6. KOLLEKSİYON SINIFLARI

Programlamada yığın halinde verilerle çalışırken, bu veriler üzerinde toplu olarak işlem yapmamız gerektiğinde dizilerden faydalanırız. Diziler aynı tipteki çok sayıda veriyi tek bir isim altında tutmamıza erişmemize ve kullanmamıza olanak sağlayan yapılardır. Ancak dizilerin bazı kısıtları vardır. Diziye yeni bir eleman eklemek, aradan bir veriyi çıkarmak, araya bir veri eklemek, eleman sayısını değiştirmek, farklı tiplerdeki elemanları dizide saklamak çok zor ve zahmetli işlemlerdir. Bazıları da mümkün değildir.

İşte bu durumlarda, bu kısıtlardan kurtulmak ve daha esnek veri kullanımına olanak sağlamak için koleksiyon sınıflarından faydalanırız. Koleksiyon sınıfları ile dizileri kullanarak yapamadığımız birçok işlem çok kolay bir şekilde gerçekleştirilir. Koleksiyon sınıflarını kullanabilmek için isim uzayına aşağıdaki eklemeyi yapmamız gerekir.

Using System.Collection;

Şimdi kullanabileceğimiz isim uzaylarını ve özelliklerini inceleyelim.

6.1. ARRAYLİST

Array List içerisindeki veriler object tipindedir. Eleman sayısı ve içerisine eklenecek veri tipleri ile ilgili bir sınırlama yoktur. Nesne tanımı;

ArrayList Liste_adı=new ArrayList();

● Veri ekleme. Listeye veri eklemek için .Add() metodu kullanılır.

Liste_adı.Add(veri);

Veri metinsel ya da sayısal olabilir. Eğer metinsel veri eklenecek ise ver çift tırnak içerisinde yazılmalıdır.

Liste_adı.Add(“Resul”); // Resul bilgisi listeye eklendi

Eğer veri sayısal ise doğrudan verilir.

Liste_adı.Add(40);

● Araya veri ekleme. Veri, istenilen index belir=lerek liste içerisinde bir araya da yerleştirilebilir. Bunun için .Insert() metodu kullanılır. Bu metotta index numarası ve veri bildirilir.

Liste_adı.Insert(2, “Tuna”); // Listenin 3. Sırasındaki eleman Tuna oldu

● Bir diziyi listeye ekleme. Var olan bir dizi de .AddRange() metodu kullanılarak Arrayliste eklenebilir.

Liste_adı.AddRange(dizi_adı); // belirtilen simdeki dizi listenin tümü listenin sonuna eklenir

● Bir diziyi araya girme. Var olan bir dizi ArraylisBe istenilen bir araya eklenebilir. Bunun için

Liste_Adi.InsertRange()

Metodu kullanılır. Metoda hangi indexten sonra ekleme yapacağımız ve hangi diziyi ekleyeceğimiz parametre olarak bildirilebilir.

Liste_adı.InsertRange(2, isimler) // isimler adında bir dizi 2 numaralı indexten itibaren listeye eklendi

● Listeden veri silme. ArrayListten veri silmek için iki metot kullanılır. Remove metodu silinmek istenilen veri bildirilerek silme yapar. RemoveAt metodu ise silinmek istenilen verinin index numarası bildirilerek silme işlemi yapar.

Liste_adı.Remove(“Resul”); // Resul verisi listeden silindi

Liste_adı.RemoveAt(“3”) // Indexi 3 olan yani listedeki 4. Veri listeden silindi

ÖĞR. GÖR. RESUL TUNA

● Listeden bir veri grubunu silme. Listeden tek bir veri değil de bir grup ver silinmek istenirse

.RemoveRange() metodu kullanılır. Bu metoda silmek istenilen index numarasından itibaren kaç eleman silineceği bildirilir.

Liste_adı.RemoveRange(1,3); // Listenin 1 numaralı indexinden itibaren 3 eleman silindi

● Arraylistte arama yapma. Arama iki şekilde yapılabilir. Veri var mı yok mu şeklinde arama yapmak için .Contains() metodu kullanılır. Geriye *true* ya da *false* değeri döner. Yerini öğrenmek için ise .IndexOf() metodu kullanılır. Geriye index numarasını döndürür. Eğer aranan veri listede yoksa -1 değeri döndürür.

Liste_adı.Contains(veri) // veri listede aranır, varsa *true* yoksa *false* değeri döner

İnt id = Liste_adı.IndexOf(veri); // veri listede aranır, varsa index numarası, yoksa -1 değeri döner.

Eğer aramaya listenin istenilen elemanından başlamak istenirse IndexOf metodunda veriden sonra sıra verilir.

İnt id = Liste_adı.IndexOf(veri,index_no) // Arama verilen index numarasından itibaren yapılır

● Listedeki eleman sayısını öğrenme. Arraylistteki eleman sayısını öğrenmek için .Count metodu kullanılır.

İnt es = Liste_adı.Count // listenin eleman sayısını verir

● Listeyi temizleme. Arraylistteki tüm verileri silmek için .Clear metodu kullanılır.

Liste_adı.Clear // Liste temizlendi

6.2 HASHTABLE

Hashtableda Arraylistten farklı olarak veriler bir anahtar (key) ile birlikte listeye eklenir. Eklenen her verinin bir de anahtarı bulunur veriye erişim genellikle bu anahtar aracılığı ile gerçekleşir.

Hashtable nesnesinin tanımı;

Hashtable Liste_adı = new Hashtable();

● Listeye veri ekleme. Hashtable nesnesine veri eklemek için .Add(“key”, “veri”) metodu kullanılır.

Liste_adı.Add(“isim”, “Resul”); // isim adında bir alana Resul verisi eklendi

Liste_adı.Add(“soyisim”, “Tuna”); // soyisim adında bir alana Tuna verisi eklendi

Add metodu kullanılmadan da veri eklemesi gerçekleştirilebilir. Bunun için;

Liste_adı [“key”] = veri   //şeklinde veri girişi yapılır.

Liste_adı [“yaşı”] = 40;   // yaşı adında bir alana 40 verisi eklendi

● Listede arama yapma. Arama keyler arasında ya da veriler arasında yapılabilir. Bir keyin var olup olmadığını .ContainsKey(“key”) metodu ile yapabiliriz. Bir verinin var olup olmadığını ise .ContainsValue(“veri”) metodu ile yapabiliriz. Her iki metot da geriye *true* ya da *false* değeri döndürür.

Liste_adı.ContainsKey(“soyisim”) // listede soyisim adında bir anahtar alan olup olmadığını verir.

Liste_adı.ContainsValue(“Resul”) // listede Resul verisinin olup olmadığını döndürür.

ÖĞR. GÖR. RESUL TUNA

● Verinin kendisine ulaşma. Verinin kendisine ulaşmak için keyi kullanılır. Liste_adı[“key”] şeklinde kullanılır.

Liste_adı[“isim”].ToString() // Listedeki etiketi isim olan veri geri döndürülür. Hashtabledaki veriler object tipinde olduğu için ekran çıktılarında verinin Convert edilmesi gerekir.

● Verileri listeleme. Hashtable içerisindeki veriler object tipinde olduğundan ve veriler index numaralarına göre değilde keylerine göre saklandığından bir standart döngü ile verilere sırası ile erişmek mümkün olmaz.

Koleksiyon sınıflarına eklenen tüm veriler **DictionaryEntry** nesnesine eklenir bu nedenle listeleme işlemini verileri bu nesneden alarak gerçekleştiririz. Bir örnek ile inceleyelim.

Listedeki verileri bir listbox’ta gösterelim;

```
foreach(DictionaryEntry veri in Liste_adı)
{
listbox1.Items.Add(veri.Key + “ ; ” + veri.Value);
//  listedeki anahtarlar ve veriler listboxa yazıldı
}
```

● Listeden veri silme. Hashtable nesnesinden veri silmek için .Remove(key) metodu kullanılır.

Liste_adı.Remove(“isim”); // isim keyi veri ile birlikte listeden silindi

6.3. QUEUE

Bu sınıf bellek alanlarını (yığın) kullanmak için gerekli olan listelerdir. Listeye ilk giren ilk çıkar (FIFO – First in First out) ilkesi ile çalışır. Bunu şu şekilde örneklendirebiliriz. Kendimizi bankada bir veznedar olarak düşünelim. Kuyrukta bekleyen birçok kişi var. Tabii ki kuyruktaki en öndeki kişinin işi en erken bitecek ve kuyruktan en önce o ayrılacaktır. Kuyruğun en arkasındaki kişinin işi ise en geç bitecektir. Bizim veznedar olarak en arkadaki kişinin işini halledebilmemiz için o kişinin önündeki kişilerin de işlerini bitirmemiz gerekir. Nesneyi oluşturmak için;

```
Queue Liste_adı = new Queue();
```

● Listeye veri ekleme. Queue nesnesine veri eklemek için .Enqueue() metodu kullanılır.

Liste_adı.Enqueue(“Resul”);

Liste_adı.Enqueue(“Nursema”);

Liste_adı.Enqueue(“Albina”); // veriler sırası ile listeye eklendi

● Listeden veri çağırma. Listeden bir veri çağırmak için iki farklı metot kullanılabilir. Bunlardan biri Dequeue() metodudur. Bu metot ile veri çağrılır ve aynı zamanda listeden de silinir. Gelin veri listeye ilk eklenen veridir.

Diğer bir veri çağırma metodu ise .Peek() metodudur. Bu metodu veriyi listeden silmeden sadece çağırmak istersek kullanabiliriz. Gelen veri yine listenin başındaki veridir.

Dequeue metodunun kullanımı;

Liste_adı.Dequeue().ToString(); // Listeye eklenmiş olan ilk veri çağrıldı ve listeden silindi

Aynı metot tekrar çağrılırsa, ilk veri listeden silindiği için daha önce eklenmiş olan ikinci veri listenin başında olacağından ikinci veri çağrılmış olur.

Peek metodunun kullanımı;

Liste_adı.Peek().ToString(); // Listeye eklenmiş olan ilk veri çağrıldı, listeden silinmedi

ÖĞR.GÖR.RESUL TUNA

Aynı metot tekrar çağrılırsa, alınan ilk veri listeden silinmediği için yine listedeki ilk veri bize getirilecektir. Bu metodu her çalıştırdığımızda aynı veriye ulaşırız.

● Listeyi temizleme. Queue nesnesindeki verileri temizlemek için .Clear() metodu kullanılır.

Liste_adı.Clear(); // Queue nesnesine eklenmiş olan tüm veriler silindi

6.4. STACK

Bu sınıf bellek alanlarını (yığın) kullanmak için gerekli olan listelerdir. Listeye son giren ilk çıkar (LIFO – Last in First out) ilkesi ile çalışır. Yığınları üst üste koyulmuş kitaplar gibi düşünebiliriz. En üstteki kitabı rahatlıkla alıp okuyabiliriz. Ancak okuyacağımız kitap ortalarda veya altlarda ise önce okuyacağımız kitabın üstündeki kitapları teker teker yığından çıkarıp sonra okuyacağımız kitabı almalıyız. Öbür türlü kitap destesi devrilebilir. İşte yığınlarda da aynı mantık söz konusudur. Yığınların mantığını şu cümleyle özetleyebiliriz:

"En önce giren en son çıkar, en son giren en önce çıkar." Yani yığına eklenen her bir elemanın birbirinin üstüne geldiğini söyleyebiliriz. Stack sınıfında tıpkı diğer koleksiyonlardaki gibi eleman eklendikçe otomatik olarak kapasite artar. Nesneyi oluşturmak için;

```
Stack Liste_adı = new Stack();
```

● Listeye veri ekleme. Stack nesnesine veri eklmek için .Push() metodu kullanılır.

Liste_adı.Push(“Resul”);

Liste_adı.Push(“Nursema”);

Liste_adı.Push(“Albina”); // veriler sırası ile listeye eklendi

● Listeden veri çağırma. Listeden bir veri çağırmak için iki farklı metot kullanılabilir. Bunlardan biri Pop() metodudur. Bu metot ile veri çağrılır ve aynı zamanda listeden de silinir. Gelen veri listeye en son eklenen veridir.

Diğer bir veri çağırma metodu ise .Peek() metodudur. Bu metodu veriyi listeden silmeden sadece çağırmak istersek kullanabiliriz. Gelen veri yine listenin sonundaki veridir.

Pop metodunun kullanımı;

Liste_adı.Pop().ToString(); // Listeye eklenmiş olan son veri çağrıldı ve listeden silindi

Aynı metot tekrar çağrılırsa, ilk veri listeden silindiği için sondan bir önceki veri listenin sonunda olacağından son ikinci veri çağrılmış olur.

Peek metodunun kullanımı;

Liste_adı.Peek().ToString(); // Listeye eklenmiş olan son veri çağrıldı, listeden silinmedi

Aynı metot tekrar çağrılırsa, alınan listenin sonundaki veri listeden silinmediği için yine listedeki son veri bize getirilecektir. Bu metodu her çalıştırdığımızda aynı veriye ulaşırız.

ÖĞR. GÖR. RESUL TUNA