Interface ve Delegate

- **Interface Kullanımı**

Interfaceler, class veya struct gibi türler için oluşturulmuş modellerdir. Sınıflar değişkenleri, özellikleri, olayları, metotları ve indeksleyicileri tanımlar ve içeriklerini oluşturur, arayüzler ise bir sınıfın temelde hangi üyelerden oluşacağını belirleyen bir nevi şablon yapılarıdır.

Genelde büyük ölçekli projelerde önem kazanmaktadırlar. Arayüzleri **interface** anahtar sözcüğü ile tanımlarız. Ara birimleri sınıflar üzerinde uygulamak için ise "**:**" işareti kullanılır ve geleneksel olarak arabirim isimleri "**I**" harfi ile başlar, tabi bu bir zorunluluk değildir. Fakat bu şekilde kullanılması evrensel olarak kabul edilir.

Interface ve Delegate

- **Interface Kullanımı**

Abstract classlarda olduğu gibi metotlar, özellikler, indexerlar, değişkenler hepsinin sadece tanıtımı yani başlığı yazılıyor, gövdesi boş bırakılıyor. Daha sonra bu interface üzerinden implement(gerçekleştirilen) edilen classlarda bu üyeler aynı şekilde tanımlanıyor ve gövdeleri burada dolduruluyor. Peki abstract classlardan buradaki farkı ne oluyor? Normalde sınıflar yalnızca bir class'tan miras alabilir ama interface ile bir class bir veya birden fazla interfaceden kalıtım alabiliyor.

Interface ve Delegate

- **Interface Kullanımı**

Abstract classlarda olduğu gibi metotlar, özellikler, indexerlar, değişkenler hepsinin sadece tanıtımı yani başlığı yazılıyor, gövdesi boş bırakılıyor. Daha sonra bu interface üzerinden implement(gerçekleştirilen) edilen classlarda bu üyeler aynı şekilde tanımlanıyor ve gövdeleri burada dolduruluyor. Peki abstract classlardan buradaki farkı ne oluyor? Normalde sınıflar yalnızca bir class'tan miras alabilir ama interface ile bir class bir veya birden fazla interfaceden kalıtım alabiliyor.

Interface ve Delegate

- **Interface Kullanımı**

Interface içinde property, event (olay), method, indexer, temsilci (delagate) gibi üyeler tanımlanabilir fakat constructor, destructor ve operator overloading gibi işlemler olamaz. Ayrıca bir interface içinde static metotlar veya static değişkenler tanımlanamaz. Interfaceler implement edildiğinde default olarak public olarak tanımlanmaktadırlar.

Interface ve Delegate

- **Interface Kullanımı**
- Bir arayüz'ün tüm üyeleri public kabul edilir. Private, Protected gibi belirtiçler kullanamayız.
- Diğer yandan bir metodu public olarakta tanımlayamayız. Çünkü zaten varsayılan olarak bütün üyeler public tanımlanmış kabul edilir.
- Arayüz elemanlarını static olarak tanımlayamayız.
- Arayüzlerin uygulandığı sınıflar, arayüzde tanımlanan bütün üyeleri kullanmak zorundadır..

Interface ve Delegate

- **Interface Kullanımı**
- arayüzleri kullanmanın en büyük nedenlerinden birisinin sınıflara çoklu kalıtım desteği vermesi

Interface ve Delegate

```
interface IkrediHesap
    {
        void hesapAc(int m, string mAdsoy);
        string isimGetir();

    }
    interface ILimitHesap
    {
        void limitArttir(double yeniLimit);
        void Yazdir();

    }
```

Interface ve Delegate

```
class KrediHesap:IkrediHesap,ILimitHesap
    {
        int mno;
        string musAdsoyad;
        double limit;

        #region IkrediHesap Uyeleri
        public KrediHesap()
        {
            mno = 1;
            musAdsoyad = "default";
            limit = 100;
        }
```

```
public void hesapAc(int m,string mAd)
        {
            mno = m;
            musAdsoyad = mAd;

        }
        public string isimGetir()
        {
            return musAdsoyad;
        }
        #endregion
```

Interface ve Delegate

```
        #region ILimitHesap tanimlari
        public void limitArttir(double yl)
        {
            limit = yl;
        }
        public void Yazdir()
        {
        Console.WriteLine("Mus Adi:" + musAdsoyad + "\n musno:" +
mno + "\n limit:" + limit);
        }
        #endregion
        static void Main(string[] args)
        {
            KrediHesap k1 = new KrediHesap();
            Console.WriteLine("isim:" + k1.isimGetir());
            k1.Yazdir();
        }
    }
```

Interface ve Delegate

- ## Delegate Kullanımı

Delegate en basit anlamıyla metod referansıdır. Yani **delegate** metod referansını tutar.

Delegate tanımlanırken metodun geri dönüş değeri ve parametreleri belirlenir. Geri dönüş değerini void ve parametresiz tanımlamak da mümkündür. Kullanılacak metod, tanımlanmış olan **delegate** geri dönüş ve parametrelerine uygun olmalıdır.

Interface ve Delegate

- **Delegate Kullanımı**

Bir örnek yapalım, markette satın alınan ürünler için fiş çıkaracaksınız. Satın alınan ürünün kaç tane olduğunu biliyorsunuz. Bu ürünler için hem KDV dahil hem de KDV hariç fiyatlarını göstereceksiniz. Bunun için iki ayrı metod tanımlayabilir ve bunları delegate yardımıyla kullanabilirsiniz.

Interface ve Delegate

- **Delegate Kullanımı**

```
//Hesaplama adında bir delegate tanımı
yapıldı.
//int tipinde bir parametre bekliyor
public delegate decimal Hesaplama(int adet);
static void Main(string[] args)
{
  Hesaplama del = Fiyat;
  Console.WriteLine("KDV Hariç 5 Ürün"+
  " Fiyatı: {0}", del(5));
  del = KDVFiyat;  //delegate değiştiriliyor
  Console.WriteLine("KDV Dahil 5 Ürün"+
  " Fiyatı: {0}", del(5));
  Console.ReadLine();
}
```

```
static decimal Fiyat(int sayi)
{
  //Fiyat 10.5 kabul edilerek hesaplandı.
(KDV Hariç)
  return sayi * 10.5m;
}

static decimal KDVFiyat(int sayi)
{
  decimal kdvharicfiyat = sayi * 10.5m;
  //KDV %18 düşünüldü
  return kdvharicfiyat + kdvharicfiyat *
0.18m;
}
```