



# BİL102 Nesne Yönelimli Programlama

Dr. Öğr. Üyesi Yavuz CANBAY  
Kahramanmaraş Sütçü İmam Üniversitesi  
Bilgisayar Mühendisliği Bölümü



---

# ›Konu 3. Nesnelerde Atama ve Kompozisyon

# Nesnelerde Atamalar

---

- › Aynı tipte (sınıf yapısında) iki nesne söz konusu olduğunda, bir nesne diğer nesneye '=' operatörü ile atanabilir.
- › Bu atama ile tüm üyelerin değerleri bir nesneden diğerine kopyalanmış olur.

# Nesnelerde Atamalar (devam)

```
9 // Simple Date class
10 class Date {
11 public:
12     Date( int = 1, int = 1, int = 1990 ); // default constructor
13     void print();
14 private:
15     int month;
16     int day;
17     int year;
18 };
19
20 // Simple Date constructor with no range checking
21 Date::Date( int m, int d, int y )
22 {
23     month = m;
24     day = d;
25     year = y;
26 }
27
28 // Print the Date in the form mm-dd-yyyy
29 void Date::print()
30 { cout << month << '-' << day << '-' << year; }
```

# Nesnelerde Atamalar (devam)

```
32 int main()
33 {
34     Date date1( 7, 4, 1993 ), date2; // d2 defaults to 1/1/90
35
36     cout << "date1 = ";
37     date1.print();
38     cout << "\ndate2 = ";
39     date2.print();
40
41     date2 = date1; // assignment by default memberwise copy
42     cout << "\n\nAfter default memberwise copy, date2 = ";
43     date2.print();
44     cout << endl;
45
46     return 0;
47 }
```

```
date1 = 7-4-1993
date2 = 1-1-1990
```

```
After default memberwise copy, date2 = 7-4-1993
```

# Kompozisyon (Composition)

---

- › Sınıflar, diğer sınıflardan nesne üyelere sahip olabilirler. Buna Composition denir
- › “Software reusability”nin en yaygın formu composition’dur

# Kompozisyon (Composition)-basit örnek

```
1  #include <iostream>
2
3  using namespace std;
4
5  class Tarih{
6  public:
7      Tarih();
8      int yil;
9      int ay;
10     int gun;
11     void print();
12 };
13
14 void Tarih::print() {
15     cout<<yil<<ay<<gun;
16 }
17
18 Tarih::Tarih(){ yil= 2020, ay = 1, gun = 1;}
19
```

```
20
21 class Personel{
22 public:
23     string adi;
24     string soyadi;
25     Tarih dogumtarihi;
26     void print();
27 };
28
29 void Personel::print() {
30     cout<<adi<<soyadi<<dogumtarihi.gun<<dogumtarihi.ay<<dogumtarihi.yil;
31 }
32
33 int main() {
34     Tarih x;
35     x.yil=1999;
36     x.ay=20;
37     x.gun=900;
38     x.print();
39
40     Personel y;
41     y.adi="sakir";
42     y.soyadi="ali";
43
44     y.print();
45
46
```

# Kompozisyon (Composition)

```
1 // Fig. 7.4: date1.h
2 // Declaration of the Date class.
3 // Member functions defined in date1.cpp
4 #ifndef DATE1_H
5 #define DATE1_H
6
7 class Date {
8 public:
9     Date( int = 1, int = 1, int = 1900 ); // default constructor
10    void print() const; // print date in month/day/year format
11    ~Date(); // provided to confirm destruction order
12 private:
13    int month; // 1-12
14    int day; // 1-31 based on month
15    int year; // any year
```



# Kompozisyon (Composition)

```
16
17 // utility function to test proper day for month and year
18 int checkDay( int );
19 };
20
21 #endif
22 // Fig. 7.4: date1.cpp
23 // Member function definitions for Date class.
24 #include <iostream>
25
26 using std::cout;
27 using std::endl;
28
29 #include "date1.h"
30
31 // Constructor: Confirm proper value for month;
32 // call utility function checkDay to confirm proper
33 // value for day.
```

# Kompozisyon (Composition)

```
34 Date::Date( int mn, int dy, int yr )
35 {
36     if ( mn > 0 && mn <= 12 )          // validate the month
37         month = mn;
38     else {
39         month = 1;
40         cout << "Month " << mn << " invalid. Set to month 1.\n";
41     }
42
43     year = yr;                          // should validate
44     day = checkDay( dy );               // validate the day
45
46     cout << "Date object constructor for date ";
47     print();                            // interesting: a print with no arguments
48     cout << endl;
49 }
50
```

Constructor  
çağrıldığında bu yazıyı  
yazacak

# Kompozisyon (Composition)

```
51 // Print Date object in form month/day/year
52 void Date::print() const
53 { cout << month << '/' << day << '/' << year; }
54
55 // Destructor: provided to confirm destruction order
56 Date::~~Date()
57 {
58     cout << "Date object destructor for date ";
59     print();
60     cout << endl;
61 }
62
63 // Utility function to confirm proper day value
64 // based on month and year.
65 // Is the year 2000 a leap year?
66 int Date::checkDay( int testDay )
67 {
68     static const int daysPerMonth[ 13 ] =
69         {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
70
71     if ( testDay > 0 && testDay <= daysPerMonth[ month ] )
72         return testDay;
```

Destructor  
çağrıldığında bu  
yazıyı yazacak.

# Kompozisyon (Composition)

```
74     if ( month == 2 &&          // February: Check for leap year
75         testDav == 29 &&
76         ( year % 400 == 0 ||
77         ( year % 4 == 0 && year % 100 != 0 ) ) )
78         return testDay;
79
80     cout << "Dav " << testDav << " invalid. Set to dav 1.\n";
81
82     return 1; // leave object in consistent state if bad value
83 }
84 // Fig. 7.4: emply1.h
85 // Declaration of the Employee class.
86 // Member functions defined in emply1.cpp
87 #ifndef EMPY1_H
88 #define EMPY1_H
89
90 #include "date1.h"
91
92 class Employee {
```

# Kompozisyon (Composition)

```
93 public:
94     Employee( char *, char *, int, int, int, int, int, int );
95     void print() const;
96     ~Employee(); // provided to confirm destruction order
97 private:
98     char firstName[ 25 ];
99     char lastName[ 25 ];
100     const Date birthDate;
101     const Date hireDate;
102 };
103
104 #endif
105 // Fig. 7.4: employ1.cpp
106 // Member function definitions for Employee class.
107 #include <iostream>
108
109 using std::cout;
110 using std::endl;
111
```

Başka sınıftan iki üye tanımlandı!

# Kompozisyon (Composition)

```
112#include <cstring>
113#include "employ1.h"
114#include "date1.h"
115
116Employee::Employee( char *fname, char *lname,
117                   int bmonth, int bdav, int bvear,
118                   int hmonth, int hdav, int hvear )
119    : birthDate( bmonth, bdav, bvear ),
120      hireDate( hmonth, hdav, hvear )
121{
122    // copy fname into firstName and be sure that
123    int length = strlen( fname );
124    length = ( length < 25 ? length : 24 );
125    strncpy( firstName, fname, length );
126    firstName[ length ] = '\0';
127
128    // copy lname into lastName and be sure that
129    length = strlen( lname );
130    length = ( length < 25 ? length : 24 );
131    strncpy( lastName, lname, length );
132    lastName[ length ] = '\0';
133}
```

Const nesnelerdeki  
üyelere ilk değerler  
atanıyor

Bu constructor tanımına  
dikkat edin: aslında  
initializer aracılığıyla üye  
nesnenin constructor'una  
bilgi aktarılıyor.

# Kompozisyon (Composition)

```
134 cout << "Employee object constructor: "  
135     << firstName << ' ' << lastName << endl;  
136}  
137  
138void Employee::print() const  
139{  
140     cout << lastName << ", " << firstName << "\nHired: "  
141     hireDate.print();  
142     cout << "    Birth date: "  
143     birthDate.print();  
144     cout << endl;  
145}
```

Constructor  
çağrıldığında bu yazıyı  
yazacak

Burada print de, date nesnesi de const  
olduğu için print fonksiyonu date nesnesini  
kullanarak ekrana yazı yazabiliyor.  
Print fonksiyonunun hiç bir parametresi  
olmadığına dikkat edin: Çünkü print  
fonksiyonu onu çağıran nesneye bağlıdır.

# Kompozisyon (Composition)

```
146
147// Destructor: provided to confirm destruction order
148Employee::~Employee()
149{
150    cout << "Employee object destructor: "
151        << lastName << ", " << firstName << endl;
152}
153// Fig. 7.4: fig07_04.cpp
154// Demonstrating composition: an object with member objects.
155#include <iostream>
156
```



# Kompozisyon (Composition)

```
157using std::cout;
158using std::endl;
159
160#include "employ1.h"
161
162int main()
163{
164    Employee e( "Bob", "Jones", 7, 24, 1949, 3, 12, 1988 );
165
166    cout << '\n';
167    e.print();
168
169    cout << "\nTest Date constructor with invalid values:\n";
170    Date d( 14, 35, 1994 ); // invalid Date values
```

Sadece **employ.h** dosyası yüklenmelidir. Bu dosya **date.h** dosyasını kendi yükler.

# Kompozisyon (Composition)

```
171  cout << endl;  
172  return 0;  
173}
```

```
Date object constructor for date 7/24/1949  
Date object constructor for date 3/12/1988  
Employee object constructor: Bob Jones
```

```
Jones, Bob  
Hired: 3/12/1988  Birth date: 7/24/1949
```

```
Test Date constructor with invalid values:  
Month 14 invalid. Set to month 1.  
Day 35 invalid. Set to day 1.  
Date object constructor for date 1/1/1994
```

```
Date object destructor for date 1/1/1994  
Employee object destructor: Jones, Bob  
Date object destructor for date 3/12/1988  
Date object destructor for date 7/24/1949
```

Dikkat: Hangi nesne ilk olarak  
oluşturuluyor ve yok ediliyor!

# Kalıtım (inheritance) - Giriş

---

- › Temel sınıflardan yeni sınıflar yaratılmasıdır.
- › Yeni (türetilmiş) sınıf, temel (kök) sınıfın özellik ve davranışlarını gösterir.
- › Türetilmiş (Derived) sınıf, kök (base) sınıfın üye fonksiyon ve değişkenlerini miras alır.
- › Bir sınıf, bir veya birden çok kök sınıftan türetilebilir.
- › Miras Çeşitleri:
  - › **public**: Türetilmiş nesnelere, kök sınıftan nesnelerce erişilebilir.
  - › **private**: Türetilmiş nesnelere, kök sınıftan nesneler erişemez.

# Kalıtım

- › Sıklıkla, bir nesne hem bir nesneye göre Türetilmiş sınıf (subclass), hem de bir başka nesne için aynı zamanda kök sınıfıdır (superclass).
- › Örneğin; bir dörtgen, çokgenler sınıfından türetilmişken, aynı zamanda dikdörtgen sınıfının köküdür.
- › Miras alma örnekleri:

Kök (Base) sınıf	Türetilmiş (Derived) sınıf
Öğrenci	ÜniversiteÖğrencisi İlkokulÖğrencisi
Şekil	Çember Üçgen
Kredi	ArabaKredisi EvKredisi
Çalışan	FakülteÇalışanı MemurÇalışanlar
Hesap	ÇekHesabı MevduatHesabı

# Kalıtım

```
1 // Fig. 9.7: point2.h
2 // Definition of class Point
3 #ifndef POINT2 H
4 #define POINT2 H
5
6 class Point {
7 public:
8     Point( int = 0, int = 0 ); // default constructor
9     ~Point(); // destructor
10 protected: // accessible by derived classes
11     int x, y; // x and y coordinates of Point
12 };
13
14 #endif
15 // Fig. 9.7: point2.cpp
16 // Member function definitions for class Point
17 #include <iostream>
18
19 using std::cout;
20 using std::endl;
21
```


# Kalıtım

```
22 #include "point2.h"
23
24 // Constructor for class Point
25 Point::Point( int a, int b )
26 {
27     x = a;
28     y = b;
29
30     cout << "Point constructor: "
31         << '[' << x << ", " << y << ']' << endl;
32 }
33
34 // Destructor for class Point
35 Point::~~Point()
36 {
37     cout << "Point destructor: "
38         << '[' << x << ", " << y << ']' << endl;
39 }
```

# Kalıtım

```
40 // Fig. 9.7: circle2.h
41 // Definition of class Circle
42 #ifndef CIRCLE2_H
43 #define CIRCLE2_H
44
45 #include "point2.h"
46
47 class Circle : public Point {
48 public:
49     // default constructor
50     Circle( double r = 0.0, int x = 0, int y = 0 );
51
52     ~Circle();
53 private:
54     double radius;
55 };
56
57 #endif
```

Circle  
Point'den  
türetiliyor.



# Kalıtım

```
58 // Fig. 9.7: circle2.cpp
59 // Member function definitions for class Circle
60 #include <iostream>
61
62 using std::cout;
63 using std::endl;
64
65 #include "circle2.h"
66
67 // Constructor for Circle calls constructor for Point
68 Circle::Circle( double r, int a, int b )
69     : Point( a, b ) // call base-class constructor
70 {
71     radius = r; // should validate
72     cout << "Circle constructor: radius is "
73         << radius << " [" << x << ", " << y << "]"
74 }
```

İlk değer verme yapısı kullanılıyor. ilk olarak Circle constructor Point constructor' u çağırır.



# Kalıtım

```
75
76 // Destructor for class Circle
77 Circle::~~Circle()
78 {
79     cout << "Circle destructor: radius is "
80         << radius << " [" << x << ", " << y << "]"
81 }
82 // Fig. 9.7: fig09_07.cpp
83 // Demonstrate when base-class and derived-class
84 // constructors and destructors are called.
85 #include <iostream>
86
87 using std::cout;
88 using std::endl;
89
90 #include "point2.h"
91 #include "circle2.h"
92
```

Son olarak Circle destructor' u Point' in destructor' unu çağırır.

# Kalıtım

```
93 int main()
94 {
95     // Show constructor and destructor call
96     {
97         Point p( 11, 22 );
98     }
99
100     cout << endl;
101     Circle circle1( 4.5, 72, 29 );
102     cout << endl;
103     Circle circle2( 10, 5, 5 );
104     cout << endl;
105     return 0;
106 }
```

Point constructor: [11, 22]

Point destructor: [11, 22]

Hatırlatma: **Point**  
constructor **circle** nesnesi  
içinde **circle**  
constructordan önce çağrılır.

Point constructor: [72, 29]

Circle constructor: radius is 4.5  
[72, 29]

**Point destructor Circle**  
destructor 'dan sonra çağrılır.  
(outside in).

Point constructor: [5, 5]

Circle constructor: radius is 10 [5, 5]

Circle destructor: radius is 10 [5, 5]

Point destructor: [5, 5]

Circle destructor: radius is 4.5 [72, 29]

Point destructor: [72, 29]

# Kalıtım

```
Point constructor: [11, 22]
Point destructor:  [11, 22]

Point constructor: [72, 29]
Circle constructor: radius is 4.5 [72, 29]

Point constructor: [5, 5]
Circle constructor: radius is 10 [5, 5]

Circle destructor: radius is 10 [5, 5]
Point destructor:  [5, 5]
Circle destructor: radius is 4.5 [72, 29]
Point destructor:  [72, 29]
```

# Hatırlatma

- › Tüm Kodları kendi bilgisayarınızda uygulayın

