BM102
Algoritma ve Programlama II

Miras

İçerik

1. Miras Kavramı
2. Miras Yönteminin Uygulanması
3. Erişim Etiketleri: `public`, `private`, `protected`
4. Miras ve Yapıcı Fonksiyonlar
5. Üst Sınıf Tanımlamasında `public`, `private` ve `protected` Erişim Etiketlerinin Kullanımı
6. Üst Sınıfdaki Fonksiyonların Alt Sınıflarda Tekrar Tanımlanması
7. Çoklu Miras

Çözümlü Sorular

Hedefler

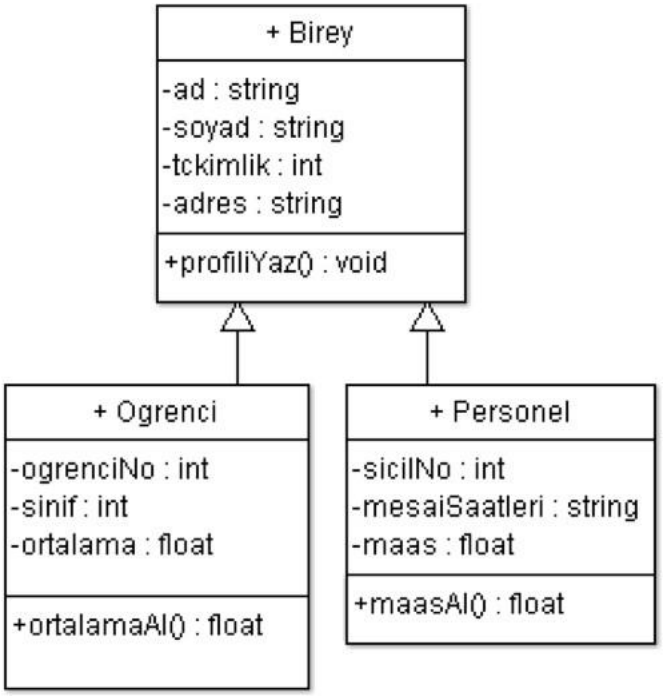
- Verilen sınıflar arasında miras ilişkisi kurma
- Aralarında miras ilişkisi olan sınıf tanımlarını dikkate alarak UML sınıf diyagramını çizme
- `public`, `private` ve `protected` erişim etiketlerinin farklarını açıklama
- Verilen problem tanımını dikkate alarak, üst sınıf tanımlamasında, `public`, `private` ve `protected` etiketlerinden uygun olanı seçme
- Alt sınıflarda üst sınıfa ait yapıcı fonksiyonları çağırma
- Üst sınıfta tanımlı olan fonksiyonları alt sınıflarda tekrar tanımlama
- Sınıflar arasında çoklu miras ilişkisi kurma
- Yeniden kullanılabilirlik prensibini uygulama

1. Miras Kavramı

- *İng. Inheritance*
- Var olan bir sınıftan bir alt sınıf türeterek üyelerin alt sınıfa aktarılmasına *miras* adı verilir.
- Sınıflar arası ilişkiyi tanımlayarak bir hiyerarşi kurulmasını sağlar.
- İki sınıf arasındaki veri üyelerine ve üye fonksiyonlarına bakıldığında benzerlikler var ise, ortak olan bu üyeler temel bir sınıfa toplanır. Bu temel sınıftan türeyen alt sınıflar bu üyelere miras yoluyla sahip olurlar.
- Ortak üyelere ek olarak, alt sınıf kendine özgü yeni veri üyeleri ve üye fonksiyonlar tanımlayabilir.

1. Miras Kavramı...

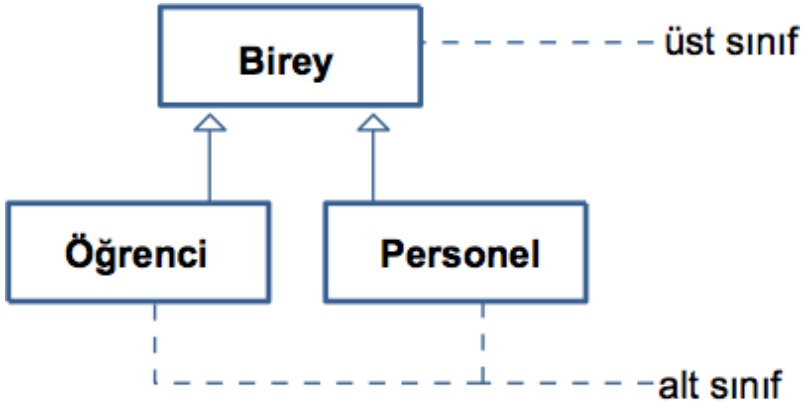
- Ogrenci ve Personel sınıfları; ad, soyad, tckimlik ve adres bilgilerini Birey sınıfından miras olarak alır.



Şekil 13.3 Birey, Ogrenci ve Personel Sınıf Diyagramları

1. Miras Kavramı...

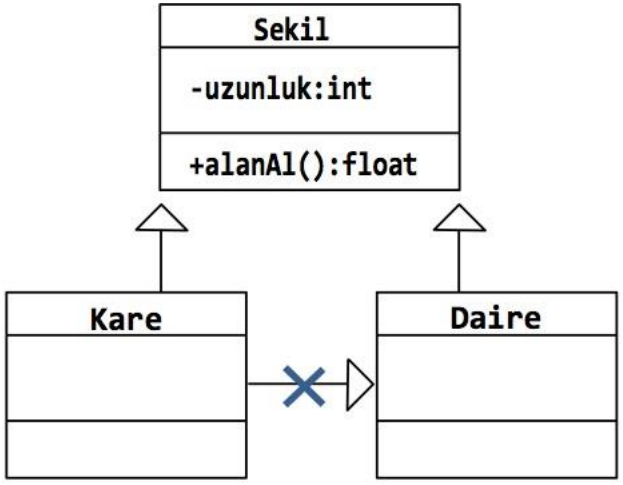
- Özellikleri miras alınan sınıfa ***üst sınıf*** (superclass) veya ***taban sınıf*** (base class), miras alan sınıfa ise ***alt sınıf*** (subclass) veya ***türemiş sınıf*** (derived class) adı verilir



Şekil 13.4 Üst ve Alt Sınıf Kavramları

1. Miras Kavramı...

- Bir üst sınıf sadece kendi bünyesinde tanımlı üyelere erişebilir, alt sınıflarına ait üyelere erişemez.
  - Öğrenci bir Birey'dir. *// doğru bir önermedir*
  - Kare bir Daire'dir. *// doğru bir önerme değildir*



Şekil 13.5 Sekil, Kare ve Daire Sınıfları Arasındaki İlişki

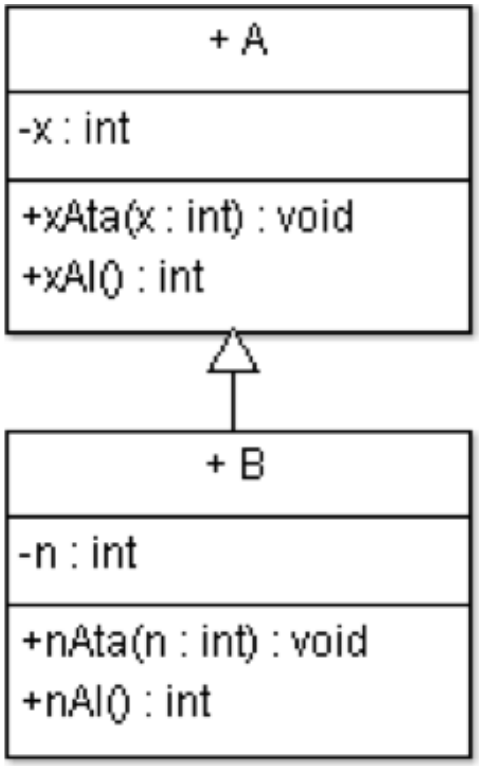
2. Miras Yönteminin Uygulanması

```
class <alt sınıf>  : <erişim etiketi> <üst sınıf>
{...};
```

- Miras ilişkisi, **public** (genel), **private** (özel) ve **protected** (korumalı) olmak üzere üç farklı erişim etiketi ile kurulabilir.
- `public` erişim etiketi, üst sınıftan aktarılan `public` üyelere alt sınıftan da `public`, `protected` üyelere `protected` ve `private` üyelere ise `private` olarak erişileceğini ifade eder.

2. Miras Yönteminin Uygulanması…

- Örnek

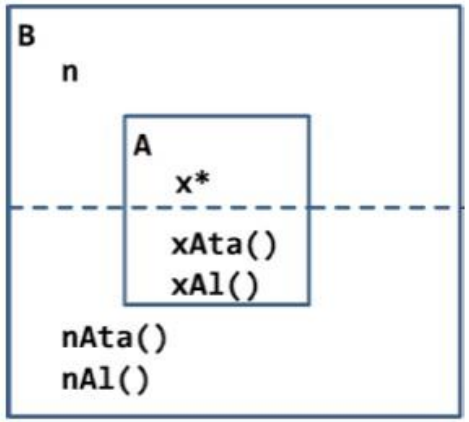


Şekil 13.6 A Üst Sınıf ve B Alt Sınıfı

2. Miras Yönteminin Uygulanması…

Örnek 13.1

```
//---------- A.h ----------
class A                          // Üst sınıf tanımı
{
    int x;
    public:
        void xAta(int _x){x=_x;}
        int xAl(){ return x;}
};
```



Şekil 13.7 B Tipinde nsn Nesnesi

```
//---------- B.h ----------
#include "A.h"
class B: public A          // Alt sınıf tanımı: B sınıfı A sınıfından türemiştir
{
    int n;
    public:
        void nAta(int _n){n = _n;}
        int nAl(){return n;}
};
//---------- Uygulama.h ----------
#include <iostream>
#include "B.h"
using namespace std;
int main()
{
    B nsn;
    nsn.xAta(3);
    nsn.nAta(4);
    cout<<"x:"<<nsn.xAl()<<endl;
    cout<<"y:"<<nsn.nAl()<<endl;
    return 0;
}
```

Çıktı

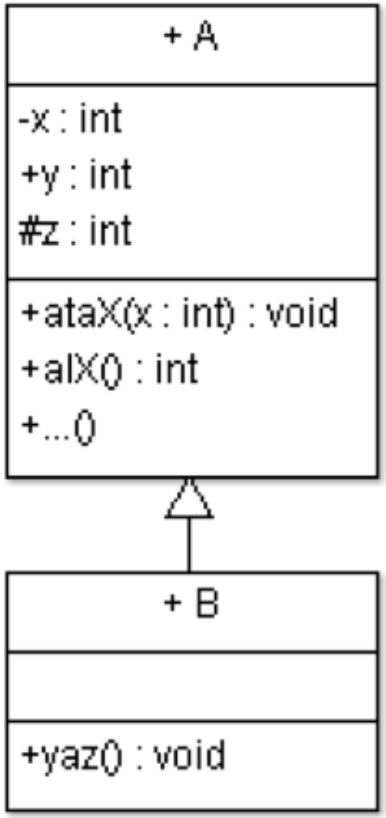
```
x:3
y:4
```

3. Erişim Etiketleri: `public`, `private`, `protected`

- `public` erişim etiketi ile tanımlanmış tüm üyeler, aralarında miras ilişkisi olsun olmasın diğer tüm sınıflar tarafından erişilebilir üyelerdir.
- `private` erişim etiketine sahip tüm sınıf üyelerine dışarıdan erişim kapalıdır.
- `protected` etiketi ile tanımlanan üyelere sadece alt sınıflar erişebilir.

3. Erişim Etiketleri: public, private, protected…

- Örnek



Şekil 13.8 Erişim Etiketleri

3. Erişim Etiketleri: public, private, protected…

Örnek 13.2

```
//---------- A.h ----------
class A                          // Üst sınıf tanımı
{
    int x;
    public:
        int y;
    protected:
        int z;
    public:
        void ataX(int _x){x=_x;}
        int alX(){ return x;}
        void ataY(int _y){y=_y;}
        int alY(){ return y;}
        void ataZ(int _z){z=_z;}
        int alZ(){ return z;}
};
//---------- B.h ----------
#include <iostream>
#include "A.h"
using namespace std;
class B: public A                // Alt sınıf tanımı
{
    public:
        void yaz();
};
```

3. Erişim Etiketleri: public, private, protected…

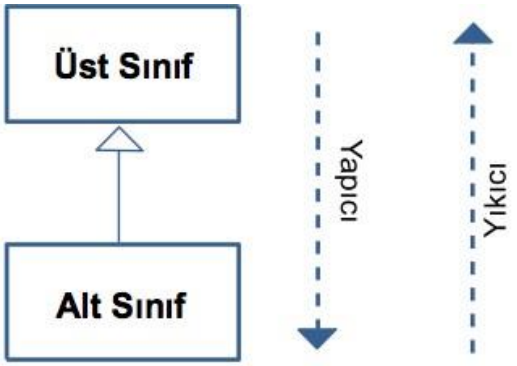
```
//---------- B.cpp ----------
#include <iostream>
#include "B.h"
void b::yaz()
{
    cout<<"x:"<<alX()<<endl;
    cout<<"y:"<<y<<endl;
    cout<<"z:"<<z<<endl;
}
//---------- Uygulama.h ----------
#include "B.h"
int main()
{
    B b;
    b.ataX(3);
    b.y = 10;
    b.ataZ(22);
    b.yaz();
    return 0;
}
```

Çıktı

```
x:3
y:10
z:22
```

4. Miras ve Yapıcı Fonksiyonlar

- Alt sınıf tipinde bir nesne oluşturulurken öncelikle üst sınıfın yapıcı fonksiyonu çağrılır, daha sonra alt sınıfın yapıcı fonksiyonu çağrılır.
- Yıkıcı fonksiyonlar için ise bu tam tersidir: öncelikle alt sınıfın yıkıcı fonksiyonu, onun ardından üst sınıfın yıkıcı fonksiyonu çağrılır.



Şekil 13.9 Yapıcı ve Yıkıcı Fonksiyonların Çağırılma Sırası

4. Miras ve Yapıcı Fonksiyonlar...

Örnek 13.3

```
//----------------A.h  ----------------
class A                              // Üst sınıf tanımı
{
    public:
          int x;
          A(int _x):x(_x){}
          ~A(){cout<<"A sinifi yikici fonksiyonu"<<endl;
               system("pause");}
};
//---------------- B.h  ----------------
#include <iostream>
#include "A.h"
using namespace std;
class B: public A                    // Alt sınıf tanımı
{
    public:
          int y;
          B(int _x, int _y):A(_x){y = _y;}
          ~B(){cout<<"B sinifi yikici fonksiyonu"<<endl;
                 system("pause");}
          void yaz();
};
```

4. Miras ve Yapıcı Fonksiyonlar…

```
//---------------- B.cpp ----------------
#include "B.h"
void B::yaz()
{
    cout<<x<<" "<<y<<endl;
}
//---------------- Uygulama.cpp ----------------
#include "B.h"
int main(){
    B nsn(3,4);
    nsn.yaz();
    return 0;
}
```

Çıktı

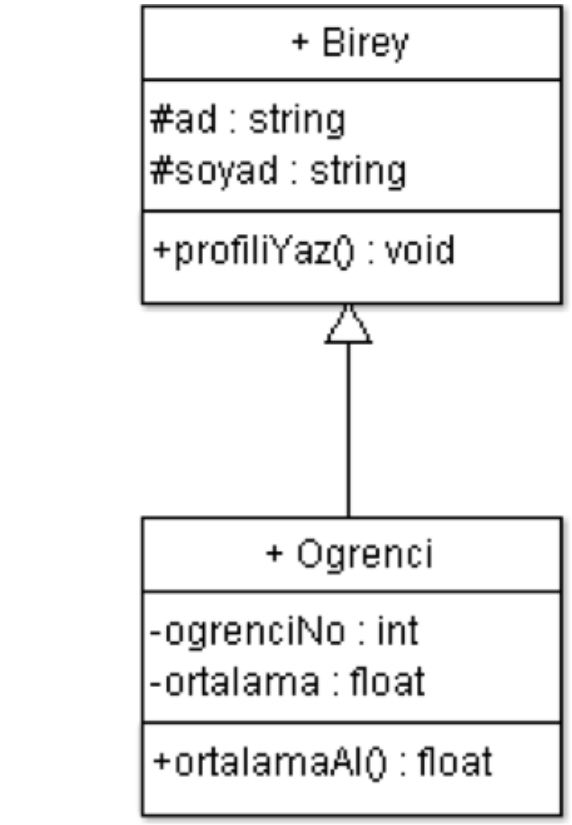
```
3 4

B sinifi yikici fonksiyonu

A sinifi yikici fonksiyonu
```

4. Miras ve Yapıcı Fonksiyonlar...

- **Örnek**
  - `Birey` sınıfı üst sınıftır
  - `Ogrenci` sınıfı `Birey` sınıfının alt sınıfıdır
  - `Ogrenci` sınıfının sahip olduğu veri üyeleri, miras yoluyla aldığı `ad`, `soyad` ve kendi sahip olduğu `ogrenciNo` ve `ortalama`'dır.
  - `Ogrenci` sınıfının sahip olduğu üye fonksiyonlar, miras yoluyla aldığı `profiliYaz()` ve kendi içerisinde tanımladığı `ortalamaAl()` fonksiyonlarıdır.



Şekil 13.10 `Ogrenci` Sınıf Diyagramı

4. Miras ve Yapıcı Fonksiyonlar...

Örnek 13.4

```
//---------------- Birey.h ----------------
#ifndef BIREY_H
#define BIREY_H
class Birey          // Üst sınıf tanımı
{
    protected:
         string ad;
         string soyad;
    public:
         Birey(string _ad, string _soyad):ad(_ad),soyad(_soyad){}
         void profiliYaz();
};
#endif
//---------------- Birey.cpp ----------------
#include "Birey.h"
#include <iostream>
using namespace std;
void Birey::profiliYaz()
{
    cout << "Ad:" << ad << endl;
    cout << "Soyad:" << soyad << endl;
}
```

4. Miras ve Yapıcı Fonksiyonlar…

```
//---------------- Ogrenci.h ----------------
#include "Birey.h"
class Ogrenci : public Birey            // Birey sınıfından türemiş alt sınıf tanımı
{
    int ogrenciNo;
    float ortalama;
   public:
        Ogrenci(string, string, int, float);
        float ortalamaAl();
};
//---------------- Ogrenci.cpp ----------
#include "Ogrenci.h"
Ogrenci::Ogrenci(string _ad, string _soyad,
          int _ogrenciNo, float _ortalama)
          :Birey (_ad, _soyad),
           ogrenciNo(_ogrenciNo),
           ortalama(_ortalama){}
```

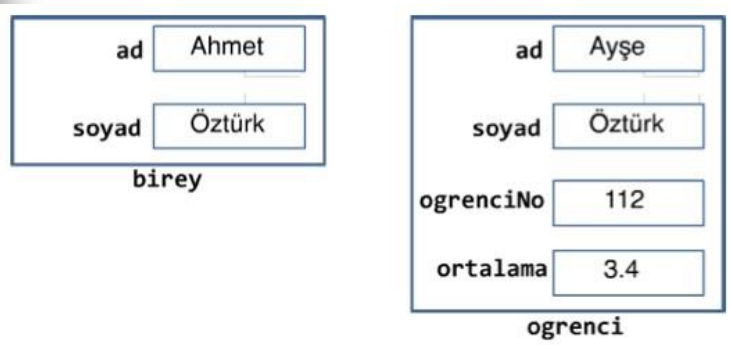
Üst sınıf olan `Birey` sınıfına ait yapıcı fonksiyona parametre geçilmektedir

4. Miras ve Yapıcı Fonksiyonlar...

```
float Ogrenci::ortalamaAl()
{
    return ortalama;
}
//---------- Uygulama.cpp ----------
#include "Birey.h"
#include "Ogrenci.h"
int main()
{
    Birey birey("Ahmet","Ozturk");
    Ogrenci ogrenci("Ayse","Ozturk", 112, 3.4);
    birey.profiliYaz();
    ogrenci.profiliYaz();
    cout<<"Ortalama="<<ogrenci.ortalamaAl()<<endl;
    return 0;
}
```

Çıktı

```
Ad: Ahmet
Soyad: Ozturk
Ad: Ayse
Soyad: Ozturk
Ortalama=3.4
```



Şekil 13.11 birey ve ogrenci Nesneleri

5. Üst Sınıf Tanımlamasında `public`, `private` ve `protected` Erişim Etiketlerinin Kullanımı

- İki sınıf arasında miras ilişkisinin kurulması için **:** işaretinin hemen ardından üst sınıfın adı yazılır
- Üst sınıfın adı yazılırken, söz konusu sınıfın hemen önüne bir erişim etiketi konur.
  - `public`, `private` veya `protected`

5. Üst Sınıf Tanımlamasında `public`, `private` ve `protected` Erişim Etiketlerinin Kullanımı...

- Erişim etiketleri, alt sınıfın üst sınıfın üyelerine erişip erişemeyeceğini ya da nasıl erişeceğini farklı şekillerde tanımlar.

```
class A
{   private:
        int x;
    protected:
        int y;
    public:
        int z;
};
```

5. Üst Sınıf Tanımlamasında public, private ve protected Erişim Etiketlerinin Kullanımı...

private erişim etiketi

- Üst sınıftaki private üyelere alt sınıftan erişilemez.
- Üst sınıftaki protected üyeleri alt sınıfın private üyeleri olarak kabul edilir.
- Üst sınıftaki public üyeleri, alt sınıfın private üyeleri olarak kabul edilir.

```
class B: private A {};
```

A Sınıfı Üyeleri		*B Sınıfına Aktarımları*
`private int x`	→	x'e erişilemez
`protected int y`	→	`private int y`
`public int z`	→	`private int z`

5. Üst Sınıf Tanımlamasında public, private ve protected Erişim Etiketlerinin Kullanımı...

protected erişim etiketi

- Üst sınıftaki `private` üyelere alt sınıftan erişilemez.
- Üst sınıftaki `protected` üyeleri alt sınıfın `protected` üyeleri olarak kabul edilir.
- Üst sınıftaki `public` üyeleri, alt sınıfın `protected` üyeleri olarak kabul edilir.

```
class B: protected A {};
```

A Sınıfı Üyeleri		***B Sınıfına Aktarımları***
`private int x`	→	x'e erişilemez
`protected int y`	→	`protected int y`
`public int z`	→	`protected int z`

5. Üst Sınıf Tanımlamasında public, private ve protected Erişim Etiketlerinin Kullanımı...

public erişim etiketi

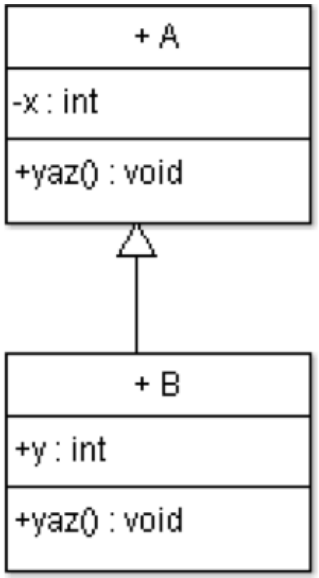
- Üst sınıftaki `private` üyelere alt sınıftan erişilemez.
- Üst sınıftaki `protected` üyeleri alt sınıfın protected üyeleri olarak kabul edilir.
- Üst sınıftaki `public` üyeleri, alt sınıfın `public` üyeleri olarak kabul edilir.

```
class B: public A {};
```

A Sınıfı Üyeleri		*B Sınıfına Aktarımları*
`private int x`	→	x'e erişilemez
`protected int y`	→	`protected int y`
`public int z`	→	`public int z`

6. Üst SınıSaki Fonksiyonların Alt Sınıflarda Tekrar Tanımlanması

- *İng. Redefining*
- Üst sınıfa tanımlanan fonksiyon, alt sınıfta aynı isimle yeniden tanımlanmasıdır.
- Örnek:



Şekil 13.12 Alt sınıflarda yeniden tanımlama

6. Üst SınıSaki Fonksiyonların Alt Sınıflarda Tekrar Tanımlanması...

Örnek 13.5

```
//---------------- A.h ----------------
#ifndef A_H
#define A_H
class A                          // Üst sınıf tanımı
{
    private:
        int x;
    public:
        A(){x=1;}
        void yaz();              // A sınıfına ait yaz() fonksiyonunun prototipi
};
#endif
//---------------- A.cpp ----------------
#include "A.h"
#include <iostream>
using namespace std;
void A::yaz()                    // A sınıfına ait yaz() fonksiyonu
{
    cout<<"x="<<x<<endl;
}
```

6. Üst SınıSaki Fonksiyonların Alt Sınıflarda Tekrar Tanımlanması...

```
//---------------- B.h ----------------
#include "A.h"
class B:public A              // A sınıfından türemiş B alt sınıfı
{
    public:
         int y;
         B(){y=5;}
         void yaz();          // B sınıfına ait yaz() fonksiyonunun prototipi
};
//---------------- B.cpp ----------------
#include "B.h"
void B::yaz()                 // B  sınıfına ait yaz() fonksiyonu
{
    cout<<"y="<<y<<endl;
    A::yaz();                 // A sınıfına ait yaz() fonksiyonu çağırılır
}
```

6. Üst SınıSaki Fonksiyonların Alt Sınıflarda Tekrar Tanımlanması...

```
//---------------- Uygulama.cpp ----------------
#include "A.h"
#include "B.h"
int main()
{
    A a1;
    B b1;
    a1.yaz();          // A sınıfına ait yaz() fonksiyonu çağırılır
    b1.yaz();          // B sınıfına ait yaz() fonksiyonu çağırılır
    b1.A::yaz();       // A sınıfına ait yaz() fonksiyonu çağırılır
    return 0;
}
```

Çıktı

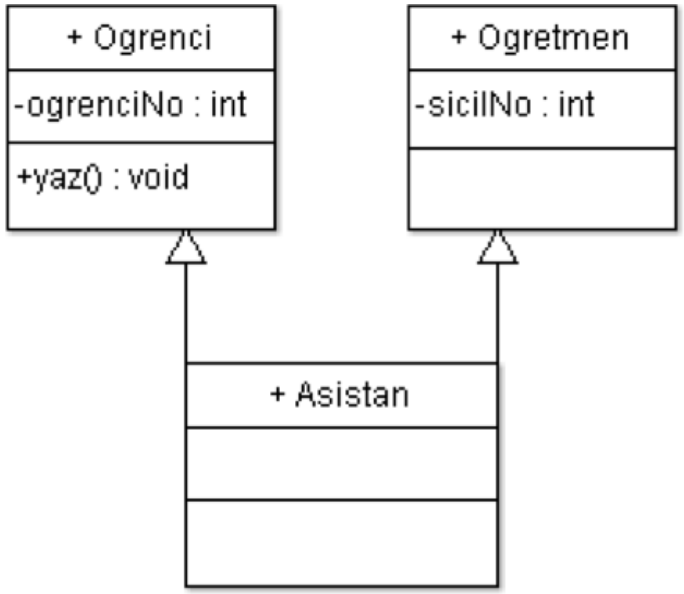
```
x=1
y=5
x=1
x=1
```

7. Çoklu Miras

- *İng. Multiple Inheritance*
- Bir sınıfın birden fazla üst sınıfa sahip olması durumudur
- Çoklu miras ile bir sınıf, birden çok sınıfa ait özellikleri aynı anda miras olarak alabilir.
- Tekli miras kadar yaygın bir kullanımı olmasa da birden çok sınıfla ortak özelliklerin olduğu durumlarda tercih edilen bir yöntemdir.

7. Çoklu Miras...

- Örnek: Bir akademik kurumda henüz doktora öğrencisi olan ve bir yandan da öğretmenlik yapan bir asistanı ele alalım. Bu asistan doktora öğrencisi olduğu için Ogrenci'dir ve aynı zamanda öğretmenlik yapiğı için birOgretmen'dir.



Şekil 13.13 `Ogrenci`, `Ogretmen` ve `Asistan` Sınıflarına Ait UML Sınıf Diyagramı

7. Çoklu Miras...

Örnek 13.6

```
//----------------Ogrenci.h ----------------
class Ogrenci                    // Üst sınıf tanımı
{
    int ogrenciNo;
    public:
        Ogrenci(int _ogrenciNo):ogrenciNo(_ogrenciNo){}
         int ogrenciNoAl(){return ogrenciNo;}
};
//----------------Ogretmen.h ----------------
class Ogretmen                   // Üst sınıf tanımı
{
    int sicilNo;
    public:
        Ogretmen(int _sicilNo):sicilNo(_sicilNo){}
         int sicilNoAl(){return sicilNo;}
};
```

7. Çoklu Miras...

```
//----------------Asistan.h ----------------
#include "Ogrenci.h"
#include "Ogretmen.h"
class Asistan: public Ogrenci, public Ogretmen // İki sınıftan türemiş Asistan alt sınıfı
{
    public:
        Asistan(int ogrenciNo, int sicilNo)
                    :Ogrenci(ogrenciNo),Ogretmen(sicilNo){}
};
//----------------Uygulama.h ----------------
#include <iostream>
#include "Asistan.h"
using namespace std;
int main()
{
    int ogrenciNo = 123;
    int sicilNo = 987;
    Asistan asistan(ogrenciNo, sicilNo);
    cout<<"Ogrenci No:"<<asistan.ogrenciNoAl()<<endl;
    cout<<"Sicil No:"<<asistan.sicilNoAl()<<endl;
    return 0;
}
```

Çıktı

```
Ogrenci No:123
Sicil No:987
```

Çözümlü Sorular…

Soru

2. Verilen X, Y ve Z sınıflarını dikkate alarak aşağıdaki tabloda a,b,c,d ve e veri üyelerinin, söz konusu sınıflardan erişilebilirlik durumlarını ✓ ve × işaretleri ile belirtiniz.

```
//----------X.h ----------
class X
{
    int a;
    protected:
        int b;
    public:
        int c;
};
//----------Y.h ----------
#include "X.h"
class Y: public X
{
    private:
        int d;
};
//----------Z.h ----------
class Z
{
    private:
        int e;
};
```

	a	b	c	d	e
X					
Y					
Z					

Cevap

2.

	a	b	c	d	e
X	✓	✓	✓	×	×
Y	×	✓	✓	✓	×
Z	×	×	✓	×	✓

Çözümlü Sorular…

Soru…

4. Aşağıdaki programın çıktısını yazınız.

```
//----------------A.h  ----------------
#ifndef A_H
#define A_H
class A
{
     protected:
          int x;
     public:
          A(int _x):x(_x){}
          void f1();
          int f2(int y);
          friend void f3(int y);
};
#endif
//----------------A.cpp  ----------------
#include <iostream>
#include "A.h"
using namespace std;
void A::f1()
{
      x *= 2;
      cout<<"1."<<x<<endl;
}
```

```
int A::f2(int y)
{
      x = x+y;
      cout<<"2."<<x<<endl;
      return x;
}
//----------------B.h  ----------------
#include "A.h"
class B:public A
{
     public:
          B(int _x):A(_x){}
          void f1();
          int f2();
          void f3(int y);
};
```

Çözümlü Sorular…

Soru

```
//----------------B.cpp ----------------
#include <iostream>
#include "B.h"
using namespace std;
void B::f1()
{
     x *= 4;
     cout<<"4."<<x<<endl;
}
int B::f2()
{
    x = 3*A::f2(2);
    cout<<"5."<<x<<endl;
    return x;
}
void B::f3(int y)
{
    x = 2*x + 2*y;
    cout<<"6."<<x<<endl;
}
```

```
//----------------Uygulama.cpp ----------------
#include "A.h"
#include "B.h"
void f3(int y)
{
    cout<<"3."<<y<<endl;
}
int main()
{
    A a(5);
    B b(3);
    a.f1();
    a.f2(2);
    f3(3);
    b.f1();
    b.f2();
    b.A::f2(2);
    b.f3(3);
    return 0;
}
```

Çözümlü Sorular...

Cevap

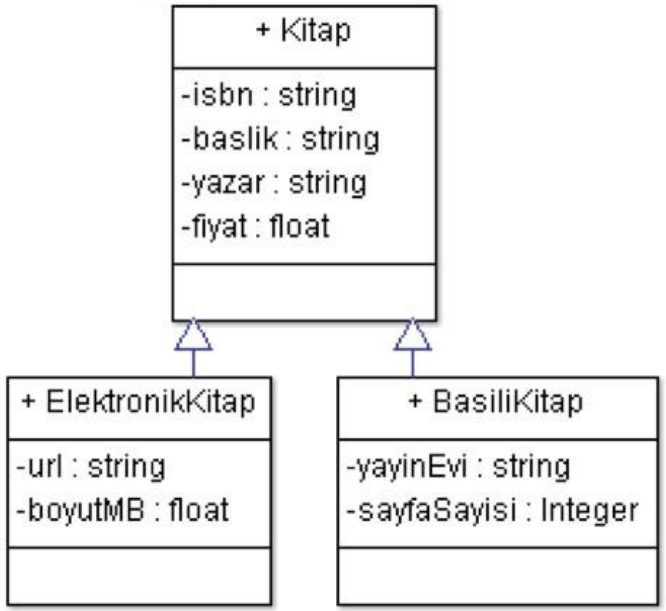
4.

```
1.10
2.12
3.3
4.12
2.14
5.42
2.44
6.94
```

Çözümlü Sorular…

Soru

6. Aşağıda verilen UML sınıf diyagramını ve örnek nesne tanımlarını dikkate alarak gerekli sınıf ve yapıcı/yıkıcı fonksiyon tanımlarını yapınız.



```
ElektronikKitap k2("112323","Cok bi cocuk","Can
        Yucel",6.40,"http://bubir.ornek.com.tr/kitap.pdf",34);
BasiliKitap k3("","b","Elif Safak",34.5,"Palmiye",230);
```

Çözümlü Sorular…

Cevap…

6.

```
// ---------------------- Kitap.h ----------------------
#ifndef KITAP_H
#define KITAP_H
#include <iostream>
using namespace std;
class Kitap
{
    string isbn;
    string baslik;
    string yazar;
    float fiyat;
    public:
        Kitap(string _isbn, string _baslik, string _yazar, float
              _fiyat):
                  isbn(_isbn),baslik(_baslik),
                  yazar(_yazar),fiyat(_fiyat){}
};
#endif
```

Çözümlü Sorular…

Cevap…

```
#endif
// ---------------------- ElektronikKitap.h ----------------------
#include <iostream>
#include "Kitap.h"
using namespace std;
class ElektronikKitap: public Kitap
{
    string url;
    float boyutMB;

    public:
        ElektronikKitap(string _isbn, string _baslik,
              string _yazar, float _fiyat,
              string _url, float _boyut):url(_url),boyutMB(_boyut),
              Kitap(_isbn,_baslik,_yazar,_fiyat){}
};
```

Çözümlü Sorular…

Cevap…

```
// ---------------------- BasiliKitap.h ----------------------
#include <iostream>
#include "Kitap.h"
using namespace std;
class BasiliKitap: public Kitap
{
    string basimEvi;
    int sayfaSayisi;
    public:
        BasiliKitap(string _isbn, string _baslik,
              string _yazar, float _fiyat,
              string _basimEvi,
              int _sayfaSayisi):basimEvi(_basimEvi),
              sayfaSayisi(_sayfaSayisi),
              Kitap(_isbn,_baslik,_yazar,_fiyat){}
};
```

Çözümlü Sorular…

Cevap

```
// ---------------------- Uygulama.h ----------------------
#include <iostream>
#include "Kitap.h"
#include "ElektronikKitap.h"
#include "BasiliKitap.h"
using namespace std;
int main()
{
    Kitap k1("a","b","c",34.5);
    ElektronikKitap k2("a","b","c",34.5,
                       "http://www.atilim.edu.tr/c++.pdf",34);
    BasiliKitap k3("a","b","c",34.5,"Palmiye",230);
    return 0;
}
```