Bölüm 20

Koleksiyonlar

Koleksiyon Sınıfları

Klâsik programlama dillerinde `array` çok önemli bir veri tipidir. Çok sayıda değişkeni kolayca tanımlar ve o değişkenlere istemli (random) erişim sağlar. Ancak *array* tipinin iki önemli handikapı vardır:

1. Array 'in öğeleri aynı veri tipinden olmalıdır,

2. Array 'in boyutu (öğe sayısı) önceden belli edilmelidir.

Oysa, programlama işinde, çoğunlukla aynı veri tipinden olmayan topluluklarla karşılaşırız. C# bu tür toplulukları ele alabilmek için, array yapısından çok daha genel olan koleksiyon (collection) veri tipini getirmiştir. Koleksiyon veri tpi, array veri tipinin çok kullanışlı bazı özelliklerini herhangi bir nesneler topluluğuna taşıma olanağı sağlamıştır.

Koleksiyon sınıfları nesnelerden oluşan topluluklardır. C#, koleksiyonları oluşturmak, koleksiyona yeni öğe katmak, koleksiyondan öğe atmak, koleksiyonun öğelerini sıralamak, numaralamak, koleksiyon içinde öğe aramak vb işleri yapmamızı sağlayan sınıflar, metotlar ve arayüzlerden oluşan çok geniş bir kütüphaneye sahiptir. Ayrıca, kullanıcı kendi koleksiyonunu oluşturabilir, onlarla istediği işi yapmayı sağlayacak metotları ve arayüzleri oluşturabilir.

C# dilinde koleksiyonlar `System.Collections` aduzayı (namespace) içinde birer veri tipidir. Klasik dillerdeki array yapısının çok daha gelişmiş biçimleridir. Bu bölümde C# dilinin koleksiyonlarından bazılarını oluşturacağız. Burada yapacaklarımız, başka koleksiyonların kullanımı için de yeterli olacaktır.

Ön-tanımlı koleksiyonlar

`System.Collections` aduzayı (namespace) içinde hemen kullanılmaya hazır koleksiyonlar vardır. Onlardan bazılarını ele alacağız. Bunların ilki `ArrayList` koleksiyonudur.

ArrayList Sınıfı

`ArrayList` sınıfı objelerden oluşan array yaratır. Array 'e bir obje eklemek için `Add()` metodu kullanılır.

Koleksiyon01.cs

```
using System;
using System.Collections;

namespace Koleksiyonlar
{
    class Dizi
    {
        static void Main(string[] args)
        {
            ArrayList birDizi = new ArrayList();
            birDizi.Add(12);
            birDizi.Add(3);
            birDizi.Add(8);
            birDizi.Add(7);
            birDizi.Add(15);

            foreach (int n in birDizi)
                Console.WriteLine(n.ToString());
        }
    }
}
```

ArrayList 'in öğelerini sıralamak (sort) için `Sort()` metodunu kullanırız. Aşağıda önce sıralanmamış liste, sonra sıralanmış liste yazılmaktadır.

Koleksiyon02.cs

```
using System;
using System.Collections;

namespace Koleksiyonlar
{
    class Dizi
    {
        static void Main(string[] args)
        {
            ArrayList birDizi = new ArrayList();
            birDizi.Add("Zonguldak");
            birDizi.Add("Urfa");
```

```
            birDizi.Add("Adana");
            birDizi.Add("Bursa");
            birDizi.Add("İzmir");
            Console.WriteLine("Sıralanmamış Liste");
            foreach (string s in birDizi)
                Console.WriteLine(s.ToString());
            Console.WriteLine();
            Console.WriteLine("Sıralanmış Liste");
           birDizi.Sort();
            foreach (string s in birDizi)
                Console.WriteLine(s.ToString());
        }
    }
}
```

Aşağıdaki program `ArrayList` 'i yaratmak için `ArrayYap()` metodunu ve yaratılan array'in öğelerini listeleyen `ArrayYaz()` metotlarını tanımlamıştır. Kodları dikkatle inceleyiniz.

Koleksiyon03.cs

```
using System;
using System.Collections;

namespace Koleksiyonlar
{
    class Diziler
    {
        public static void DiziYap(ArrayList arr)
        {
            for (int k = 1; k <= 10; k++)
              arr.Add(k);
        }

        public static void DiziYaz(ArrayList arr)
        {
            foreach (int n in arr)
                Console.WriteLine(n.ToString());
        }
    }

    class Uygulama
    {
        static void Main(string[] args)
        {
            ArrayList birDizi = new ArrayList();
            Diziler.DiziYap(birDizi);
            Diziler.DiziYaz(birDizi);
        }
    }
}
```

Yaratılan bir `ArrayList`' ten istenen öğeler atılabilir. Bunun için `Remove()` metodu kullanılır. Aşağıdaki program, yukarıda oluşan array'in bir öğesini attıktan sonra klan öğeleri tekrar listelemektedir.

Koleksiyon04.cs

```
using System;
using System.Collections;

namespace Koleksiyonlar
{
    class Diziler
    {
        public static void DiziYap(ArrayList arr)
        {
            for (int k = 1; k <= 10; k++)
              arr.Add(k);
        }

        public static void DiziYaz(ArrayList arr)
        {
            foreach (int n in arr)
                Console.WriteLine(n.ToString());
        }
    }

    class Uygulama
    {
        static void Main(string[] args)
        {
            ArrayList birDizi = new ArrayList();
            Diziler.DiziYap(birDizi);
            Diziler.DiziYaz(birDizi);
            birDizi.Remove(8);
            Diziler.DiziYaz(birDizi);
        }
    }
}
```

Alıştırma

Yukarıdaki programlarda `static` nitelemesi alan `ArrayYap()` ve `ArrayYaz()` metotlarını dinamik kılarak; yani static nitelemelerini kaldırarak kodları yeniden yazıp çalışır hale getiriniz.

C# dilinin bütün tipleri (sınıfları) `Object` tipinden türetilmiştir; yani bütün sınıfların atası `Object` 'dir. Dolayısıyla, koleksiyonların nesnelerden (object) oluşuyor olması doğaldır. Bazı koleksiyonlar aynı tipten objeleri içerir. Bazıları farklı tipten objeleri içerebilir. Örneğin, `array` sınıfına ait bir nesne aynı tipten öğeleri içerebilir. Ama `ArrayList` sınıfına ait nesneler farklı tipten nesneleri içerebilmektedir. Aşağıdaki program farklı tiplerden oluşan bir `ArrayList` oluşturmaktadır.

Koleksiyon05.cs

```
using System;
using System.Collections;
using System.Collections.Generic;


namespace Collections
{
    class Koleksiyon
    {
        static void Main()
        {
```

```
            int i = 10;
            double d = 17.3;
            ArrayList arrayList = new ArrayList();
            arrayList.Add("Başkent");
           arrayList.Add(i);
           arrayList.Add(d);
            for (int index = 0; index < arrayList.Count; index++)
                Console.WriteLine(arrayList[index]);
        }
    }
}
```

ArrayList 'in başlangıç kapasitesi 16 dır, ama 17-inci öğe geldiğinde kapasite 1 artar ve her yeni öğe gelişte bu olgu tekrarlanır. Ancak, her yeni öğenin gelişinde ona bellekte bir yer ayırma ve geleni oraya yerleştirme performansı düşürecektir. O nedenle, istenirse, başlangıçta ArrayList 'in kapasitesi, capacity özgeni kullanılarak belirlenebilir. Başka bir seçenek olarak, ArrayList sınıfının aşkın bir kurucusu kullanılabilir. Aşağıdaki program, bunun nasıl yapılacağını göstermektedir.

Koleksiyon06.cs

```
using System;
using System.Collections;
using System.Collections.Generic;


namespace Collections
{
    class Koleksiyon
    {
         static void Main()
        {
            int i = 10;
            double d = 17.3;
            ArrayList arrayList = new ArrayList();
           arrayList.Capacity = 2;
            arrayList.Add("Başkent");
           arrayList.Add(i);
           arrayList.Add(d);
            for (int index = 0; index < arrayList.Count; index++)
                Console.WriteLine(arrayList[index]);
        }
    }
}
```

StringCollection Sınıfı

`StringCollection` sınıfı `IList` arayüzünü oldurur ve stringlerden oluşan `ArrayList`'e benzer. Aşağıdaki program `StringCollection` sınıfının nasıl kullanılacağını göstermektedir.

Koleksiyon07.cs

```
using System;
using System.Collections;
using System.Collections.Specialized;

class Koleksiyonlar
```

```
{
  static void Main()
  {
    StringCollection stringList = new StringCollection();
    stringList.Add("Manisa");
    stringList.Add("Konya");
    stringList.Add("Kayseri");
    stringList.Add("Van");

    foreach (string str in stringList)
    {
      Console.WriteLine(str);
    }

  }
}
```

StringDictionary Sınıfı

`StringDictionary` sınıfı, anahtarları string olan bir Hashtable 'dir. Hashtable genel olarak her tipten anahtar kabul eder. Dolayısıyla, `StringDictionary`, anahtarları `string` sınıfına kısıtlı bir Hashtable 'dir. Aşağıdaki örnek `StringDictionary` sınıfının nasıl kullanıldığını göstermektedir.

Koleksiyon08.cs

```
using System;
using System.Collections;
using System.Collections.Specialized;

class Test
{
    static void Main()
    {
        StringDictionary stringList = new StringDictionary();
        stringList.Add("A", "Manisa");
        stringList.Add("B", "Konya");
        stringList.Add("C", "Kayseri");
        stringList.Add("D", "Van");

        foreach (string str in stringList.Values)
        {
            Console.WriteLine(str);
        }

    }
}
```

Stack Sınıfı

`Stack` sınıfı programlamada LIFO (Last-in-First-out , son giren ilk çıkar) diye bilinen önemli bir yapıdır. Bunu üst üste yığılan bir kitap koleksiyonu gibi düşününüz. Yığındakilere zarar vermeden istenen kitaba ulaşmak için, en üstteki kitap alınır ve bu işlem istenen kitaba erişinceye kadar devam eder. Stack 'a bir öğe eklemek için `Push()` metodu, stack 'ın üstünden bir öğe çekmek için `Pop()` metodu kullanılır. Aşağıdaki program `Stack` sınıfının nasıl kullanıldığını göstermektedir.

Koleksiyon09.cs

```
using System;
using System.Collections;

class Test
{
    static void Main()
    {
        Stack stackObject = new Stack();
        stackObject.Push("Sinop");
        stackObject.Push("Diyarbakır");
        stackObject.Push("Edirne");
        stackObject.Push("Muş");
        stackObject.Push("Aydın");
        stackObject.Push("Trabzon");
        while (stackObject.Count > 0)
            Console.WriteLine(stackObject.Pop());
    }
}
```

Queue Sınıfı

`Queue` sınıfı programlamada FIFO (First-in-First-out , ilk giren ilk çıkar) diye bilinen önemli bir yapıdır. Bu tam anlamıyla gündelik hayatta yaşadığımız kuyruk oluşturmayı andırır. Banka veya alış-veriş merkezlerinin vezne kuyruğunun oluşması gibidir. Kuyruğa ilk giren, veznedeki işini biirip ilk çıkan olacaktır. Arkasındakiler de aynı sıraya uyacaklardır. `Queue` (kuyruk) yapısı `Stack` yapısının yaptığının tersini yapar. `Queue` 'nun sonuna bir öğe eklemek için `Enqueue()` metodu, önünden bir öğe çekmek için `Dequeue()` metodu kullanılır. Aşağıdaki program `Queue` sınıfının nasıl kullanıldığını göstermektedir.

Koleksiyon10.cs

```
using System;
using System.Collections;

class Test
{
    static void Main()
    {
        Queue queueObject = new Queue();
        queueObject.Enqueue("Lale");
        queueObject.Enqueue("Gül");
        queueObject.Enqueue("Menekşe");
        queueObject.Enqueue("Sümbül");
        queueObject.Enqueue("Çiğdem");
        queueObject.Enqueue("Yasemin");
        while (queueObject.Count > 0)
            Console.WriteLine(queueObject.Dequeue());
    }
}
```

BitArray Sınıfı

`BitArray` sınıfı bit'lerden oluşan array yaratır. Aşağıda gösterildiği gibi, bit arraylerine *true* veya *false* değeri atanabilir.

```
BitArray bitArray = new BitArray(5,false);
```

ya da

```
BitArray bitArray = new BitArray(5,true);
```

Yukarıdaki sınıflarda olduğu gibi `BitArray` sınıfının da, öğe sayısını bildiren `Count` özgeni vardır. `BitArray` sınıfı öğeleri arasında aşağıdaki mantıksal işlemleri yapar.

```
And
Or
Not
Xor
```

Hashtable Sınıfı

`Hashtable` sınıfı, `Object` tiplerin hızla depolanması ve depodan hızla çekilmesi için iyi yöntemleri olan bir yapıdır. Anahtarlara dayalı arama yapar. Anahtarlar belli tiplere hasredilmiş hash kodlardan ibarettir. `GetHashCode()` metodu yaratılan bir nesnenin hash kodunu verir. Aşağıdaki program parçası `Hashtable` sınıfının nasıl kullanıldığını göstermektedir.

Koleksiyon11.cs

```
using System;
using System.Collections;

class Test
{
    static void Main()
    {
        Hashtable hashTable = new Hashtable();
        hashTable.Add(1, "Gökova");
        hashTable.Add(2, "Belek");
        hashTable.Add(3, "Çamdibi");
        hashTable.Add(4, "Marmaris");
        Console.WriteLine("Anahtarlar:--");
        foreach (int k in hashTable.Keys)
        {
            Console.WriteLine(k);
        }

        Console.WriteLine("Aramak için anahtarı giriniz :");
        int n = int.Parse(Console.ReadLine());
        Console.WriteLine(hashTable[n].ToString());
    }
}
```

`Hashtable` objelerini listelemek için, yukarıdaki listeleme yöntemi yerine, `IdictionaryEnumerator` 'i kullanarak aşağıda gösterildiği gibi de yapabiliriz.

Koleksiyon12.cs

```
using System;
using System.Collections;

class Test
{
    static void Main()
    {
        Hashtable hashTable = new Hashtable();
        hashTable.Add(1, "Matematik");
        hashTable.Add(2, "Fizik");
        hashTable.Add(3, "Kimya");
        hashTable.Add(4, "Biyoloji");
        hashTable.Add(5, "Bilgisayar");
        hashTable.Add(6, "Jeoloji");
        Console.WriteLine("Anahtarlar:--");
        IDictionaryEnumerator en = hashTable.GetEnumerator();
        string str = String.Empty;

        while (en.MoveNext())
        {
            str = en.Value.ToString();
             Console.WriteLine(str);
        }
    }
}
```

Hashtable sınıfından bir öğe atmak için Remove() metodu kullanılır. Örneğin,

```
hashTable.Remove(4) ;
```

deyimi, aşağıdaki listeden "Biyoloji" yi atacaktır.

SortedList Sınıfı

`SortedList` sınıfı `System.Object` tiplerini anahtar-değer çiftine göre yerleştirir; ayrıca sıralama yapar. Aşağıdaki program bunu gösteriyor.

Koleksiyon13.cs

```
using System;
using System.Collections;
using System.Collections.Specialized;

class Test
{
    static void Main()
    {
        SortedList sortedList = new SortedList();
        sortedList.Add(1, "Matematik");
        sortedList.Add(2, "Fizik");
        sortedList.Add(3, "Kimya");
        sortedList.Add(4, "Biyoloji");
        sortedList.Add(5, "Bilgisayar");
        sortedList.Add(6, "Jeoloji");

        IDictionaryEnumerator en = sortedList.GetEnumerator();
```

```
        Console.WriteLine("Listeyi giriliş sırasıyla yazar:");
        foreach (string str in sortedList.Values)
        {
            Console.WriteLine(str);
        }

    }
}
```

Çıktı

Listeyi giriliş sırasıyla yazar:

```
Matematik
Fizik
Kimya
Biyoloji
Bilgisayar
Jeoloji
```

Aynı listeyi `IdictionaryEnumerator` 'i kullanarak da yazdırabiliriz.

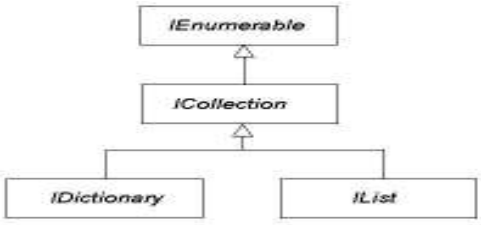
Koleksiyon14.cs

```
using System;
using System.Collections;
using System.Collections.Specialized;

class Test
{
    static void Main()
    {
        SortedList sortedList = new SortedList();
        sortedList.Add(1, "Matematik");
        sortedList.Add(2, "Fizik");
        sortedList.Add(3, "Kimya");
        sortedList.Add(4, "Biyoloji");
        sortedList.Add(5, "Bilgisayar");
        sortedList.Add(6, "Jeoloji");

        IDictionaryEnumerator en = sortedList.GetEnumerator();
        string str = String.Empty;
        while (en.MoveNext())
        {
            str = en.Value.ToString();
            Console.WriteLine(str);
        }
    }
}
```

Bütün koleksiyonlar *IEnumerable* arayüzünü oldururlar. *IEnumerable* arayüzü bütün koleksiyonların atasıdır. *ICollection* arayüzü `IEnumerable` arayüzünden türemiştir; yani oğuldur. Onun da iki tane oğulu vardır: `Idictionary` ve `IList`.



`IList` arayüzü değerler (value) koleksiyonudur. Onun oldurduğu koleksiyonlar şunlardır:

System.Array
System.Collections.ArrayList
System.Collections.Specialized.StringCollection

`IDictionary` arayüzü (Anahtar, Değer) [(Key, Value)] çiftlerinden oluşan koleksiyonlardır. Bunun oldurduğu koleksiyonların listesi şudur:

System.Collections.Hashtable
System.Collections.Specialized.ListDictionary
System.Collections.SortedList
System.Collections.Specialized.HybridDictionary

`ICollection` arayüzünden türetilen başka koleksiyonlar:

System.Collections.BitArray
System.Collections.Stack
System.Collections.Queue
System.Collections.Specialized.NameValueCollection

Bunların dışında olan koleksiyonlar da vardır. Örneğin, System.Collections.Specialized.StringDictionary, System.Collections.Specialized.BitVector32. Bunların ayrıntıları için *msdn* web sitesine bakınız.

Koleksiyon15.cs

```
using System;
using System.Collections;
using System.Collections.Generic;


namespace Collections
{
    class Koleksiyon
    {
        static void Main()
        {
            String[] names =new String[2] {"Joydip","Jini"};
for(IEnumerator e =
```

```
names.GetEnumerator();e.MoveNext();Response.Write(e.Current));

String[] adlar = new String[2] {"Joydip","Jini"};
foreach(string str in adlar)
Response.Write(str);
        }
    }
}
```

Alıştırma

Aşağıdaki program bir liste yaratmaktadır. Çıktısı ile karşılaştırarak programı çözümleyiniz.

Koleksiyon16.cs

```
using System;
public class ListeYap
{
    static int[] list = new int[5];
    static int sayaç = 0;
    static void add(int k)
    {
        if (sayaç == list.Length)
        {
            int[] newlist = new int[(int)(sayaç +1)];
            for (int i = 0; i < sayaç; i++)
              newlist[i] = list[i];
           list = newlist;
        }
        list[sayaç++] = k;
    }
    public static void Main(string[] args)
    {
        for (int i = 1; i <= 20; i++)
           add(i);
        Console.WriteLine("length = " + sayaç);
        Console.Write("kents = ");
        for (int i = 0; i < sayaç; i++)
            Console.Write(list[i] + " ");
        Console.WriteLine();
    }
}
```

Çıktı

```
length = 20
elements = 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
```