BİL102 Nesne Yönelimli Programlama

Dr. Öğr. Üyesi Yavuz CANBAY

Kahramanmaraş Sütçü İmam Üniversitesi

Bilgisayar Mühendisliği Bölümü

›Yapıcılar-header files

Sınıf Yapısı

› Sınıf '{ }' işaretleri arasında tanımlanır ve bu tanım ';' işareti ile sonlandırılmalıdır.

```
1  class Time {
2  public:
3     Time();
4     void setTime( int, int, int );
5     void printMilitary();
6     void printStandard();
7  private:
8     int hour;      // 0 - 23
9     int minute;    // 0 - 59
10    int second;    // 0 - 59
11 };
```

Public: yada **Private:** erişim kontrolü içindir.

setTime, **printMilitary**, ve **printStandard** üye fonksiyonlardır.

Time ise **constructor**'dür.

Üye Erişim Kısıtlamaları

- Sınıf'lar kendi üyelerine (veri ve fonksiyonlarına) dışarıdan erişimi sınırlayabilirler.
  - Public: sınıf'a erişilen her yerde bu üyelere de erişilebilir.
  - Private: sadece sınıf'ın kendi üye fonksiyonları bu üyeye erişebilir.
  - Protected: private'a benzer. Daha sonra ayrıntılı olarak anlatılacak.

Yapıcılar (Constructor)

- Sınıf yapısının özel bir fonksiyonudur.
- Sınıf'la aynı isimdedir.
- Sınıf'tan bir nesne oluşturulurken çalıştırılır.
- Sınıf üyelerini hazırlar.
- Geri dönüş değeri olmaz ama parametre alabilir.
- Bir kez sınıf tanımlanınca, bir değişken tipi gibi kullanılır.
- Dizi, pointer yada normal bir değişken tanımlanabilir.

Yapıcı Örneği

```
1  // Fig. 6.3: fig06_03.cpp
2  // Time class.
3  #include <iostream>
4
5  using std::cout;
6  using std::endl;
7
8  // Time abstract data type (ADT) definition
9  class Time {
10 public:
11    Time();                          // constructor
12    void setTime( int, int, int ); // set hour, minute, second
13    void printMilitary();            // print military time format
14    void printStandard();            // print standard time format
15 private:
16    int hour;      // 0 - 23
17    int minute;    // 0 - 59
18    int second;    // 0 - 59
19 };
20
```

Yapıcı Örneği (devam)

```
21 // Time constructor initializes each data member to zero.
22 // Ensures all Time objects start in a consistent state.
23 Time::Time() { hour = minute = second = 0; }
24
25 // Set a new Time value using military time. Perform validity
26 // checks on the data values. Set invalid values to zero.
27 void Time::setTime( int h, int m, int s )
28 {
29    hour = ( h >= 0 && h < 24 ) ? h : 0;
30    minute = ( m >= 0 && m < 60 ) ? m : 0;
31    second = ( s >= 0 && s < 60 ) ? s : 0;
32 }
33
34 // Print Time in military format
35 void Time::printMilitary()
36 {
37    cout << ( hour < 10 ? "0" : "" ) << hour << ":"
38         << ( minute < 10 ? "0" : "" ) << minute;
39 }
```

: : operatörü class' ların fonksiyonlarına erişmek için kullanılıyor.

Yapıcı Örneği (devam)

```
41 // Print Time in standard format
42 void Time::printStandard()
43 {
44    cout << ( ( hour == 0 || hour == 12 ) ? 12 : hour % 12 )
45         << ":" << ( minute < 10 ? "0" : "" ) << minute
46         << ":" << ( second < 10 ? "0" : "" ) << second
47         << ( hour < 12 ? " AM" : " PM" );
48 }
49
50 // Driver to test simple class Time
51 int main()
52 {
53    Time t;  // instantiate object t of class Time
54
55    cout << "The initial military time is ";
56    t.printMilitary();
57    cout << "\nThe initial standard time is ";
58    t.printStandard();
59
```

Yapıcı Örneği (devam)

```
60     t.setTime( 13, 27, 6 );
61     cout << "\n\nMilitary time after setTime is ";
62     t.printMilitary();
63     cout << "\nStandard time after setTime is ";
64     t.printStandard();
65
66     t.setTime( 99, 99, 99 );  // attempt invalid settings
67     cout << "\n\nAfter attempting invalid settings:"
68          << "\nMilitary time: ";
69     t.printMilitary();
70     cout << "\nStandard time: ";
71     t.printStandard();
72     cout << endl;
73     return 0;
74 }
```

Yapıcı Örneği (devam)

```
The initial military time is 00:00
The initial standard time is 12:00:00 AM

Military time after setTime is 13:27
Standard time after setTime is 1:27:06 PM

After attempting invalid settings:
Military time: 00:00
Standard time: 12:00:00 AM
```

Yıkıcılar (Destructor)

- ‘Destructor’ fonksiyon sınıf isminin önüne ‘~’ işareti konularak isimlendirilir.
- Nesne yok edilmeden önce yapılması gerekenleri yapar.
- Fonsiyon aşırı-yüklemesi (overloading) yapılamaz. Parametresi olamaz.
- Görevi biten nesneyi yok eder.

```
class time {
    public:
    time();         // constructor
    ~time();        // destructor
    };
```

Binary Scope Resolution Operatörü (::)

› Sınıf'ın üyelerine erişirken sınıf isminin de kullanılabilmesini sağlar.

› Başka sınıf'ların da aynı isimli üyeleri olabileceği için gereklidir

```
dönüştipi ClassAdı::ÜyeFonksiyonAdı( )
    {
    ...
    }
```

Erişim Alanı

- **Class Scope;** Bir sınıfın veri ve fonksiyon üyeleri bu sınıf alanına aittir.
- **File Scope;** Üye olmayan fonksiyonlar bu alanda tanımlanır.
- **Function Scope;** Bir üye fonksiyon içinde tanımlanan değişkenler sadece tanımlandıkları fonksiyon tarafından bilinirler. Fonksiyon çağırıldığında oluşturulup, fonksiyondan çıkışta yok edilirler.

Üyelere Erişim

› Üyelere erişim aynı struct yapısındaki gibidir.

› Normalde '**.**', Pointer'larla ise '**->**' operatörleri kullanılır:

› ***t.hour*** yada ***timePtr->hour*** gibi.

Üyelere Erişim

```
1  // Fig. 6.4: fig06 04.cpp
2  // Demonstrating the class member access operators . and ->
3  //
4  // CAUTION: IN FUTURE EXAMPLES WE AVOID PUBLIC DATA!
5  #include <iostream>
6
7  using std::cout;
8  using std::endl;
9
10 // Simple class Count
11 class Count {
12 public:
13    int x;
14    void print() { cout << x << endl; }
15 };
16
17 int main()
18 {
```

public üye değişkenler çok nadiren kullanılır. Genelde gizlenen değişkenlere public fonksiyonlar ile erişim tercih edilerek bilgi gizlenir. Bu sayede nesne soyutlanır.

Üyelere Erişim (devam)

```
19    Count counter,                // create counter object
20          *counterPtr = &counter, // pointer to counter
21          &counterRef = counter;  // reference to counter
22
23    cout << "Assign 7 to x and print using the object's name: ";
24    counter.x = 7;          // assign 7 to data member x
25    counter.print();        // call member function print
26
27    cout << "Assign 8 to x and print using a reference: ";
28    counterRef.x = 8;       // assign 8 to data member x
29    counterRef.print();     // call member function print
30
31    cout << "Assign 10 to x and print using a pointer: ";
32    counterPtr->x = 10;     // assign 10 to data member x
33    counterPtr->print(); // call member function print
34    return 0;
35 }
```

```
Assign 7 to x and print using the object's name: 7
Assign 8 to x and print using a reference: 8
Assign 10 to x and print using a pointer: 10
```

Sınıf ve Program Bölümlerinin Ayrılması

- Sınıf yapısının tanımlanması bir başlık dosyasında (**header file**) yapılır.
- Üye fonksiyonlar, program dosyasında tanımlanır.
- Bu programların değiştirilmesini kolaylaştırır.

Örnek Header (Başlık) Dosyası

time1.h →

```
5  // prevent multiple inclusions of header file
6  #ifndef TIME1_H
7  #define TIME1 H
8
9  // Time abstract data type
10 class Time {
11 public:
12    Time();                          // constructor
13    void setTime( int, int, int ); // set hour, minute, second
14    void printMilitary();            // print military time format
15    void printStandard();            // print standard time format
16 private:
17    int hour;     // 0 - 23
18    int minute;   // 0 - 59
19    int second;   // 0 - 59
20 };
21
22 #endif
```

eğer **time1.h** (**TIME1_H**) tanımlı değilse (**#ifndef**) onu tanımla (**#define TIME1_H**). Eğer **TIME1_H** zaten tanımlı ise **#endif**'ten öncesi çalıştırılmaz.

Bu aynı header file'ın tekrar yüklenmesini önler.

Örnek Program Dosyası

```
23 // Fig. 6.5: time1.cpp
24 // Member function definitions for Time class.
25 #include <iostream>
26
27 using std::cout;
28
29 #include "time1.h"
30
31 // Time constructor initializes each data member to zero.
32 // Ensures all Time objects start in a consistent state.
33 Time::Time() { hour = minute = second = 0; }
34
35 // Set a new Time value using military time. Perform validity
36 // checks on the data values. Set invalid values to zero.
37 void Time::setTime( int h, int m, int s )
38 {
39    hour   = ( h >= 0 && h < 24 ) ? h : 0;
40    minute = ( m >= 0 && m < 60 ) ? m : 0;
41    second = ( s >= 0 && s < 60 ) ? s : 0;
42 }
```

#include ile sınıf'ın tanımlandığı dosyayı (time1.h) yüklüyoruz!

Örnek Program Dosyası (devam)

```
43
44 // Print Time in military format
45 void Time::printMilitary()
46 {
47    cout << ( hour < 10 ? "0" : "" ) << hour << ":"
48         << ( minute < 10 ? "0" : "" ) << minute;
49 }
50
51 // Print time in standard format
52 void Time::printStandard()
53 {
54    cout << ( ( hour == 0 || hour == 12 ) ? 12 : hour % 12 )
55         << ":" << ( minute < 10 ? "0" : "" ) << minute
56         << ":" << ( second < 10 ? "0" : "" ) << second
57         << ( hour < 12 ? " AM" : " PM" );
58 }
```

Fonksiyonların içeriği program dosyasında yazılı!

Üyelere Erişim Kısıtlamaları

- **Public**: bu bölümdeki veri ve fonksiyonlara erişilebilir.
- **Private**: bu bölümdekilere sadece üye yada **friend** fonksiyonlar üzerinden erişilebilir.
- Hiçbiri yazılmazsa her şey **private** kabul edilir.

Scope Örneği

```
1  // Fig. 6.6: fig06_06.cpp
2  // Demonstrate errors resulting from attempts
3  // to access private class members.
4  #include <iostream>
5
6  using std::cout;
7
8  #include "time1.h"
9
10 int main()
11 {
12    Time t;
13
14    // Error: 'Time::hour' is not accessible
15    t.hour = 7;
16
17    // Error: 'Time::minute' is not accessible
18    cout << "minute = " << t.minute;
19
20    return 0;
21 }
```

Scope Örneği (Devam)

```
Compiling...
Fig06_06.cpp
D:\Fig06_06.cpp(15) : error C2248: 'hour' : cannot access private
member declared in class 'Time'
D:\Fig6_06\time1.h(18) : see declaration of 'hour'
D:\Fig06_06.cpp(18) : error C2248: 'minute' : cannot access private
member declared in class 'Time'
D:\time1.h(19) : see declaration of 'minute'
Error executing cl.exe.

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

Hatırlatma

› Tüm Kodları kendi bilgisayarınızda uygulayın