Bölüm 16 - Sınıflar (Class) ve Veri Soyutlaması

İçerik

16.1 Giriş
16.2 Bir Zaman Soyut Veri Tipini bir Class ile Uygulama
16.3 Class Alanı ve Class Üyelerine Erişim
16.4 Bütünlemeden Arayüzü Ayırma
16.5 Üyelere Erişim Kontrolü
16.6 Erişim ve Yarar Fonksiyonları
16.7 Class Nesnelerini Belirleme: Oluşturucular (Constructors)
16.8 Oluşturucularla Default Argümentlerin Kullanımı
16.9 Yokedicilerin (Destructors) Kullanımı
16.10 Oluşturucular ve Yokediciler Çağrıldığında
16.11 Veri Üyelerini ve Üye Fonksiyonlarını Kullanma
16.12 Bir İnce Tuzak: Özel (`private`) bir Veri Üyesine bir Referans Gönderme
16.13 Default Üyebazlı Kopyayla Atama
16.14 Yazılım Kullanılabilirliği

16.1 Giriş

- Nesne -Tabanlı programlama (OOP)
  - Verileri (nesne) ve fonksiyonları (davranış) *class* adı verilen paketlere *yükleme*
  - Veri ve fonksiyonlar yakın ilişkiye sahiptir
- Bilgi gizleme
  - Uygulama detayları class-ların içinde gizlidir
- C++ programlama birimi: class
  - class bir mavi kopya (plan) gibidir– tekrar kullanılabilirdir
  - Nesneler class-lardan oluşturulur
  - Örneğin, bir ev "mavi kopya class" dan oluşturulur
  - C programcıları fonksiyonlara konsantre olurlar

16.2 Bir Zaman Soyut Veri Tipini bir Class ile Uygulama

- Class
  - Niteliklere (veri üyeleri) ve davranışlara (üye fonksiyonları) sahip olan model nesneleri
  - **`class`** anahtar kelimesi ile tanımlanır

```
1  class Zaman {
2  public:
3     Zaman();
4     void kurZaman( int, int, int );
5     void yazAskeri();
6     void yazStandard();
7  private:
8     int saat;      // 0 - 23
9     int dakika;    // 0 - 59
10    int saniye;    // 0 - 59
11 };
```

`Public:` ve **`Private:`** üye erişim belirteçleridir

kurZaman, **yazAskeri**, ve **yazStandard** üye fonksiyonlardır.
Zaman oluşturucudur.

saat, **dakika**, ve **saniye** veri üyeleridir.

16.2 Bir Zaman Soyut Veri Tipini bir Class ile Uygulama

- Format
  - Gövde küme parantezleri (`{` ve `}`) içinde yazılır
  - Class tanımı noktalıvirgül ile biter

- Üye fonksiyonlar ve veriler

  `Public` - Programın **Zaman** class nesnesine her erişitiğinde erişilebilirdir

  `Private` – sadece class-ın üye fonksiyonlarına erişilebilirdir

  `Protected` – sonra tartışılacak

16.2 Bir Zaman Soyut Veri Tipini bir Class ile Uygulama

- Oluşturucu
  - Bir class nesnesinin veri üyelerini belirten özel üye fonksiyondur
  - Değer göndermez
  - class ile aynı ada sahiptir
- Deklarasyonlar
  - class tanımlandıktan sonra bir veri tipi olarak kullanılabilirdir

```
Zaman  aksam,                      //  Zaman tipi nesne
       zamanDizisi[ 5 ],           //  Zaman nesnesinin dizisi
       *zamanaPointer,             //  nesneye pointer
       &yemekZamanı= aksam;        //  bir Zaman nesnesine referans
```

Not: class adı yeni tip bir belirteç oldu.

16.2 Bir Zaman Soyut Veri Tipini bir Class ile Uygulama

Binary hedef karar operatörü (: :)

Üye fonksiyona hangi class-ın sahip olduğunu belirtir

Farklı class-ların üye fonksiyonlar aynı ada sahip olabilir

Tanım class-ı üye fonksiyonları için format

Return_Tipi ClassAdı : : *ÜyeFonksiyonAdı*( ){

...

}

Eğer üye fonksiyon class-ın *içinde* tanımlanmış ise

- Hedef karar operatörü ve class adına gerek yoktur
- Bir fonksiyonu class dışında tanımlamak **`public`** veya **`private`** olmasını değiştirmez

Class-lar yazılımın tekrar kullanımını teşvik eder

- Kalıtsallık özelliği yeni class-ların eskilerinden üretilmesine izin verir

Aşağıdaki programda

- **Zaman** oluşturucusu veri üyelerini sıfırlar
  - Oluşum aşamasında nesnenin kararlı konumda olmasına emin olunuz

```
1  // Fig. 16.2: fig16_02.cpp
2  // Zaman class-ı.
3  #include <iostream>
4
5  using std::cout;
6  using std::endl;
7
8  // zaman soyut veri tipi tanımı
9  class Zaman {
10 public:
11    Zaman();                              // oluşturucu
12    void kurZaman( int, int, int ); // saat dakika ve saniyeyi kur
13    void yazAskeri();              // askeri zaman formatında yaz
14    void yazStandard();            // standard zaman formatında yaz
15 private:
16    int saat;      // 0 - 23
17    int dakika;    // 0 - 59
18    int saniye;    // 0 - 59
19 };
20
21 // Zaman  oluşturucu  her bir veri üyesini sıfır alır.
22 // tüm Zaman nesnelerinin kararlı durumda olduğuna garantile.
23 Zaman::Zaman() { saat = dakika = saniye = 0; }
24
25 // Askeri zamanı kullanarak yeni bir zaman değeri gir.  Ğeçerliliğini kontrol et
26 // Geçersiz değerleri sıfır yap
27 void Zaman::kurZaman( int s, int d, int sn )
28 {
29    saat = ( s >= 0 && s < 24 ) ? s : 0;
30    dakika = ( d >= 0 && d < 60 ) ? d : 0;
31    saniye= ( sn >= 0 && sn < 60 ) ? sn : 0;
32 }
```

le Uygulama

```
33
34 // Askeri formatta zamanı taz
35 void Zaman::yazAskeri()
36 {
37    cout << ( saat < 10 ? "0" : "" ) << saat << ":"
38         << ( dakika < 10 ? "0" : "" ) << dakika;
39 }
40
41 // Standart formatta zamanı yaz
42 void Zaman::yazStandard()
43 {
44    cout << ( ( saat == 0 || saat == 12 ) ? 12 : saat % 12 )
45         << ":" << ( dakika < 10 ? "0" : "" ) << dakika
46         << ":" << ( saniye < 10 ? "0" : "" ) << saniye
47         << ( saat < 12 ? " AM" : " PM" );
48 }
49
50 // Test et
51 int main()
52 {
53    Zaman t;  // Zaman sınıfının t nesnesini belirle
54
55    cout << "İlk askeri zaman: ";
56    t.yazAskeri();
57    cout << "\n İlk standart zaman: ";
58    t.yazStandard();
59
```

16.2 Bir Zaman Soyut Veri Tipini bir Class ile Uygulama

```
60     t.kurZaman( 13, 27, 6 );
61     cout << "\n\n kurZaman dan sonra askeri zaman: ";
62     t.yazAskeri();
63     cout << "\n kurZaman dan sonra standart zaman: ";
64     t.yazStandard();
65
66     t.kurZaman( 99, 99, 99 );  // yanlış girdi dene
67     cout << "\n\n Yanlış girdiden sonra:"
68          << "\nAskeri zaman: ";
69     t.yazAskeri();
70     cout << "\nStandart zaman: ";
71     t.yazStandard();
72     cout << endl;
73     return 0;
74 }
```

```
İlk askeri zaman: 00:00
İlk standart zaman: 12:00:00 AM

KurZaman dan sonra askeri zaman: 13:27
KurZaman dan sonra standart zaman: 1:27:06 PM

Yanlış girdiden sonra:
Askeri zaman: 00:00
Standard zaman: 12:00:00 AM
```

16.3 Class Alanı(Hedefi) ve Class Üyelerine Erişim

- Class alanı
  - Veri üyeleri ve Veri Fonksiyonlarına Erişim
- Dosya Alanı
  - Üye olmayan fonksiyonlar
- Fonksiyon Alanı
  - Üye fonksiyonlarda tanımlanan değişkenler, fonksiyon işi bitince yokedilirler
- Alan İçi
  - Tüm üye fonksiyonlarca erişilebilinen üyeler
  - İsim ile refere edilirler

16.3 Class Alanı(Hedefi) ve Class Üyelerine Erişim

- Alan dışı
  - Bir kulp kullan
    - Bir nesne adı, nesneye pointer veya nesneye referans
- Class üyelerine erişim
  - **`struct`** – daki gibidir
  - Nesneler için nokta (**`.`**) veya pointer-lar için ok (**`->`**)
  - Örnek: **`t.saat,`** **`t`** nin saat elemanıdır
  - **`zamanPtr->saat`** saat elemanıdır

```
1  // Fig. 16.3: fig16_03.cpp
2  // class üyesine erişim operatörleri . ve -> kullanımı
3  //
4  // UYARI: BUNDAN SONRAKİ ÖRNEKLERDE KULLANICI VERİSİ YOK!
5  #include <iostream>
6
7  using std::cout;
8  using std::endl;
9
10 // basit class Sayac
11 class Sayac {
12 public:
13    int x;
14    void yaz() { cout << x << endl; }
15 };
16
17 int main()
18 {
19    Sayac sayac,                    // sayac nesnesi oluştur
20          *sayacPtr = &sayac, // sayaca pointer
21          &sayacRef = sayac;  // sayaca referans
22
23    cout << "x e 7 ata ve nesne adını kullanarak yaz: ";
24    sayac.x = 7;          // veri üyesi x e 7 ata
25    sayac.yaz();       // yaz üye fonksiyonunu çağır
26
27    cout << "x e 8 ata ve referans kullanarak yaz: ";
28    sayacRef.x = 8;      // veri üyesi x e 8 ata
29    sayacRef.yaz();  // call member function print
30
```

16.3 Class Alanı(Hedefi) ve Class Üyelerine Erişim

```
31      cout << "x e 10 ata  ve pointer kullanarak yaz: ";
32      sayacPtr->x = 10;   // veri üyesi x e 10 ata
33      sayacPtr->yaz(); // yaz üye fonksiyonunu çağır
34      return 0;
35  }
```

```
x e 7 ata ve   nesne adı kullanarak yaz:  7
x e 8 ata ve   referans  kullanarak yaz:  : 8
x e 10 ata ve   pointer kullanarak yaz: 10
```

16.4 Bütünlemeden Arayüzü Ayırma

- Bütünlemeden arayüzü ayırma
  - Program düzenlemelerini kolaylaştırır
  - C++ programları iki parçaya ayrılabilir:

    Header dosyaları– class tanımları ve fonksiyon prototipleri içerir

    Kaynak-kod dosyaları – üye fonksiyon tanımlarını içerir

- Program taslağı:
  - Önceki **`Zaman`** sınıfını kullanarak, header dosyası oluştur
  - Bir kaynak kod dosyası oluştur
    - Sınıf tanımlarını elde etmek için header dosyasını yükle
    - Sınıfın üye fonksiyonlarını tanımla

```
1  // Fig. 16.4: zaman1.h
2  // Zaman sınıfının deklarasyonu.
3  // üye fonksiyonlar zaman1.cpp de tanımlandı
4
5  // header dosyasının çoklu kullanılmasını engelle
6  #ifndef ZAMAN1_H
7  #define ZAMAN1_H
8
9  // soyut veri tipi tanımı Zaman
10 class Zaman {
11 public:
12    Zaman();                              // constructor
13    void kurZaman( int, int, int ); // saat, dakika saniyeyi kur
14    void yazAskeri();              // askeri formatta yaz
15    void yazStandard();            // standart formatta yaz
16 private:
17    int saat;      // 0 - 23
18    int dakika;    // 0 - 59
19    int saniye;    // 0 - 59
20 };
21
22 #endif
```

```
23 // Fig. 16.4: zaman1.cpp
24 // Zaman sınıfı için üye fonksiyonların tanımı.
25 #include <iostream>
26
27 using std::cout;
28
29 #include "zaman1.h"
30
31 // Zaman oluşturucusu her bir veri üyesini sıfırlar.
32 // Tüm Zaman üyelerinin uygun değerlerle başlamasını garantile.
33 Zaman::Zaman() { saat = dakika = saniye = 0; }
34
35 // Askeri zaman kullanarak yeni bir Zaman değeri kur. Veri üyelerinin
36 // geçerliliğini test et. Geçersiz değerleri sıfırla.
37 void Zaman::kurZaman( int s, int d, int sn )
38 {
39    saat   = ( s >= 0 && s < 24 ) ? s : 0;
40    dakika = ( d >= 0 && d < 60 ) ? d : 0;
41    saniye = ( sn >= 0 && sn < 60 ) ? sn : 0;
42 }
43
44 // Askeri formatta zamanı yaz
45 void Zaman::yazAskeri()
46 {
47    cout << ( saat < 10 ? "0" : "" ) << saat << ":"
48         << ( dakika < 10 ? "0" : "" ) << dakika;
49 }
```

16.4 Bütünlemeden Arayüzü Ayırma

```
50
51 // Standart formatta zamanı yaz
52 void Zaman::yazStandard()
53 {
54    cout << ( ( saat == 0 || saat == 12 ) ? 12 : saat % 12 )
55         << ":" << ( dakika < 10 ? "0" : "" ) << dakika
56         << ":" << ( saniye < 10 ? "0" : "" ) << saniye
57         << ( saat < 12 ? " AM" : " PM" );
58 }
```

```
59 // Fig. 16.4: fig16_04.cpp
60 // Zaman1 sınıfı için sürücü
61 // NOT: zaman1.cpp ile birlikte derle
62 #include <iostream>
63
64 using std::cout;
65 using std::endl;
66
67 #include "zaman1.h"
68
69 // basit Zaman sınıfı testi için sürücü
70 int main()
71 {
72    Zaman t;  // zaman sınıfı t nesnesini sabitle
73
74    cout << "İlk askeri zaman: ";
75    t.yazAskeri();
76    cout << "\n İlk standart zaman:";
77    t.yazStandard();
78
79    t.kurZaman( 13, 27, 6 );
80    cout << "\n\n kurZaman dan sonra askeri zaman:";
81    t.yazAskeri();
82    cout << "\n kurZamandan sonra Standard zaman:";
83    t.yazStandard();
84
```

16.4 Bütünlemeden Arayüzü Ayırma

```
85      t.kurZaman( 99, 99, 99 );  // hatalı veri girişi dene
86      cout << "\n\n Yanlış girdiden sonra:\n"
87           << "Askeri zaman: ";
88      t.yazAskeri();
89      cout << "\nStandard zaman: ";
90      t.yazStandard();
91      cout << endl;
92      return 0;
93  }
```

```
İlk askeri zaman: 00:00
İlk standart zaman: 12:00:00 AM

kurZamandan sonra askeri zaman: 13:27
kurZamandan sonra standart zaman: 1:27:06 PM

Yanlış girdiden sonra:
Askeri zaman: 00:00
Standard Zaman: 12:00:00 AM
```

16.5 Üyelere Erişim Kontrolü

`public` –in amacı

Kullanıcıya sınıfın sağladığı servisleri gösterir (arayüz)

`private` –ın amacı

Default kurulum

Sınıfın görevleri nasıl yaptığının detaylarını gizler (bütünleme)

`Private` üyelere sadece **`public`** üye fonksiyonlar kullanılarak **`public`** arayüzü ile ulaşılabilir

```
1  // Fig. 16.5: fig16_05.cpp
2  // Özel sınıf üyelerine erişimi denemede
3  // ortaya çıkan hata.
4  #include <iostream>
5
6  using std::cout;
7
8  #include "zaman1.h"
9
10 int main()
11 {
12    Zaman t;
13
14    // Hata: 'Zaman::saat' e erişilemez
15    t.saat = 7;
16
17    // Hata: 'Zaman::dakika' ya erişilemez
18    cout << "dakika = " << t.dakika;
19
20    return 0;
21 }
```

```
Compiling...
Fig06_06.cpp
D:\Fig06_06.cpp(15) : error C2248: 'saat' : cannot access private
member declared in class 'Zaman'
D:\Fig6_06\zaman1.h(18) : see declaration of 'saat'
D:\Fig06_06.cpp(18) : error C2248: 'dakika' : cannot access private
member declared in class 'Zaman'
D:\zaman1.h(19) : see declaration of 'dakika'
Error executing cl.exe.

test.exe - 2 error(s), 0 warning(s)
```

Yarar (Utility) fonksiyonları

- Public fonksiyonların operasyonlarını destekleyen **`private`** fonksiyonlardır
- Kullanıcı tarafından doğrudan kullanıma yönelik değildir

Erişim fonksiyonları

- Veri okuyup/görüntüleyen veya koşulları kontrol eden **`public`** fonksiyonlardır
- Bir içerici için, **`isEmpty`** fonksiyonunu çağırabilir

Örnek

- Aylık satışları alan ve toplamı veren bir program
- Sadece erişim fonksiyonları gösteriliyor

16.6 Erişim ve Yarar Fonksiyonları

```
87 // Fig. 16.6: fig16_06.cpp
88 // bir yarar fonksiyonu kullanımı
89 // satisele.cpp ile kullan
90 #include "satisele.h"
91
92 int main()
93 {
94    SatisEleman s;           // bir s SatisEleman nesnesi oluştur
95
96    s.kullanSatisFormAl();   // basit dizisel kod
97    s.yazYillikSatis();      // main de kontrol yapısı yok
98    return 0;
99 }
```

```
1. ay için satış miktarını gir: 5314.76
2. ay için satış miktarını gir: 4292.38
3. ay için satış miktarını gir: 4589.83
4. ay için satış miktarını gir: 5534.03
5. ay için satış miktarını gir: 4376.34
6. ay için satış miktarını gir: 5698.45
7. ay için satış miktarını gir: 4439.22
8. ay için satış miktarını gir: 5893.57
9. ay için satış miktarını gir: 4909.67
10. ay için satış miktarını gir: 5123.45
11. ay için satış miktarını gir: 4024.97
12. ay için satış miktarını gir: 5923.92

Toplam yıllık satış: $60120.59
```

16.7 Class Nesnelerini Belirleme: Oluşturucular (Constructors)

Constructor fonksiyonları

- Sınıf üyelerine ilk atamaları yapabilir
- Sınıf ile aynı ada sahiptir, return tipi yoktur
- Üye fonkisyonların ilk değerleri constructor ile atanabilir veya sonradan verilebilir

Nesnelerin deklarasyonu

- İlk değerler sağlanabilir
- İlk değerler sınıf oluşturucusuna argüment olarak geçer

16.7 Class Nesnelerini Belirleme: Oluşturucular (Constructors)

- Format

  Tip NesneAdı **(** *değer1*, *değer2*, ...**);**

  – Constructor üye değişkenlerine *değer1*, *değer2*, vs. atamalarını yapar

  – Yeterince değer belirtilmemişse, en sağdaki değerler (programcının belirlediği) default değer alır

  – **Sınıfım Nesnem(3, 4.0);**

16.8 Oluşturucularla Default Argümentlerin Kullanımı

- Default constructor
  - Her sınıf için bir tane
  - Argümentsiz olabilir
  - default argumentlere sahiptir
- Default argumentler
  - Default constructor fonksiyon prototipinde belirtilir (class tanımında)
    - Sınıf dışındaki fonksiyon tanımında default değerler yazılmaz
  - Örnek:

```
ÖrnekSınıf( int = 0, float = 0);
```

    - Constructor, sınıf ile aynı ada sahiptir

```
1  // Fig. 16.7: zaman2.h
2  // Zaman  sınıfı deklarasyonu.
3  // Üye fonksiyonlar  zaman2.cpp de
4
5  // önişlemci komutları
6  // header dosyasının çoklu kullanımını engeller
7  #ifndef ZAMAN2_H
8  #define ZAMAN2_H
9
10 // Zaman soyut veri tipi deklarasyonu
11 class Zaman {
12 public:
13    Zaman( int = 0, int = 0, int = 0 );  // default constructor
14    void kurZaman( int, int, int ); // saat, dakika ve saniyeyi kur
15    void yazAskeri();           // Askeri formatta yazar
16    void yazStandard();           // Standard formatta yazar
17 private:
18    int saat;      // 0 - 23
19    int dakika;    // 0 - 59
20    int saniye;    // 0 - 59
21 };
22
23 #endif
```

```
61 // Fig. 16.7: fig16_07.cpp
62 // default constructor
63 // Zaman sınıfı için fonksiyon.
64 #include <iostream>
65
66 using std::cout;
67 using std::endl;
68
69 #include "zaman2.h"
70
71 int main()
72 {
73    Zaman t1,              // tüm argümentler default-landı
74          t2(2),           // dakika ve saniye default-landı
75          t3(21, 34),      // saniye default-landı
76          t4(12, 25, 42), // tüm değerler belirlendi
77          t5(27, 74, 99); // tüm geçersiz değerler belirlendi
78
79    cout << "Oluşum:\n"
80         << "tüm argümentler default:\n   ";
81    t1.yazAskeri();
82    cout << "\n   ";
83    t1.yazStandart();
84
85    cout << "\n saat verildi; dakika ve saniye default:"
86         << "\n   ";
87    t2.yazAskeri();
88    cout << "\n   ";
89    t2.yazStandard();
90
91    cout << "\nsaat ve dakika verildi; saniye default:"
92         << "\n   ";
93    t3.yazAskeri();
```

```
94      cout << "\n    ";
95      t3.vazStandard();
96
97      cout << "\nsaat, dakika ve sanive verildi:"
98           << "\n    ";
99      t4.vazAskeri();
100     cout << "\n    ";
101     t4.vazStandard();
102
103     cout << "\ngecersiz değerler verildi:"
104          << "\n    ";
105     t5.vazAskeri();
106     cout << "\n    ";
107     t5.vazStandard();
108     cout << endl;
109
110     return 0;
111 }
```

```
Oluşum:
tüm argümentler default:
   00:00
   12:00:00 AM
saat verildi; dakika ve saniye default:
   02:00
   2:00:00 AM
saat ve dakika verildi; saniye default:
   21:34
   9:34:00 PM
saat, dakika ve saniye verildi:
   12:25
   12:25:42 PM
geçersiz değerler verildi:
   00:00
   12:00:00 AM
```

16.9 Yokedici (Destructor) Kullanımı

Destructor

- Sınıfın üye fonksiyonu
- Sistem nesne belleği istemeden sonlandırmayı sağlar
- constructor –ın tümleyenidir
- Sınıf adından önce *tilda* (**~**) yazılır

```
~Zaman
```

  constructor adının sınıf adı olduğunu hatırlayınız

- Hiç bir parameter almaz, değer göndermez
- Her sınıf için bir yokedici- çoklu yüklemeye izin yok

16.10 Oluşturucu ve Yokediciler Çağrıldığında

- Oluşturucu ve yokediciler otomatik olarak çağrılır
  - Sıra, nesnelerin hedefine bağlıdır
- Global hedef nesneleri
  - Oluşturucular diğer fonksiyonlardan önce çağrılır (**`main`** dahil)
  - Yokediciler, **`main`** bittikten sonra (veya **`exit`** fonksiyonu çağrıldıktan sonra) çağrılır
  - Eğer program **`abort`** ile sonlanırsa yokedici çağrılmaz

Otomatik yerel nesneler

- Nesneler tanımlandığında oluşturucu çağrılır
- Yokedici nesne hedefi terkettiğinde çağrılır (tanımlandıkları bloktan çıkarken)
- Eğer program **`exit`** veya **`abort`** ile sonlanırsa yokedici çağrılmaz

`static` yerel nesneler

- Oluşturucular, program çalışması nesnelerin tanımlandığı noktaya ulaştığında çağrılır
- Yokediciler, **`main`** bittiğinde veya **`exit`** fonksiyonu çağrıldığında çağrılır
- Program **`abort`** ile sonlandığında yokedici çağrılmaz

16.10 Oluşturucu ve Yokediciler Çağrıldığında

```
1  // Fig. 16.8: olustur.h
2  // OlusturVeYoket sınıfı tanımı.
3  // Üye fonksiyonlar olustur.cpp de tanımlı
4  #ifndef OLUSTUR_H
5  #define OLUSTUR_H
6
7  class OlusturVeYoket {
8  public:
9   OlusturVeYoket( int );  // constructor
10    ~ OlusturVeYoket();      // destructor
11 private:
12    int data;
13 };
14
15 #endif
```

16.10 Oluşturucu ve Yokediciler Çağrıldığında

```
16 // Fig. 16.8: olustur.cpp
17 // OlusturVeYoket sınıfı için üye fonksiyonlar tanımı
18 #include <iostream>
19
20 using std::cout;
21 using std::endl;
22
23 #include "olustur.h"
24
25  OlusturVeYoket :: OlusturVeYoket( int deger )
26 {
27    data = deger;
28    cout << "Nesne " << data << "   constructor";
29 }
30
31  OlusturVeYoket ::~ OlusturVeYoket()
32    { cout << "Nesne " << data << "   destructor " << endl; }
```

```
// Fig. 16.8: fig16_08.cpp
// Oluşturucu ve Yokedicilerin çağrılma sırası
// gösterimi.
#include <iostream>

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

#include "olustur.h"

void olustur( void );   // prototip

 OlusturVeYoket ilk( 1 );  // global nesne

int main()
{
   cout << "   (global main den önce oluşturuldu)" << endl;

 OlusturVeYoket ikinci( 2 );         // yerel nesne
   cout << "   (main-de yerel otomatik)" << endl;

   static OlusturVeYoket ucuncu( 3 );  // yerel nesne
   cout << "   (main-de yerel static)" << endl;

   olustur();  // nesne oluşturmak için fonksiyonu çağır

 OlusturVeYoket dorduncu( 4 );         // yerel nesne
   cout << "   (main-de yerel otomatik)" << endl;
   return 0;
}
```

16.10 Oluşturucu ve Yokediciler Çağrıldığında

```
63
64 // Nesneler oluşturmak için fonksiyonlar
65 void olustur( void )
66 {
67  OlusturVeYoket besinci( 5 );
68    cout << "   (olustur-da yerel otomatik)" << endl;
69
70    static OlusturVeYoket altinci( 6 );
71    cout << "   (olustur-da yerel static)" << endl;
72
73  OlusturVeYoket yedinci( 7 );
74    cout << "   (olustur-da yerel otomatik)" << endl;
75 }
```

```
Nesne 1   constructor   (global main den önce oluşturuldu)
Nesne 2   constructor   (main-de yerel otomatik)
Nesne 3   constructor   (main-de yerel static)
Nesne 5   constructor   (olustur-da yerel otomatik)
Nesne 6   constructor   (olustur-da yerel static)
Nesne 7   constructor   (olustur-da yerel otomatic)
Nesne 7   destructor
Nesne 5   destructor
Nesne 4   constructor   (main-de yerel otomatik)
Nesne 4   destructor
Nesne 2   destructor
Nesne 6   destructor
Nesne 3   destructor
Nesne 1   destructor
```

16.11 Veri Üyelerini ve Üye Fonksiyonlarını Kullanma

- Sınıflar **`public`** üye fonksiyonlar sağlarlar
  - **`private`** veri üyelerinin değerlerini kur (yani yaz) veya al (yani oku)
  - Üye fonksiyon **`faizHesapla`** (**`BankaHesabi`** sınıfının bir **`private`** veri üyesi) ile bir banka hesabını ayarlama

- Adlandırma
  - **`faizOrani`** nı *kuran* üye fonksiyon için tipik bir isim **`kurFaizOrani`** alınabilir
  - **`faizOrani`** nı *alan* üye fonksiyon için tipik bir isim **`alFaizOrani`** alınabilir

16.11 Veri Üyelerini ve Üye Fonksiyonlarını Kullanma

- *Kurma* ve *alma* kapasitelerini etkili kullanma veri üyelerini **`public`** yapar mı?
  - Hayır!
  - Fonksiyonun ne kuracağına ve hangi bilgileri alacağına programcı karar verir

- **`public`** kur fonksiyonu
  - Veri üyelerini düzenleme girişimini kontrol etmeli,
  - O veri için yeni değerin uygunluğunu sağlamalı;
  - Örnek: ayın gününü 37 ye kurma girişimi engellenmeli.
  - Programcı bu özellikleri eklemelidir

16.12 Bir İnce Tuzak: Özel (private) bir Veri Üyesine bir Referans Gönderme

- Bir nesneye referans
  - Nesne adı ile aynı
  - Bir atama deyiminin sol tarafında kullanılabilir
  - Referans, orjinal nesneyi de değiştirebilen bir değer alabilir

- Bu kapasiteyi kullanmanın bir yolu (malesef!)
  - Bir **`private`** veri üyesine bir **`const`** olmayan referans gönderen bir sınıfın bir **`public`** üye fonksiyonunu al
  - Bu referans orjinal veriyi değiştirecek şekilde düzenlenebilir

16.12 Bir İnce Tuzak: Özel (private) bir Veri Üyesine bir Referans Gönderme

```
1  // Fig. 16.10: time4.h
2  // Zaman sınıfı deklarasyonu.
3  // üye fonksiyonlar zaman4.cpp de tanımlı
4
5  // önişlemci komutları
6  // header dosyasının çoklu kullanımını engeller
7  #ifndef ZAMAN4_H
8  #define ZAMAN4_H
9
10 class Zaman {
11 public:
12    Zaman( int = 0, int = 0, int = 0 );
13    void kurZaman( int, int, int );
14    int alSaat();
15    int &gecKurSaat( int );  // tehlikeli referans döner
16 private:
17    int saat;
18    int dakika;
19    int saniye;
20 };
21
22 #endif
```

```
23 // Fig. 16.10: zaman4.cpp
24 // Zaman sınıfı için üye fonksiyonların tanımı.
25 #include "zaman4.h"
26
27 // Private veri girmek için Constructor.
28 // değişkenleri kurmak için kurZaman fonksiyonunu çağırır.
29 // Default değerler 0 (class tanımına bakınız).
30 Zaman::Zaman( int sa, int dk, int sny )
31    { kurZaman( sa, dk, sny ); }
32
33 // saat dakika ve saniye değerlerini kur.
34 void Zaman::kurZaman( int s, int d, int sn )
35 {
36    saat   = ( s >= 0 && s < 24 ) ? s : 0;
37    minute = ( d >= 0 && d < 60 ) ? d : 0;
38    second = ( sn >= 0 && sn < 60 ) ? sn : 0;
39 }
40
41 // saat değerini al
42 int Zaman::alSaat() { return saat; }
43
44 // ZAYIF PROGRAMLAMA ÖRNEĞİ:
45 // Private veri üyesine bir referans gönderiyor.
46 int &Zaman::gecKurSaat( int sa )
47 {
48    saat = ( sa >= 0 && sa < 24 ) ? sa : 0;
49
50    return saat;  // tehlikeli referans dönüyor
51 }
```

```
52 // Fig. 16.10: fig16 10.cpp
53 // public üye fonksiyonu gösterimi
54 // özel veri üyesine referans gönderiyor.
55 // Bu örnekte Zaman sınıfı dağıtıldı.
56 #include <iostream>
57
58 using std::cout;
59 using std::endl;
60
61 #include "zaman4.h"
62
63 int main()
64 {
65    Zaman t;
66    int &saatRef = t.gecKurSaat( 20 );
67
68    cout << "Değiştirmeden önce saat: " << saatRef;
69    saatRef = 30;  // geçersiz değerle değiştirme
70    cout << "\nDeğişimden sonra saat: " << t.alSaat();
71
72    // Tehlikeli: Fonksiyon çağrımı
73    // bir sdeğer olarak kullanılabilecek bir referans gönderir!
74    t.gecKurSaat(12) = 74;
75    cout << "\n\n*********************************\n"
76         << "ZAYIF PROGRAMLAMA ÖRNEĞİ!!!!!!!!\n"
77         << "Sdeğer olarak gecKurSaat, Saat: "
78         << t.alSaat()
79         << "\n*********************************" << endl;
80
81    return 0;
82 }
```

16.12 Bir İnce Tuzak: Özel (private) bir Veri Üyesine bir Referans Gönderme

```
Deiğtirmeden önce saat: 20
Değişimden sonra saat: 30

**********************************
ZAYIF PROGRAMLAMA ÖRNEĞİ!!!!!!!!
Sdeğer olarak gecKurSaat, Saat: 74
**********************************
```

16.13 Default Üyebazlı Kopyayla Atama

- Atama operatörü (=)
  - Değişkenleri eşitler, yani, **`x = y;`**
  - Bir nesneyi aynı tipten bir nesneye atamak için kullanılabilir
  - Üyebazlı kopya — üyeden üyeye kopya

    `Nesne1= Nesne2;`

- Nesneler
  - Fonksiyon argümentleri olarak geçebilir
  - Fonksiyonlardan gönderilebilir (default-u ; değer-ile çağırma)
    - Referans ile çağırma için pointer kullan

16.13 Default Üyebazlı Kopyayla Atama

```
1  // Fig. 16.11: fig16 11.cpp
2  // default üyebazlı kopya kullanarak
3  // sınıf nesnelerinin birbirlerine atanması
4  #include <iostream>
5
6  using std::cout;
7  using std::endl;
8
9  // Basit Gün sınıfı
10 class Gun {
11 public:
12    Gun( int = 1, int = 1, int = 2004 ); // default constructor
13    void print();
14 private:
15    int ay;
16    int gun;
17    int yil;
18 };
19
20 // Hedef kontrolsüz basit Gun oluşturucusu
21 Gun::Gun( int a, int g, int y )
22 {
23    ay = a;
24    gun = g;
25    yil = y;
26 }
27
28 // aa-gg-yyyy formunda veriyi yaz
29 void Gun::print()
30    { cout << ay << '-' << gun << '-' << yil; }
```

16.13 Default Üyebazlı Kopyayla Atama

```
31
32 int main()
33 {
34    Gun gun1( 7, 4, 2003 ), gun2;  // g2 default-u  1/1/04
35
36    cout << "gun1 = ";
37    gun1.print();
38    cout << "\ngun2 = ";
39    gun2.print();
40
41    gun2 = gun1;   // default üyebazlı kopya ataması
42    cout << "\n\nDefault üyebazlı kopyadan sonra, gun2 = ";
43    gun2.print();
44    cout << endl;
45
46    return 0;
47 }
```

```
gun1 = 7-4-2003
gun2 = 1-1-2004

Default üyebazlı kopyadan sonra, gun2 = 7-4-2003
```

16.14 Software (Yazılım) Kullanılabilirliği

- Nesne-tabanlı programlama
  - Kullanışlı sınıflar oluşturma konsantrasyonu
- Sınıflara hakim olmak ve kataloglamak için olağanüstü fırsatlar
  - Her çaptan programcının erişebilirliği
  - Class kütüphaneleri bu amaç için oluşturulur
- Software
  - Varolan, iyi-tanımlı, dikkatlice test edilmiş, kullanışlı, geniş-çaplı ulaşılabilir parçalar
  - Güçlü ve yüksek kalite yazılımda hız gelişimi