar() const = 0;
```

  - Soyut **`virtual`** fonksiyon

- Polimorfizm:
  - Farklı sınıflardan nesnelerin (kalıtsallıklarına bağlı olarak) aynı fonksiyon çağrımlarına farklı yanıt verme yeteneği
  - Baz-sınıfı pointer (veya referans) bir **`virtual`** fonksiyon çağırır
    - C++ nesnedeki doğru çoklu - yüklü fonksiyonu seçer
  - **`yaz`** bir **`virtual`** fonksiyon olmasın. Bu durumda

```
Isci e, *ePtr = &e;
SaatlikIsci h, *hPtr = &h;
ePtr->yaz();   //baz(Isci)-sınıfı yaz fonksiyonunu çağırır
hPtr->yaz(); // türev(SaatlikIsci)-sınıfı yaz fonksiyonunu çağırır
ePtr=&h;          //izin verilebilir kapalı dönüşüm
ePtr->yaz();   // yine baz(Isci)-sınıfı yaz fonksiyonunu çağırır
```

# 20.6 Yeni Sınıflar ve Dinamik Birleştirme

- Dinamik Birleştirme (sonradan birleştirme)
  - **`virtual`** fonksiyonları derlerken nesne tipine gereksinim yok
  - Derlemeden sonra oluşturulan yeni sınıflara yer açar
  - Kaynak kodu müşterilerine vermek istemeyen bağımsız yazılımcılar için önemli

## 20.7 Virtual Yokediciler

- Problem:
  - Eğer (virtual olmayan yokedicili) türev nesnesi baz sınıfı pointer-a uygulanan **`delete`** operatörü ile yokedilirse, (o türev nesnesininki yerine) nesnedeki baz sınıfı yokedicisi çağrılır

- Çözüm:
  - **`virtual`** baz sınıfı yokedicisi tanımla
  - Bu durumda uygun yokedici çağrılır

## 20.8 Örnek: Kalıtsal Arayüz ve Uygulama

- **`Nokta`**, **`Daire`**, **`Silindir`** hiyerarşisini tekrar inceleme
  - Hiyerarşinin başında Soyut baz sınıfı **`Sekil`** kullanılacak

# 20.8 Örnek: Kalıtsal Arayüz ve Uygulama

1. `Sekil` Tanımı
   (soyut baz sınıfı)

---

1. `Nokta` tanımı
   (türev sınıfı)

```
// Fig. 20.1: sekil.h
// soyut baz classı Sekil
#ifndef SEKIL_H
#define SEKIL_H

class Sekil {
public:
   virtual double alan() const { return 0.0; }
   virtual double hacim() const { return 0.0; }

   // saf virtual fonksiyonlar sürücü sınıflarında atılır
   virtual void yazSekilAd() const = 0;
   virtual void yaz() const = 0;
};

#endif
```

```
// Fig. 20.1: nokta1.h

#ifndef NOKTA1_H
#define NOKTA1_H

#include <iostream>

using std::cout;

#include "sekil.h"

class Nokta: public Sekil{
public:
   Nokta( int = 0, int = 0 );  // default constructor
   void kurNokta( int, int );
   int alX() const { return x; }
   int alY() const { return y; }
```

# 20.8 Örnek: Kalıtsal Arayüz ve Uygulama

```
virtual void yazSekilAd() const { cout << "Nokta: "; }
   virtual void yaz() const;
private:
   int x, y;
};

#endif
```

```
// Fig. 20.1: nokta1.cpp

#include "nokta1.h"

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

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

void Nokta::yaz() const
   { cout << '[' << x << ", " << y << ']'; }
```

1. `Nokta` tanımı (türev sınıfı)

1.1 Fonksiyon tanımları

# 20.8 Örnek: Kalıtsal Arayüz ve Uygulama

1. `Daire` tanımı (türev sınıfı)

```
// Fig. 20.1: daire1.h
#ifndef DAIRE1_H
#define DAIRE1_H
#include "nokta1.h"

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

   void kurycap( double );
   double alycap() const;
   virtual double alan() const;
   virtual void yazSekilAd() const { cout << "Daire: "; }
   virtual void yaz() const;
private:
   double ycap;
};

#endif
```

# 20.8 Örnek: Kalıtsal Arayüz ve Uygulama

```
// Fig. 20.1: daire1.cpp

#include <iostream>

using std::cout;

#include "daire1.h"

Daire::Daire( double r, int a, int b )
   : Nokta( a, b )
{ kurycap( r ); }

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

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

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

void Daire::yaz() const
{
   Nokta::yaz();
   cout << "; Yarıçap= " << ycap;
}
```

# 20.8 Örnek: Kalıtsal Arayüz ve Uygulama

```
// Fig. 20.1: silindir1.h

#ifndef SILINDIR1_H
#define SILINDIR1_H
#include "daire1.h"

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

   void kurYuk( double );
   double alYuk();
   virtual double alan() const;
   virtual double hacim() const;
   virtual void yazSekilAd() const {cout << "Silindir: ";}
   virtual void yaz() const;
private:
   double yuk;
};

#endif
```

1. `Silindir` tanımı (türev sınıfı)

# 20.8 Örnek: Kalıtsal Arayüz ve Uygulama

```
// Fig. 20.1: silindir1.cpp
// Silindir için Üye ve friend fonksiyon tanımları
#include <iostream>

using std::cout;

#include "silindir1.h"

Silindir::Silindir( double h, double r, int x, int y )
   : Daire( r, x, y )
{ kurYuk( h ); }

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

double Silindir::alYuk() { return yuk; }

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

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

void Silindir::yaz() const
{
   Daire::yaz();
   cout << "; Yükseklik= " << yuk;
}
```

# 20.8 Örnek: Kalıtsal Arayüz ve Uygulama

```
// Fig. 20.1: fig20_01.cpp
// Şekil nokta daire ve silindir hiyerarşisi için sürücü
#include <iostream>

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

#include <iomanip>

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

#include "sekil.h"
#include "nokta1.h"
#include "daire1.h"
#include "silindir1.h"

void PointerileSanal( const Sekil * );
void RefileSanal( const Sekil & );

int main()
{
```

# 20.8 Örnek: Kalıtsal Arayüz ve Uygulama

```
   cout << setiosflags( ios::fixed | ios::showpoint )
        << setprecision( 2 );
   Nokta nokta( 7, 11 );                    // nokta yarat
   Daire daire( 3.5, 22, 8 );
   Silindir silindir( 10, 3.3, 10, 10 );

   nokta.yazSekilAd();    // static bağlama
   nokta.yaz();             // static bağlama
   cout << '\n';

   daire.yazSekilAd();   // static bağlama
   daire.yaz();            // static bağlama
   cout << '\n';

   silindir.yazSekilAd(); // static bağlama
   silindir.yaz();          // static bağlama
   cout << "\n\n";

   Sekil *sekillerinDizisi[ 3 ];  // baz-class pointer dizisi

   sekillerinDizisi[ 0 ] = &nokta;

   sekillerinDizisi[ 1 ] = &daire;

   sekillerinDizisi[ 2 ] = &silindir;

// sekillerinDizisi ni tara ve PointerIleSanal ı çağırarak
   // dinamik bağlama kullanarak herbir nesnenin şekil adını
   // durumunu, alanını, ve hacmini yaz
```

# 20.8 Örnek: Kalıtsal Arayüz ve Uygulama

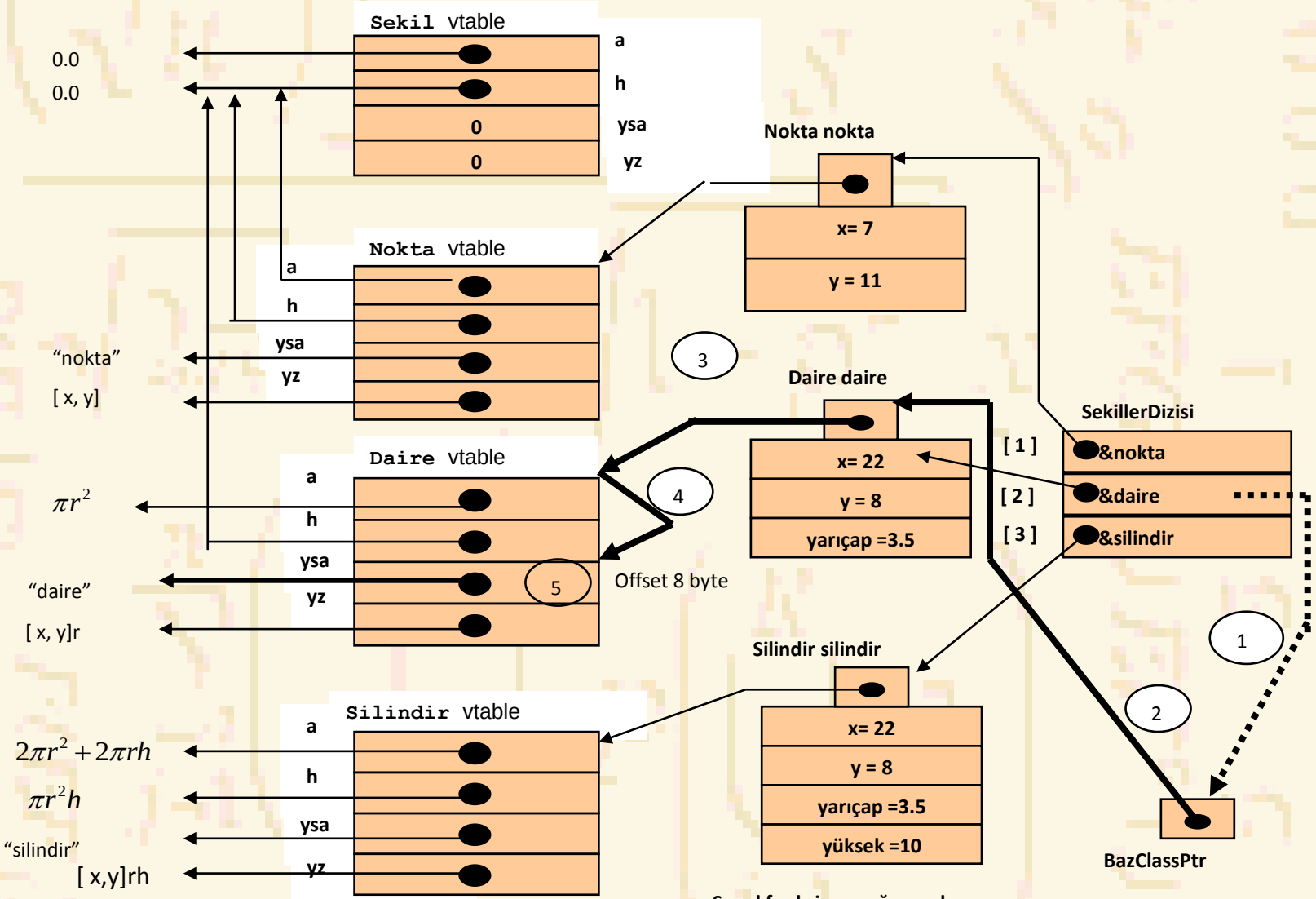
```
cout << "Baz-class pointerlar ile "
        << "sanal(virtual) fonksiyon çağrımı\n";

   for ( int i = 0; i < 3; i++ )
      PointerileSanal( sekillerinDizisi[ i ] );
   cout << "Baz-class referanslar ile"
        << "sanal fonksiyon çağrımı\n";

   for ( int j = 0; j < 3; j++ )
      RefileSanal( *sekillerinDizisi[ j ] );

   return 0;
}
// baz-class pointer ile dinamik bağlama
void PointerileSanal( const Sekil *bazClassPtr )
{
   bazClassPtr->yazSekilAd();
   bazClassPtr->yaz();
   cout << "\nAlan = " << bazClassPtr->alan()
        << "\nHacim= " << bazClassPtr->hacim() << "\n\n";
} // fonks sonu

// baz-class referans ile dinamik bağlama

void RefileSanal( const Sekil &bazClassRef )
{
   bazClassRef.yazSekilAd();
   bazClassRef.yaz();
   cout << "\nAlan= " << bazClassRef.alan()
        << "\nHacim= " << bazClassRef.hacim() << "\n\n";
} // fonks sonu
```

# 20.8 Örnek: Kalıtsal Arayüz ve Uygulama

```
Nokta: [7, 11]
Daire: [22, 8]; Yarıçap= 3.50
Silindir: [10, 10]; Yarıçap= 3.30; Yükseklik= 10.00
Baz-class pointerlar ile sanal(virtual) fonksiyon çağrımı
Nokta: [7, 11]
Alan = 0.00
Hacim = 0.00
Daire: [22, 8]; Yarıçap= 3.50
Alan= 38.48
Hacim = 0.00
Silindir: [10, 10]; Yarıçap= 3.30; Yükseklik= 10.00
Alan= 275.77
Hacim = 342.12
Baz-class referanslar ile sanal fonksiyon çağrımı
Nokta: [7, 11]
Alan = 0.00
Hacim = 0.00
Daire: [22, 8]; Yarıçap= 3.50
Alan= 38.48
Hacim = 0.00
Silindir: [10, 10]; Yarıçap= 3.30; Yükseklik= 10.00
Alan= 275.77
Hacim = 342.12
```

- Polimorfizm ne zaman kullanılır
  - Az bellek kullanımı ve hızlı çalışma zamanı gerekli ise
  - Performansı artırmak için STL (Standard Template Library) de kullanılmamalı
- **`virtual`** fonksiyon tablosu (vtable)
  - **`virtual`** fonksiyonlu her sınıf bir vtable- a sahiptir
  - Her **`virtual`** fonksiyon için, vtable uygun fonksiyona bir pointer-a sahiptir
    - Türev sınıfı baz sınıfı olarak aynı fonksiyona sahipse, bu durumda fonksiyon pointer-ı baz fonksiyonunu işaret eder
  - Detaylı açıklama aşağıdadır:

# 20.9 Polimorfizm, `virtual` Fonksiyonlar ve Dinamik Birleştirme




**a** = **alan** fonksiyonu **h** = **hacim** fonksiyonu
**ysa** = **yazSekilAdi** fonksiyonu **yz = yaz** fonksiyonu
**0** giriş somut virtual fonksiyonu anlamında
x= **yaricap** ; h = **yuksek**

**Sanal fonksiyon çağrımı akışı**
**BazClasssPtr ->yazSekilAdi( );** koyu oklarla gösterilmiştir
**1** **&daire** yi **bazClassPtr** ye aktar
**2** **daire** nesnesini al **3** **daire** vtable –ını al
**4** vtable daki **yazSekilAdi** pointer-ını al
**5 yazSekilAdi nı** Daire için çalıştır