13. KALITIM

Giriş

Kalıtım, sınıf ve nesnelerden sonra nesneye yönelik programlamanın en popüler konusu en güçlü özelliğidir. Kalıtım, bir sınıfın diğer sınıfın özelliklerini ve işlevlerini miras yoluyla alma tekniğidir. Bu şekilde önceden yazılmış ve doğrulanmış olan kodu tekrar tekrar yazmadan kullanma şansına sahip oluruz.

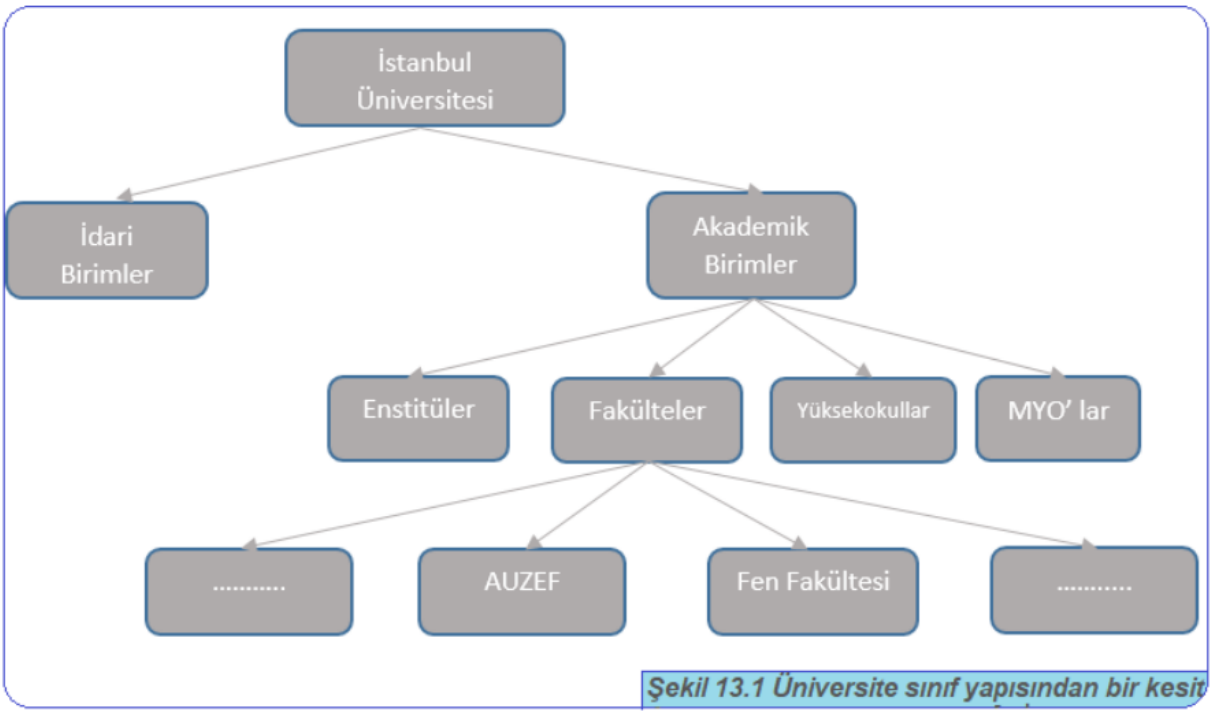
Kalıtımda, miras yoluyla özellikleri ve işlevleri alınan sınıfa ***temel sınıf*** veya ***üst sınıf*** denir. Söz konusu bu özellik ve işlevleri alan sınıfa da **türetilmiş sınıf** denir. Türetilmiş sınıf türetildiği temel sınıfın bütün özelliklerini ve işlevlerini kullanabilme özelliğine sahip olur.

Temel Sınıf: Temel sınıf, Nesne Tabanlı Programlama dilinde, diğer sınıfların türetildiği sınıfın adıdır. Temel sınıfı devralan sınıf, bu temel sınıfın tüm üyelerine sahiptir ve ayrıca bazı ek özelliklere de sahip olabilir. Temel sınıf üyeleri ve üye işlevleri, türetilmiş sınıfın Object öğesine devralınır. Yukarıda da belirtildiği gibi temel sınıf, üst sınıf olarak da adlandırılır.

Türetilmiş Sınıf: Var olan bir sınıftan (Temel Sınıftan, üst sınıftan) oluşturulan sınıftır. Türetilmiş sınıf, bir temel sınıfın tüm üyelerini ve üye işlevlerini devralır. Türetilmiş sınıf, temel sınıfına göre daha fazla işlevselliğe sahip olabilir ve temel sınıfına kolayca erişebilir. Türetilmiş alt sınıf da denir.

Nesneye yönelik programlamanın daha önce kullanılan yapısal programlama yaklaşımına göre, kalıtım tekniği ile programlama sürecine sağladığı önemli avantajlar vardır. Bu avantajlardan bazıları yazılmış olan kodun yeniden kullanılmasına izin verilmesi, bunun sonucu olarak sınıf kütüphanelerinin dağıtımının kolaylaşması, yazılım projelerinin geliştirilmesi ve bakımı sırasında zamandan ve paradan tasarruf edilmesinin sağlaması, bu teknik kullanılarak yazılan programların okunabilirliğinin artması şeklinde sayılabilir.

Aşağıda şekil 13.1'de üniversite sınıf yapısından temsili bir kesit gösterilmiştir.



Şekil 13.1 Üniversite sınıf yapısından bir kesit

Şekil 13.1'de en alt seviyedeki türetilmiş sınıfların fakülteler olduğu görülmektedir. Çok büyük olmasının önüne geçmek için, şekilde bütün fakültelere yer verilmemiştir. Şekle baktığımızda temel sınıfın İstanbul Üniversitesi olduğunu görüyoruz. Bu temel sınıfa ait özelliklerin İstanbul üniversitesinin adresi, adrese ek olarak üniversitenin kuruluş yılı, eski dönemlere ait ve hâlihazırdaki üst yönetim organları, yöneticiler vb. olduğunu varsayabiliriz. Bunun yanında mesela bunların dökümünün alınması ile ilgili bir işlev olması

gerektiğini de söyleyebiliriz. Burada tamamını saymadığımız bütün bu özellik ve işlevler Üniversitenin Rektörlük birimini ilgilendirdiği gibi örneğin fakülteleri de doğrudan ilgilendirmektedir.

Konunu anlaşılması için, temel sınıfın özelliklerinden olan üniversite rektörünün isminin, bütün fakülteleri, öğrencilerin diplomalarının imzalanması açısından ilgilendirdiğini hatırlayalım. Dolayısı ile bütün fakülteler, fakültelerin yanında enstitüler, yüksekokullar MYO 'lar temel sınıfın bu özelliğini kendilerine ait sınıf veya sınıflardan birinde bu özelliği bulundurmadan kullanabileceklerdir. Tabi temel sınıfın özelliklerinin yanında işlevlerinin (üye fonksiyonlar) de aynı şekilde türetilmiş sınıflar tarafından kullanabileceklerini hatırdan çıkarmamalıyız. Bu basit örnek bile kod yazarken, verileri depolarken nesneye yönelik programlama yaklaşımının kalıtım özelliğinin bize ne kadar katkı sağladığını açık olarak anlatmaktadır. Sonuç olarak kalıtım yoluyla elde ettiğimiz temel avantajın, kodun tekrar kullanılabilirliği olduğunu burada söylemeliyiz.

C ++ 'da temel sınıftan türetilmiş sınıf oluşturmak için aşağıdaki sözdizimi kullanılır:

```
class Türetilmiş_Sınıf : Erişim_Belirteci Temel Sınıf
```

İpucu! *Temel sınıftan türetilmiş sınıf oluşturulması sırasında erişim belirteci yazılmamış ise erişim belirtecinin varsayılanı*

Kalıtım yoluyla sınıfların türetilmesi göstermek amacıyla çizilen şekil 13.1 için aşağıdaki açıklamaları okuyunuz:

En üstteki İstanbul Üniversitesi sınıfı tüm sınıfların temel sınıfıdır.

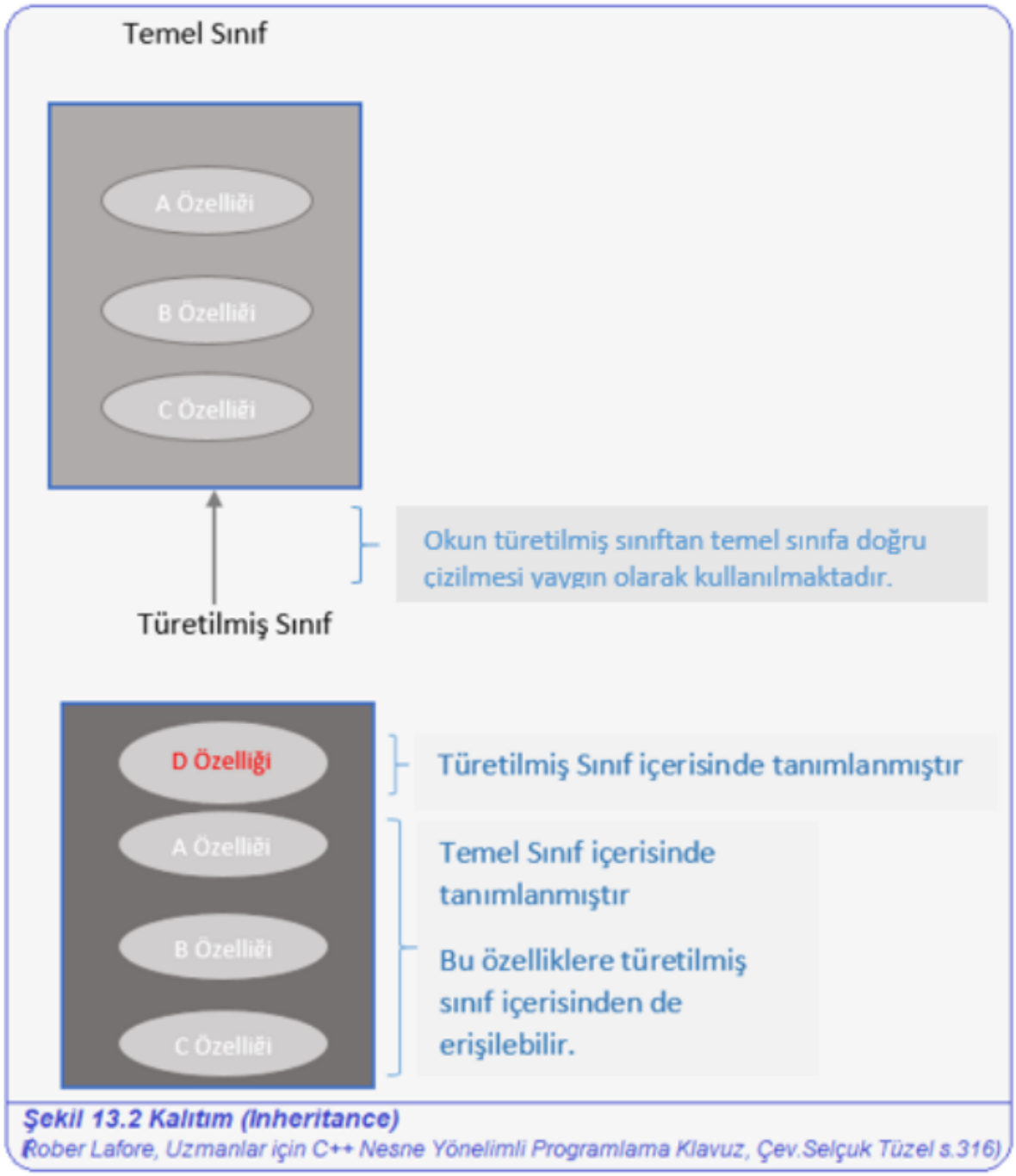
İdari Birimler ve Akademik Birimler İstanbul Üniversitesi sınıfı için türetilmiş sınıftır.

Enstitüler, Fakülteler, Yüksekokullar ve MYO 'lar için Akademik Birimler üst sınıftır.

İstanbul Üniversitesi sınıfından bulunan tüm özellikler, bu sınıftan türetilmiş İdari Birimler ve Akademik Birimler sınıflarının da özellikleridir. Ancak İdari Birimler ve Akademik Birimler sınıflarında bulunan bir özellik İstanbul Üniversitesi sınıfında olmayabilir. Türetilen sınıflar temel sınıfların bütün özelliklerini taşırken kendilerine ait özellikler de barındırabilirler.

AUZEF sınıfı Fakülteler sınıfına göre daha az nesneyi ifade eder. Çünkü AUZEF sınıfında sadece o sınıfa ait nesneler olurken Fakülteler sınıfında her bir fakülteye ait sınıftan nesneler oluşturulur.

Temel sınıf ile Türetilmiş sınıf arasındaki kalıtım ilişkisi şekil 13.2'de gösterilmiştir.



Şekil 13.2 Kalıtım (Inheritance)
(Rober Lafore, Uzmanlar için C++ Nesne Yönelimli Programlama Klavuz, Çev.Selçuk Tüzel s.316)

13.1 Türetilmiş Sınıf Tanımlama

Sınıflar ve Nesneler başlıklı önceki ünitede public ve private erişim belirteçlerinin sınıf tanımlanırken nasıl kullanıldığını ve sınıf içerisinde public ve private olarak tanımlanmış olan üyelere erişimle ilgili kuralları açıklamıştık. Bu kuralları burada da özetleyecek olursak, bir sınıfa ait public elemanlara o sınıfın üye fonksiyonları ile erişilebileceği gibi sınıf dışından, üye fonksiyon kullanmadan da o sınıfın public elemanlarına erişilebilir. Bir sınıfa ait private üye değişken veya fonksiyonlarına ise sadece o sınıftan veya sınıfa ait arkadaş fonksiyonlardan erişilebilir, sınıf dışından asla erişilemez.

Sınıflar ve Nesneler ünitesinde (12. Bölüm) kullanmadığımız fakat sınıfların tanımlanması sırasında kullanılan üçüncü bir erişim belirteci daha vardır. Bu erişim belirteci protected erişim belirtecidir. Bir sınıf içerisinde protected olarak tanımlanmış üyeler, erişilebilirlik açısından sınıfın private üyelerine çok benzer. Protected üyelerin private üyelerden tek farkı türetilmiş sınıflardan protected üyelere erişilebilir.

İpucu! *Bir sınıfa ait üye fonksiyon veya değişkenler tanımlanırken erişim belirteci kullanılmamış ise, sınıfın o üyelerinin private olarak tanımlandığı varsayılır.*

Bir sınıf bir temel sınıftan türetilirken, temel sınıf public, protected veya private miras yoluyla miras alınabilir. Kalıtımın türü, erişim belirteci tarafından belirtilir.

Public Kalıtım – Bir temel sınıftan public erişim belirteci (public niteleyici) kullanarak bir sınıf türetildiğinde, ana sınıfın public üyeleri türetilmiş sınıfın da public üyeleri olur ve temel sınıfın protected üyeleri türetilmiş sınıfın protected üyeleri olur. Bir temel sınıfın private üyelerine hiçbir zaman doğrudan türetilmiş bir sınıftan erişilemez, ancak temel sınıfın public ve private üyelerine çağrılar yoluyla erişilebilir.

Protected Kalıtım – Bir temel sınıftan protected erişim belirteci (protected niteleyici) kullanarak bir sınıf türetildiğinde, temel sınıfın public ve protected üyeleri türetilmiş sınıfın protected üyeleri haline gelir.

Private Kalıtım – Bir temel sınıftan private erişim belirteci (private niteleyici) kullanarak bir sınıf türetildiğinde, temel sınıfın public ve private üyeleri türetilmiş sınıfın private üyeleri haline gelir.

Yukarıda, bir temel sınıftan bir türetilmiş sınıf oluşturulması sırasında kullanılan erişim belirtecine göre temel sınıfın üyelerinin türetilmiş sınıf için nasıl nitelenmesi gerektiği açıklandı. Tablo 13.1'te Bu açıklamalara göre temel sınıf elemanlarına erişim sınırlılıkları özet olarak gösterildi.

Temel Sınıf üyelerinin erişim belirteç tipi	Kalıtım Tipi		
	Public	Protected	Private
Public	public	protected	private
Protected	protected	protected	private
Private	Erişim Yok	Erişim Yok	Erişim Yok

Tablo 13.1 Temel Sınıf üyelerine Kalıtım tipine göre erişim seviyeleri

13.1.1 Public Erişim Belirteci Kullanarak Sınıf Türetme

Temel sınıftan türetilmiş sınıfa miras alınırken erişim belirteci olarak public kullanılır. Public niteleyicisi ile sınıf türetmek için kullanılan söz dizimi aşağıdaki gibidir.

```
class türetilecek_sınıf_ismi : public temel_sınıfın_ismi
```

Türetilen sınıf için, türetildiği ana sınıfın public olan bütün elemanları türetilmiş sınıfın public, protected elemanları protected eleman olur. Türetilmiş sınıfın bütün nesneleri temel sınıfın public elemanlarına ve protected elemanlarına erişebilir. Türetilmiş sınıf nesneleri temel sınıfın private elemanlarına erişemez.

Program 13.1 public erişim belirteci ile temel sınıftan türetilmiş sınıf oluşturmaya örnek olması için yazılmış ve kaynak kodları ile ekran çıktısı aşağıda verilmiştir.

```cpp
#include <iostream>
using namespace std;
class personel
{
    public:
    string prsNo,ad,soyad;
    float maas;
        personel(){};
        void goruntule()
        {
            cout<<prsNo<<" "<<ad<<" "<<soyad<<" "<<maas<<endl;
        }
};
class muhendis:public personel
{
    public:
      muhendis(string n,string a,string s,float m)
        {
            prsNo=n;
            ad=a;
            soyad=s;
            maas=m;
        }
};
class isci:public personel
{
    public:
    isci(string n,string a,string s,float m)
        {
            prsNo=n;
            ad=a;
            soyad=s;
            maas=m;
        }
};
int main() {
    muhendis m1("123","ali","Dereli",10500);
    isci i1("321","veli","Deresiz",7500);
    m1.goruntule();
    i1.goruntule();
    return 0;
}
```

Program 13.1 Public ile Temel sınıftan türetilmiş sınıf oluşturma

```
123 ali Dereli 10500
321 veli Deresiz 7500

--------------------------------
Process exited after 4.137 seconds with return value 0
Press any key to continue . . .
```

Prohram 13.1'in Ekran Çıktısı

Program 13.1'in işlem adımları aşağıda sıralanmıştır.

Program 13.1 'de temel sınıf olarak personel sınıfı oluşturuldu ve her personelin numarası, adı ve soy adı olacağı için PrsNo, ad, soyad üye değişkenleri ve goruntule() üye fonksiyonu oluşturuldu.

Daha sonra personel temel sınıfından muhendis adında bir sınıf, public erişim belirteci kullanılarak türetildi ve yapıcı fonksiyon ile personel sınıfının üye değişkenlerine atama yapıldı.

Aynı şekilde işlemler isci sınıfı için de tekrarlandı.

main() fonksiyonu içerisinde m1 ve i1 nesneleri tanımlandı ve atanan değerler görüntüle fonksiyonu ile yazdırıldı.

13.1.2. Private Erişim Belirteci Kullanarak Sınıf Türetme

Temel sınıftan türetilmiş sınıfa miras alınırken erişim belirteci olarak private kullanılır. C++ 'ta Temel sınıftan türetilmiş sınıfa miras alınırken erişim belirteci yazılmaz ise varsayılan olarak erişim belirtecinin private olduğu kabul edilir. private niteleyicisi ile sınıf türetmek için kullanılan söz dizimi aşağıdaki gibidir.

```
class Türetilecek_Sınıf_İsmi : private Temel_Sınıfın_İsmi
```

Private türetmede ana sınıfın public ve protected üyeleri türetilmiş sınıf için private hale gelir. Burada dikkat edilmesi gereken husus türetme şekli ne olursa olsun ana sınıfın private elemanlarına erişilemez.

Program 13.2 private erişim belirteci ile temel sınıftan türetilmiş sınıf oluşturmaya örnek olması için yazılan programın kaynak kodları ve ekran çıktısı aşağıda verilmiştir.

Program 13.2'nin işlem adımları aşağıda verildiği gibi olacaktır.

Program 13.2 'de temel sınıf olarak personel sınıfı oluşturuldu ve PrsAd, prsSoyad üye değişkenleri public olarak temel sınıfta tanımlandı.

Daha sonra personel temel sınıfından prsAta adında bir sınıf, private erişim belirteci kullanılarak türetildi ve yapıcı fonksiyon ile personel sınıfının üye değişkenlerine atama yapıldı. (Temel sınıf olan personel sınıfına ait public üye değişkenlerin PrsAta sınıfı için private niteliğe dönüştüğünü hatırınızdan çıkarmayınız)

main() fonksiyonu içerisinde türetilmiş sınıfa ait kişi nesnesi tanımlandı PrsAd ve prsSoyad üye değişkenlerine türetilen sınıfın kurucu fonksiyonu ile atama yapıldı.

Türetilen sınıfa ait görüntüle üye fonksiyonu ile personelin ad ve soyadı görüntülendi.

```cpp
#include <string>
#include <iostream>
using namespace std;
class personel
{
      public:
            string prsAd,prsSoyad;
            X(){};
      };
class prsAta:private personel
{
      public:
           prsAta(string a,string s)
            {
                  prsAd=a;
                  prsSoyad=s;
                  }
            void goruntule()
            {
                cout<<prsAd<<"\t"<<prsSoyad<<endl;
             }
      };
int main()
{
    prsAta kisi("Mehmet","Oduncu");
    kisi.goruntule();
    system("PAUSE");
    return 0;
}
```

Program 13.2 private il temel sınıftan türetilmiş sınıf oluşturma

13.1.3. Protected Erişim Belirteci Kullanarak Sınıf Türetme

Temel sınıftan türetilmiş sınıfa miras alınırken erişim belirteci olarak protected kullanılır. protected niteleyicisi ile sınıf türetmek için kullanılan söz dizimi aşağıdaki gibidir.

public ile sınıf türetme işleminde ana sınıfın public elemanlarının türetilmiş sınıf için public, ana sınıfın protected elemanlarının türetilmiş sınıf için protected olduğunu ve private ile sınıf türetmede ana sınıfın public ve protected elemanlarının türetilen sınıf için private olduğunu daha önce gördük.

Protected ile sınıf türetmede ise ana sınıfın public veya protected elemanları türetilmiş sınıfın protected elemanı haline gelir.

Program 13.3 protected erişim belirteci ile temel sınıftan türetilmiş sınıf oluşturmaya örnek olması için yazılan programın kaynak kodları ve ekran çıktısı aşağıda verilmiştir.

13.2 Kalıtım Türleri

Bu ünitede şu ana kadar kalıtım hakkındaki temel bilgileri öğrendik. C++ programlama dilinde sınıfların türetilme şekillerine veya türetilmesi sırasında kaç sınıftan miras aldığına bağlı olarak birbirinden farklı kalıtım türleri vardır. Bu farklı kalıtım türleri aşağıda listelenmiştir:

1. Tekli Kalıtım
2. Çoklu Kalıtım
3. Çok seviyeli Kalıtım
4. Hiyerarşik Kalıtım
5. Hibrit Kalıtım

Bu farklı kalıtım türleri temsili olarak şekil 13.3'te gösterilmiştir.

13.2.1. Tekli Kalıtım

Tekli kalıtımda, bir sınıf sadece bir temel sınıftan türetilir. Bu, bir üst sınıftan türetilen yalnızca bir alt sınıfın olduğu anlamına gelir. Tekli kalıtım için kullanılan söz dizimi aşağıdaki gibidir.

Program 13.4 Tekli Kalıtıma örnek olarak yazılmıştır.

13.2.2. Çoklu Kalıtım

Çoklu kalıtım, bir sınıfın birden fazla ana sınıftan türetildiği bir kalıtım türüdür. Aşağıdaki şekilde gösterildiği gibi, C sınıfı ana sınıfı A sınıfı ve B sınıfı olan bir türetilmiş sınıftır. Çoklu kalıtım için kullanılan söz dizimi aşağıdaki gibidir.

Çoklu kalıtım şeması aşağıda gösterilmiştir.

Program 13.5 Çoklu Kalıtıma örnek olarak yazılmıştır.

```cpp
#include <iostream>
using namespace std;
class A {
public:
  A(){
     cout<<"Kurucu A class"<<endl;
  }
};
class B {
public:
  B(){
     cout<<"Kurucu B class"<<endl;
  }
};
class C: public A, public B {
public:
  C(){
     cout<<"Kurucu C class"<<endl;
  }
};
int main() {
   C obj;
   return 0;
}
```

Program 13.5 Çoklu kalıtım örneği

```
Kurucu A class
Kurucu B class
Kurucu C class

--------------------------------
Process exited after 3.538 seconds with return value 0
Press any key to continue . . .
```

Program 13.5'in ekran çıktısı

13.2.3. Çok Seviyeli Kalıtım

Çok düzeyli kalıtımda, bir sınıf önceden türetilmiş başka bir sınıftan türetilir. Aşağıdaki şemadan, C sınıfının B Sınıfından ve B Sınıfının da A Sınıfından türetildiği görülmektedir.

Çoklu seviyeli kalıtım için kullanılan söz dizimi aşağıdaki gibidir:

13.2.4. Hibrit Kalıtım

Hibrit kalıtım genellikle birden fazla kalıtım türünün birlikte kullanılmasıyla ortaya çıkar. Aşağıdaki gösterimde, melez bir kalıtım elde etmek için çoklu kalıtım (B, C ve D) ve çok seviyeli kalıtım (A, B ve D) yer almaktadır. Bu şekilde bir uygulamada birden fazla kalıtım türü yar alıyorsa bu kalıtım türüne Hibrit kalıtım denmektedir.

Hibrit Kalıtım Şeması:

13.2.5. Hiyerarşik Kalıtım

Hiyerarşik kalıtım, Aşağıdaki şemada gösterildiği gibi, birden fazla sınıfa tek bir temel sınıftan miras alınarak oluşturulur. Bundan dolayı kalıtım alma şeklinde hiyerarşik bir düzen vardır.

Hiyerarşik Kalıtım Şeması:

C ++ dili yukarıda açıklanan tüm bu kalıtım türlerini desteklemektedir. Buradan hareketle, C ++ 'ın kalıtım için çok güçlü bir desteği olduğunu, C ++ kullanarak birçok problemin daha etkin bir şekilde modelleyebileceğini söyleyebiliriz.

Bölüm Özeti

Bu bölümde Kalıtımın, Nesne yönelimli programlamanın temel özelliklerinden biri olduğunu, programcıların nesneye yönelik programlamanın bu özelliğini kullanarak mevcut bir sınıftan (temel sınıf) yeni bir sınıf (türetilmiş sınıf) oluşturabildiklerini öğrendik. Bu yöntem kullanılarak yazılmış olan kodun yeniden kullanılmasına izin verilmesi, bunun sonucu olarak sınıf kütüphanelerinin dağıtımının kolaylaşması, yazılım projelerinin geliştirilmesi ve bakımı sırasında zamandan ve paradan tasarruf edilmesi, bu teknik kullanılarak yazılan programların okunabilirliğinin artması gibi avantajların programlama sürecine kazandırıldığının farkına vardık.

Türetilmiş sınıfların, temel sınıftaki tüm özellikleri devraldığını ve ayrıca kendine özgü ek özelliklere sahip olabileceklerini öğrendik. Temel sınıflar türetilirken farklı erişim belirteçleri (public, private, protected) kullanılarak yazacağımız programlara farklı güvenlik özellikleri kazandırabileceğimizi gördük.

C++ programlama dilinde sınıfların türetilme şekillerine veya türetilmesi sırasında kaç sınıftan miras aldığına bağlı olarak, birbirinden farklı kalıtım türleri vardır. Bu farklı kalıtım türleri tekli kalıtım, çoklu kalıtım, çok seviyeli kalıtım, hiyerarşik kalıtım, Hibrit kalıtım şeklinde sıralanabilir. Bu bölümde farklı kalıtım türlerinin nasıl kullanılacaklarını öğrendik.

C ++ 'ın kalıtım için çok güçlü bir desteği olduğunu, C ++ kullanarak birçok problemin daha etkin bir şekilde modelleyebileceği konusundaki farkındalığımız arttı.