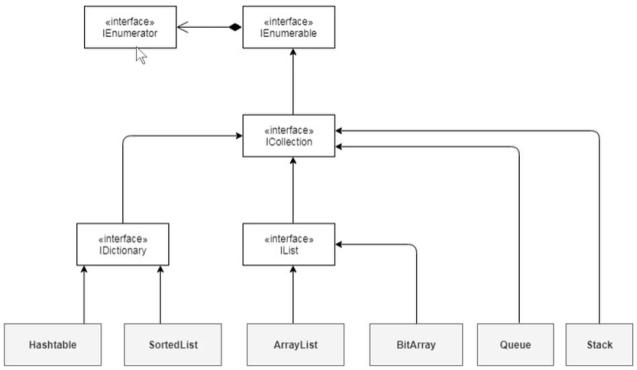
C# Collections

```
9           // Array
10          int[] sayilar = new int[5];
11          sayilar[0] = 10;
12
13          //Collections
14           //Non-generic C.
15              //** ArrayList,SortedList
16           //Generic list <str>
17              //** int,string,product
```

- Collections'ların normal arrayler ile genel farkı arraylar da dizi tanımlaması yaparken boyut belirlemek gerekir ama Collectionslarda böyle bir şart yoktur.



- Genel şema olarak böyle bir yapısı vardır.
- Dictinary (Sözlük) yapısı içerisinde bir key-value durumları vardır. Genelde veri yapılarında bu yapı kullanılır. Örnek bir isim:bilal, okul: hitit üniversitesi gibi düşünülebilir.
- Ilist ise genel olarak listeler olarak geçer ve dizilere kıyasla istenilen eleman eklenip, çıkarılabilir. Belirli bir boyutu şart koşmaz.

ArrayList

```
10          // ArrayList
11
12          ArrayList myList = new ArrayList();
13
14          myList.Add(10);
15          myList.Add("10");
16          myList.Add("abc");
17          myList.Add(10.5);
18
19          IList myList2 = new ArrayList(){"******","10","abc",10.5};
20
21          int sayi = (int)myList[0];
22
23          myList.Insert(1,20);
24          myList.InsertRange(3,myList2);
25
26
27          // Remove Items
28
29          myList.Remove(10);
30          myList.RemoveAt(2);
31          myList.RemoveRange(0,2);
32
33          foreach (var item in myList)
34          {
35              Console.WriteLine(item);
36          }
37
38
39          Console.WriteLine(myList.Contains(10));
40
41          ArrayList sayilar = new ArrayList(){10,5,4,60};
42
43          foreach (var item in sayilar)
44          {
45              Console.WriteLine(item);
46          }
47
48          sayilar.Sort();
49
50          foreach (var item in sayilar)
51          {
52              Console.WriteLine(item);
53          }
```

- ArrayList tanımlaması 12. Satırda yapılmıştır.
- .Add ile eleman eklenmektedir.
- 19.Satırdaki gibi IList de tanımlanır lakin ArrayList özelliği kadar özelliği yoktur. Örnek Insert çalışırken InstertRange çalışmaz.
- 21. Satırdaki gibi eleman bir değişkene atanabilir.
- 23,24. Satırlardaki Intsert ile (index,eleman) istenen index'e eleman eklenebilir. (1,2,3) instert(1,6) => (1,6,2,3). İnsertRange ile liste eklenebilir.
- 29-31. Satırlarda Remove direk 10 elemanını siler, RemoveAt ise 2. İndexteki elemanı siler. RemoveRange ile 0. İndexten başla iki tane eleman sil.
- 33. Foreach ile elemanları yazdırma işlemi yapılır.
- 39. Satır ile .Contains ile listede 10 varmı kontrolü yapar true false değeri dönderir.
- 41. Satırda int listesi tanımlandı, 43 ile ekrana yazdırdık, 48. Satır ile sıraladık daha sonra 50. Satırla tekrar ekrana yazdırdık.

Generic List<T>

```
class Product
{
    public string Name { get; set; }
}
class Program
{
    static void Main(string[] args)
    {
        // Generic List
        List<int> sayilar = new List<int>();
        sayilar.Add(10);
        sayilar.Add(20);
        sayilar.Add(30);

        List<string> isimler = new List<string>();
        isimler.Add("ali");
        isimler.Add("ahmet");
        isimler.Add("yağmur");
        isimler.Add(null);

        List<Product> urunler1 = new List<Product>()
        {
            new Product(){Name="Samsung S6"},
            new Product(){Name="Samsung S7"},
            new Product(){Name="Samsung S8"},
            new Product(){Name="Samsung S9"}
        };

        IList<Product> urunler2 = new List<Product>()
        {
            new Product(){Name="IPhone 6"},
            new Product(){Name="IPhone 7"},
            new Product(){Name="IPhone 8"},
            new Product(){Name="IPhone 10"}
        };
```

```

        urunler1.AddRange(urunler2);

        foreach (var product in urunler1)
        {
            Console.WriteLine(product.Name);
        }

        urunler1.ForEach(p=> {
                Console.WriteLine(p.Name);
        });

        int count = urunler1.Count;

        for (int i = 0; i < urunler2.Count; i++)
        {
            Console.WriteLine(urunler2[i].Name);
        }

        sayilar.Insert(1,100);

        urunler1.InsertRange(1,urunler2);

        for (int i = 0; i < urunler1.Count; i++)
        {
            Console.WriteLine(urunler1[i].Name);
        }

        sayilar.RemoveAt(sayilar.Count-1);

         foreach (var sayi in sayilar)
        {
            Console.WriteLine(sayi);
        }
    }
}
```

- 16,21,27. Satırlardaki gibi generic bir liste tanımlandığında tipi belirtilmelidir.
- Genel olaral List kullanılır, IList de kullanılabilir lakin Arraylistte olduğu gibi IList, List kadar kapsamlı değildir.
- 7. Satırda Product classı oluşturulduğundan 27 ve 35. Satırlarda Product nesnesi oluşturuldu.
- 43. Satırda AddRange ile ürünler 2 listesi ürünler 1'e eklendi.
- 50. Satırda kullanılan ForEach, forech döngüsünün farklı kullanımıdır. Urunler1 listesinin elemanlarını p'ye atar sonrada p'yi yazdırır. (p değişkendir, istediğinizi yazabilirsiniz.)
- 54. Satırda normalde dizi boyutu için lenght fonksiyonu kullanılırdı, generic listelerde count kullanılmaktadır.
- Insert kullanımları arraylist ile aynıdır, belirli indexe eleman ekleme işlemi yapar.

Dictionary<TKey, TValue>

```
static void Main(string[] args)
{
    // Dictionary<TKey, TValue>
    Dictionary<int,string> plakalar = new Dictionary<int, string>();

    plakalar.Add(34,"istanbul");
    plakalar.Add(35,"izmir");
    plakalar.Add(53,"rize");

    Dictionary<int,string> sayilar = new Dictionary<int, string>()
    {
        {1,"Bir"},
        {2,"İki"},
        {3,"Üç"}
    };

    Console.WriteLine(sayilar[1]);

    foreach (KeyValuePair<int,string> plaka in plakalar)
    {
        Console.WriteLine($"{plaka.Key} {plaka.Value}");
    }

    for (int i = 0; i < plakalar.Count; i++)
    {
        Console.WriteLine($"{plakalar.Keys.ElementAt(i)} {plakalar[plakalar.Keys.ElementAt(i)]}");
    }

     Console.WriteLine(plakalar.ContainsKey(34));
     Console.WriteLine(plakalar.Contains(new KeyValuePair<int,string>(34,"istanbul")));

     plakalar.Remove(34);
     plakalar.Remove(53);

     Hashtable ht = new Hashtable();

     ht.Add(1,"ali");
     ht.Add(2,"ahmet");
     ht.Add("3","ali");
}
```

- Listelerde bizler değerler giriyorduk ve bunların da bir index numarası oluyordu. Dictionary(Sözlük) yapısında ise indexleri biz kendimiz ayarlayabiliyoruz. Örnek 34: İstanbul şeklinde.
- Dictionary yapısı 13. Satırda gösterilmiştir. İnt key(index), string Value karşılık gelmektedir.
- 15-17. Satırlarda eleman ekleme işlemleri yapılmıştır.
- 19. Satırda farklı bir kullanımla gösterilmiştir.
- 26. Satırda [] yine key bilgisi girilmesi gerekiyor. Yani plakalar sözlüğü için İstanbul değerini getirmek istiyorsanız 34 yazılmalıdır.
- 28. Satırda Foreach döngüsü kullanımı verilmiştir. KeyValuePair<int,string> yerine var da yazılabilir. 30. Satırda ise plakanın altında sonuçta iki değer var. Ondan plaka.key diye çağırıyoruz.
- 33. Satırda for döngüsü ile kullanımı gösterilmiştir. Plakalar.Keys kodu bize sadece key listesini getirmektedir. .ElementAt ile biz gelen listedeki elemanları tek tek geziyoruz. 35. Satırın sağ tarafında plakalar[] içerisinde Plakalar.Keys.ElementAt yazmasının sebebi key bilgileri gelmekte. Örnek plakalar[34].
- 38,39. Satırlarda ContainsKey ile değer sorgulaması yapılmaktadır. Böyle bir değer varmı yokmu gibisinden true false değeri döner.
- Remove içerisine key bilgisi girilerek silme işlemi yapılır. Örnek 34 değeri girildiğinden İstanbul silincek.
- Hashtable yapısı içerisinde ise nesne yapısı gibidir. Örnek index int ver sonra string ver istenilen şekilde verilebilir. Lakin Hashtable yapısı dictionary yapısına kıyasla çok daha yavaştır.