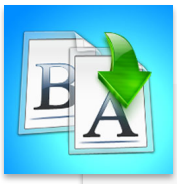
KOLEKSİYONLAR-I



İÇİNDEKİLER

- Koleksiyon Tanımı
- Non Generic Koleksiyonlar
  - ArrayList Sınıfı
  - HashTable Sınıfı
  - SortedList Sınıfı

HEDEFLER

- Bu üniteyi çalıştıktan sonra;
  - Koleksiyon kavramının tanımını yapabilecek,
  - Non generic koleksiyonlar hakkında bilgi sahibi olabilecek,
  - ArrayList, HashTable ve SortedList sınıflarının nasıl kullanıldığını öğrenebileceksiniz.

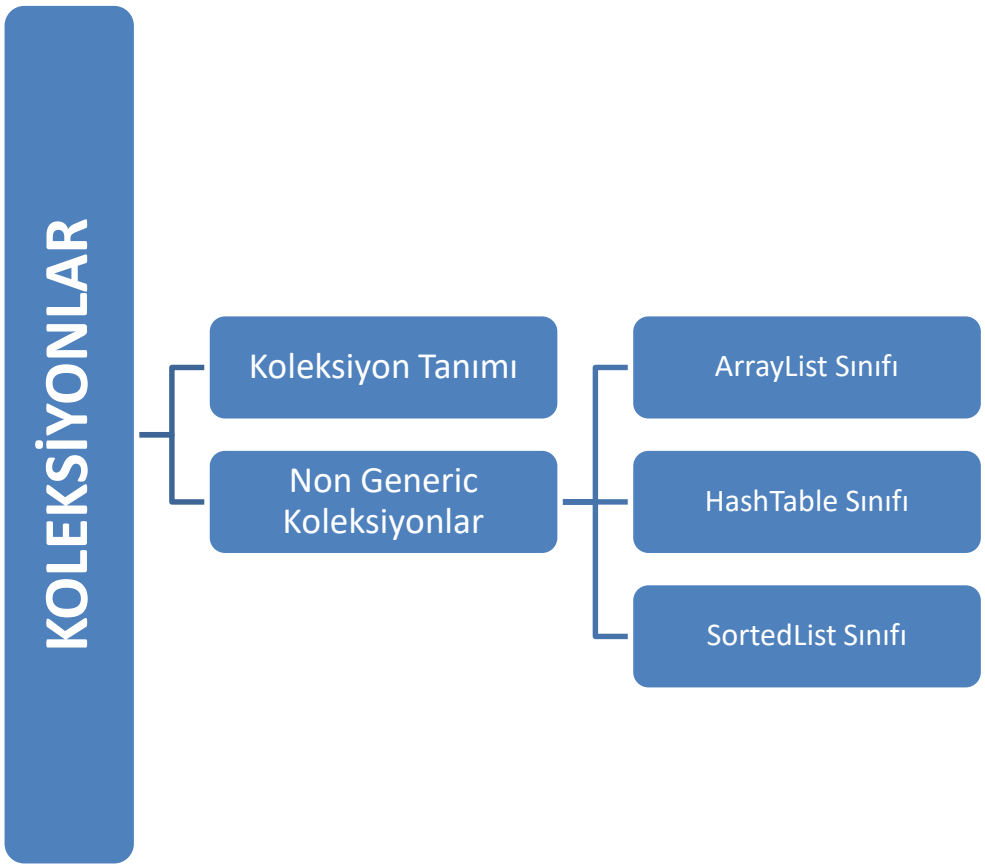
Atatürk Üniversitesi
Açıköğretim Fakültesi

NESNE TABANLI PROGRAMLAMA I

Dr. Orhan ÇELİKER

ÜNİTE

12


KOLEKSİYONLAR
Koleksiyon Tanımı
Non Generic Koleksiyonlar
ArrayList Sınıfı
HashTable Sınıfı
SortedList Sınıfı

GİRİŞ

Koleksiyonlar, programlamada sıklıkla ihtiyaç duyulan yapılardan olup *System.Collections* isim uzayı altında yer alırlar. Bu yapılar içerisinde aynı türden veriler saklanabildiği gibi farklı türden veriler de saklanabilir. Bunun için Visual Studio.Net'in sunmuş olduğu çeşitli koleksiyon sınıfları bulunmaktadır. C++ dilinde *konteyner*, Java dilinde *Collections Framework* olarak adlandırılan bu sınıflar, C# dilinde ise *koleksiyon* olarak adlandırılmaktır.

Koleksiyonlar, amaç ve yapı bakımından dizilerle benzerlik gösterse de hem verilerin saklanması hem de saklanan verilere erişilmesi açısından dizilerden ayrıldığı belli başlı noktalar bulunmaktadır. Bunların başında ise içerdiği çok çeşitli sınıflar gelmektedir. Bu sınıflar aracılığıyla geliştiriciler, nesne grupları ve veriler üzerinde daha esnek bir şekilde çalışma imkânı bulurlar.

Bu ünitede koleksiyonların tanımına değinilecek ardından *non generic* koleksiyon sınıflarına değinilecektir. Programlamada sıkça kullanılan koleksiyon tiplerinin daha iyi anlaşılması için de kod örneklerine yer verilecektir. Ünitede yer alan kod örneklerini ve bireysel etkinliği kendi bilgisayarınızda uygulamanız koleksiyon mantığını daha iyi anlamanız ve konuyu pekiştirmeniz açısından faydalı olacaktır.

KOLEKSİYON TANIMI

Koleksiyonlar, temel olarak aynı ya da farklı türde birden fazla verinin veya nesnenin gruplanıp saklanmasına ve bunlara erişim sağlanmasına olanak tanıyan bir sınıf olarak tanımlanabilir.

Koleksiyonlar; C++, Java gibi diğer programlama dillerindeki kullanım amacına bakıldığında temel olarak aynı ya da farklı türde birden fazla verinin veya nesnenin gruplanıp saklanmasına ve bunlara erişim sağlanmasına olanak tanıyan bir sınıf olarak tanımlanabilir. Bu tanıma göre dizilerle benzerlik gösterdiği görülebilir ancak koleksiyonların dizilerden ayrıldığı temel farklılıklar bulunmaktadır. Bunlardan en önemlisi dizilerde aynı türden veriler saklanabilir, koleksiyonlarda ise farklı türden verilerin saklanması mümkündür. Visual Studio.Net'te diziler ve kolaksiyonların birbirinden ayrıldığı temel farklılıklara Tablo 12.1'de yer verilmiştir.

Tablo 12.1. Dizi ve koleksiyon arasındaki temel farklılıklar

Diziler	Koleksiyonlar
System.Array isim uzayı altında yer alır.	System.Collections isim uzayı altında yer alır.
Sadece aynı türdeki verileri saklar.	Aynı veya farklı türdeki verileri saklayabilir.
İçerdiği elemanlar herhangi bir dönüşüm işlemine tabi tutulmaz.	İçerdiği elemanlar dönüşüme (boxing ve unboxing) tabi tutulabilir.
Tanımlama için *new* anahtar sözcüğü kullanılmak zorunda değildir.	*new* anahtar sözcüğü koleksiyon ismi ile kullanılmalıdır.
Diziler çok boyutlu olabilir.	Koleksiyonlar doğrusal olup çok boyutlu olamazlar.

Anahtar-değer çiftleri mantığı ile çalışan HashTable ve SortedList sınıfları IDictionary ara yüzünü uygulamaktadır.

System.Collections isim uzayının altında birçok *ara yüz* bulunmaktadır. Bu ara yüzler kullanılarak koleksiyon sınıfları için genel karakteristik tanımlaması yapılmaktadır. Örneğin *ArrayList* sınıfı koleksiyon sınıfları arasında sıklıkla kullanılır ve *IList* ara yüzünü uygulamaktadır. Bu yüzden bazı geliştiriciler tarafından *liste tipi koleksiyon* olarak adlandırılmaktadır. Aynı şekilde *HashTable* koleksiyon sınıfı ise *IDictionary* ara yüzünü kullanır ve bu da birçok geliştirici tarafından *sözlük tipi koleksiyon* olarak sınıflandırılır. Ara yüzlerle ilgili bilinmesi gereken bir başka nokta da koleksiyon sınıflarında ortak olan bazı özelliklerin uygulanan ara yüzler tarafından belirlenmesidir. Örneğin anahtar-değer çiftleri mantığı ile çalışan *HashTable* ve *SortedList* sınıfları *IDictionary* ara yüzünü uygulamaktadır. Tüm bu sebeplerden ötürü *System.Collections* isim uzayının altındaki ara yüzlerin bir kısmını genel hatlarıyla bilmek faydalı olacaktır. Tablo 12.2'de koleksiyon sınıflarının uyguladığı bazı ara yüzler hakkında kısaca bilgi verilmiştir.

Tablo 12.2. Bazı koleksiyon sınıfı ara yüzleri

Arayüz	Açıklama
ICollection	System.Collections isim uzayı altında bulunan tüm koleksiyonların sahip olduğu metotları ve temel özellikleri içerir.
IEnumerable	İçinde GetEnumerator() metodunu barındırır. Bu metot kullanılarak bir koleksiyon sınıfında ilerlemek için gerekli olan numaralandırıcı sağlanır.
IEnumerator	Birden fazla eleman olan koleksiyonlarda her bir elemana erişmeyi ve eleman içeriğinin elde edilmesini sağlar.
IComparer	İki nesneyi karşılaştıran bir metot içerir.
IDictionary	Anahtar-değer çifti mantığı ile çalışan koleksiyon sınıfları tarafından uygulanır.
IList	Koleksiyondaki elemanların her birine indeks numarası ile erişilmesine olanak tanır.

C# programlama dilinin sunduğu hazır koleksiyon sınıflarının dışında geliştiriciler, ihtiyaçları doğrultusunda kendi yazdıkları farklı koleksiyon sınıflarını da oluşturabilirler. Bu koleksiyon sınıfları farklı ara yüzleri barındırabilir ve amaca göre özelleştirilebilir. Bu yüzden C# tarafından hazır olarak sunulan ya da geliştiricinin kendi oluşturduğu her koleksiyon türünün belli bir amaca göre tasarlandığı söylenebilir. Farklı amaçlar için geliştirilen koleksiyonlar farklı yapılara sahip olacağı için kendi içerisinde gruplandırılmıştır. Genel kabule göre *Non Generic, Generic ve Specialized (özelleştirilmiş)* olmak üzere koleksiyonlar 3 grupta incelenebilir. Bu ünitede *non generic* koleksiyonlara değinilecektir.

NON GENERIC KOLEKSİYONLAR

System.Collections isim uzayının altında yer alan *non generic* koleksiyon sınıflarının temel özelliği farklı türdeki verilerin *boxing* işlemi ile *object* türüne dönüştürülerek saklanabilmesidir. Bu başlık altında *Non Generic* koleksiyonlar arasında en sık kullanılan *ArrayList, Hashtable* ve *SortedList* sınıfları incelenecektir.

ArrayList Sınıfı

ArrayList sınıfı, içerisine aktarılan değerlerin boxing işlemi ile object türüne dönüştürüldüğü ve eleman sayısı belirlenmeden tanımlama yapıldığı koleksiyon türüdür.

System.Collection isim uzayı altında yer alan *ArrayList* sınıfı, içerisine aktarılan değerlerin *boxing* işlemi ile *object* türüne dönüştürüldüğü ve eleman sayısı belirlenmeden tanımlama yapıldığı koleksiyon türüdür. Dizilerde farklı türde eleman barındırılamaması ve dizi boyutunun dinamik bir şekilde düzenlenememesi sebebiyle geliştiriciler tarafından *ArrayList* sınıfı sıklıkla tercih edilir.

ArrayList sınıfı içerisinde farklı türlerde değişkenler tutulabilir ve çalışma anında dizinin boyutu düzenlenebilir. Sağlanan bu esnekliklerin dışında C# programlama dilindeki kullanımı klasik dizilerle neredeyse aynıdır. ArrayList sınıfının genel sözdizimine yönelik kod örneği aşağıdaki gibidir:

```
ArrayList mevsimler = new ArrayList();
```

ArrayList sınıfının altında birçok metot yer alır. Bu metotlar kullanılarak eleman ekleme, çıkarma, kopyalama, sıralama gibi birçok işlem gerçekleştirilebilir. *ArrayList* sınıfının altında sıklıkla kullanılan metotların isimleri ve işlevleri Tablo 12.3'te görülmektedir.

Tablo 12.3. ArrayList sınıfına ait bazı metotlar

Metot Adı	Açıklama
Add	ArrayList içerisine eleman eklemek için kullanılır. Add metodu eklenen elemanın sıra numarasını geri döndürür.
Clear	ArrayList nesnesinin bütün elemanlarını siler ve eleman sayısı 0 olur.
Contains	ArrayList içerisinde arama yapmak için kullanılır. Tek parametre kabul eder ve bu parametrede verilen object tipindeki değerin, ArrayList elemanları arasında bulunup bulunmadığını true ya da false olarak geri döndürür.
CopyTo	ArrayList nesnesinin içerdiği elemanları başka bir koleksiyona kopyalamak için kullanılır.
Insert	Insert metodu kullanılarak ArrayList'in sonuna eleman eklenir.
Remove	Parametre olarak değeri verilen elemanı ArrayList'ten çıkarır.
RemoveAt	Parametre olarak indeks numarası verilen elemanı ArrayList'ten çıkarır.
Reverse	ArrayList'in içeriğini terse çevirmek için kullanılır.
Sort	ArrayList elemanlarını sıralamak için kullanılır.

Tablo 12.3'te verilen metotların neredeyse tamamı, farklı sınıfların metodu olarak karşılaşılabilir. Bunun sebebi aynı ara yüzü uygulayan sınıfların, bu ortak ara yüze ait sınıf ve metotları kullanabilmeleridir. Örneğin *Sort()* metodu hem dizide hem de ArrayList'te kullanılan bir metottur. Bunun sebebi *IList* ara yüzünü hem dizilerin hem de *ArrayList* sınıfının ortak olarak uygulamasıdır. Bir başka ifadeyle bazı metot ve özelliklerin birden fazla sınıfta ortak olması, bu sınıfların aynı ara yüzleri uygulamasından kaynaklanmaktadır.

Sort() metodu hem dizide hem de ArrayList'te kullanılan bir metottur. Bunun sebebi IList ara yüzünü hem dizilerin hem de ArrayList sınıfının ortak olarak uygulamasıdır.

Koleksiyonlar kullanılırken eleman ekleme ve silme işlemlerine sıklıkla başvurulur. *ArrayList* koleksiyonlarında bu işlem *Add()* metodu ile yapılır. *Add()* metodunun genel sözdizimi aşağıdaki gibidir:

Add(Eklenecek Değer);

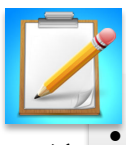
ArrayList sınıfına eleman eklemeyle ilgili Add() metodunun kullanımına yönelik kod örneği aşağıda verilmiştir.

```
ArrayList aylar = new ArrayList();
aylar.Add("Ocak");
aylar.Add("Şubat");
aylar.Add("Mart");
aylar.Add("Nisan");
```

Örnekten de görüldüğü gibi *aylar* isimli *ArrayList* koleksiyonuna *Add()* metodu kullanılarak "Ocak", "Şubat", "Mart" ve "Nisan" değerlerine sahip 4 eleman eklenmiştir. Eklenen elemanların tamamı aynı tipte olduğundan bu örnek için *ArrayList* koleksiyonu yerine *Array* dizisi de kullanılabilirdi. Ancak aşağıdaki örnekte olduğu gibi farklı tipten elemanların bir arada tutulmak istendiği durumlarda kesinlikle *ArrayList* kullanılmalıdır.

```
ArrayList sehirler = new ArrayList();
sehirler.Add("Erzurum");
sehirler.Add(25);
sehirler.Add(false);
sehirler.Add(19.18);
```

Yukarıdaki örnekte *sehirler* isimli *ArrayList* koleksiyonuna 4 farklı değişken tipinde değer içeren eleman eklenmiştir. Bu örnek kod çalıştırıldığında *string, int, bool* ve *double* tipindeki değerlere sahip elemanlar sorunsuz bir şekilde ArrayList'e eklenecektir. Eleman ekleme işlemi *Insert()* metodu kullanılarak da yapılabilir.



Bireysel Etkinlik

- Tablo 12.3'te yer alan ArrayList sınıfına ait metotlarla ilgili birer örnek yapınız.

HashTable Sınıfı

System.Collections isim uzayı altında yer alan *HashTable* sınıfı, barındırdığı elamanlara ait değerlerin indeks numarası yerine benzersiz bir değer kullanılarak saklandığı ve bu şekilde erişim sağlandığı koleksiyon türüdür. Koleksiyon elemanlarına indeks numarası yerine farklı bir değer ile erişilmesi gerekebilir. Aynı şekilde koleksiyon elemanlarının benzersiz bir değere sahip olması ve koleksiyon içerisinde arama yapılırken bu benzersiz değerin kullanılması gerekebilir. Şu ana kadar anlatılan *Array (dizi)* ve *ArrayList* sınıflarında elemanlara erişmek için *int* tipinde bir indeks numarası kullanılmış ve eleman arama, ekleme veya çıkarma gibi özellikler bu indeks numarasına göre gerçekleştirilmiştir. İndeks numarası kullanımı az sayıda elemana sahip koleksiyonlarda kullanışlı olabilir. Ancak özellikle performansın önemli olduğu durumlarda beklentileri karşılamayabilir. İşte bu tür durumlarda *HashTable* sınıfının kullanımı tercih edilebilir.

Genellikle performansın ön planda olduğu ve koleksiyon elemanlarına erişim için indeks numarası yerine benzersiz bir değer *(anahtar)* yardımıyla erişmek gerektiğinde *System.Collections* isim uzayı altında bulunan *HashTable* sınıfı kullanılmaktadır. *HashTable* sınıfının *ArrayList* sınıfından temel farkı ise çalışma algoritmasıdır.

HashTable içerisinde yer alan değerlere indeks numarası ile değil benzersiz anahtar (key) aracılığıyla erişilir.

HashTable sınıfında saklanan her eleman 2 parametre alır. Bu parametreler *değer(value)* ve *anahtar(key)* şeklindedir. *HashTable* içerisinde yer alan değerlere indeks numarası ile değil benzersiz *anahtar (key)* aracılığıyla erişilir. Aşağıda verilen örnekte bir *HashTable* sınıfı tanımlanmış ve *anahtar – değer çiftleri* eklenmiştir.

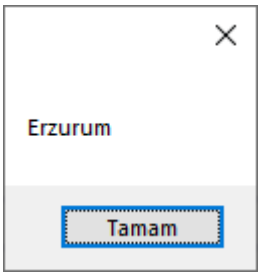
```
Hashtable sehirler = new Hashtable();
sehirler.Add(25, "Erzurum");
sehirler.Add(34, "İstanbul");
sehirler.Add(35, "İzmir");
sehirler.Add(54, "Sakarya");
```

Yukarıdaki kod örneğinden de görüldüğü gibi *Add()* metodu iki parametre almaktadır. *Bu parametrelerden ilki anahtar, ikincisi ise değer olarak kabul edilmektedir.* Örneğe göre "Erzurum" değerinin anahtarı 25'tir ve bu anahtar kullanılarak "Erzurum" elemanına erişilebilir. Burada "Erzurum" değerine sahip farklı elemanlar da oluşturulabilir ancak bu elemanlara erişmek için kullanılan *anahtar* farklı bir değere sahip olmalıdır. Yani bir *HashTable* içerisinde bir tane 25 anahtarlı "Erzurum" değerine sahip eleman olabilir ancak farklı anahtarlara sahip birden fazla "Erzurum" değerinde eleman eklenebilir. Buna örnek olarak aynı isimli birçok kişi vardır ancak bu kişilerin tamamının T.C. kimlik numaraları farklıdır.

Özet olarak *HashTable* sınıfında *anahtar değer* koleksiyon elemanlarının organize edilmesinde kritik öneme sahiptir. Aşağıda *sehirler* isimli *HashTable* sınıfının elemanlarına erişmek için kullanılabilecek kod örneği verilmiştir.

```
MessageBox.Show(sehirler[25].ToString());
```

Yukarıdaki kod örneğinde *sehirler* isimli *HashTable* sınıfının 25 anahtarlı elemanına erişim sağlanarak elemana ait değerin MessageBox'ta görüntülenmesi sağlanmıştır. Kod çalıştırıldığında Şekil 12.1'deki görüntü elde edilecektir.



Şekil 12.1. MessageBox ekran görüntüsü

HashTable sınıfı *ArrayList* sınıfındaki metotların birçoğunu kullanmaktadır. Ancak bu metotların *HashTable* sınıfı ile birlikte kullanım yöntemleri ve aldıkları parametreler açısından ayrıldıkları noktalar bulunmaktadır. Aşağıda *HashTable* sınıfına eleman ekleme *Add()* ve silmeye *Remove()* yönelik verilen kod örneği bulunmaktadır.

```
Hashtable takimlar = new Hashtable();
takimlar.Add(1903, "Beşiktaş");
takimlar.Add(1905, "Galatasaray");
takimlar.Add(1907, "Fenerbahçe");
takimlar.Add(1968, "Erzurumspor");
takimlar.Remove(1907);
```

Yukarıdaki kod örneğinde *takimlar* isimli *HashTable* sınıfına anahtar değerleri takımların kuruluş tarihleri olacak şekilde birkaç takım ismi *Add()* metodu kullanılarak eklenmiştir. Ekleme işleminin ardından *Remove()* metodu kullanılarak silinmesi istenen elemanın anahtar değeri parametre olarak belirtilmiş ve 1907 anahtar değerine sahip eleman *HashTable* sınıfından silinmiştir. Örnekten de görüldüğü gibi *HashTable* sınıfında *Remove()* metodu kullanılırken silinmesi istenen elemanın indeks numarası değil anahtar değeri parametre olarak kabul edilmektedir.

HashTable sınıfında kullanılan anahtar değerin benzersiz olması zorunludur. Yani bir anahtar değer sadece bir kere kullanılabilir.

HashTable sınıfında kullanılan anahtar değerin benzersiz olması zorunludur. Yani bir anahtar değer sadece bir kere kullanılabilir. Bir anahtar değerin HashTable sınıfı içerisinde daha önceden kullanılıp kullanılmadığını öğrenmek için *ContainsKey()* metodu kullanılır. Bu metot parametre olarak aranması istenen anahtar değeri koleksiyon içinde tarar ve sonuç olarak *bool* tipinde bir değer döndürür. Eğer parametre olarak aranan değer *HashTable* içerisinde mevcutsa *true*, bu değer daha önce kullanılmamışsa *false* değeri geri döndürülür. Bu metoda benzer şekilde *HashTable* içerisinde arama yapan ve bir elemana ait değerin bulunup bulunmadığını tarayan metot ise *ContainsValue()* metodudur. Bu metot da *ContainsKey()* metoduna benzer şekilde aranan değer koleksiyonda yer alıyorsa *true*, yoksa *false* değeri döndürür.

HashTable sınıfına yönelik bilinmesi gereken önemli noktalardan biri de koleksiyona eklenen elemanların sıralı bir şekilde tutulmamasıdır. Bu da *HashTable* sınıfının sıralı bir yapıyı desteklemediği ve içerdiği elemanların belli bir sıraya sahip olmadığı anlamına gelmektedir. *HashTable* sınıfına eklenen elemanlar *Hash* kodlarına göre dizilir ve bu da *HashTable* kullanılırken performansın artmasını sağlar.

Bireysel Etkinlik

- HashTable sınıfıyla beraber kullanılan diğer metot ve bu metotlara ait özellikleri araştırınız.
- Aile bireylerinizden oluşan HashTable sınıfının elemanlarının anahtar - değer çiftlerini alt alta MessageBoz'ta görüntüleyiniz (Anahtar değer olarak T.C. kimlik numarasını, eleman değeri olarak ad soyad kullanınız.).

SortedList Sınıfı

SortedList sınıfı, elemanlarının küçüklük/büyüklük değerine göre veya alfabetik olarak sıralanmasına olanak tanır.

SortedList sınıfı, elemanlarına erişimin hem *HashTable* sınıfında olduğu gibi *anahtar* aracılığıyla hem de *indeks numarası* kullanılarak yapıldığı bir koleksiyon türüdür. *System.Collections* isim uzayının altında yer alan *SortedList* sınıfı, elemanlarının *küçüklük/büyüklük değerine göre veya alfabetik olarak sıralanmasına* olanak tanır. Bir başka ifadeyle elemanlarını sıralı bir şekilde saklayabilen ve indeks numarası kullanılarak da erişilebilen *HashTable* olarak tanımlanabilir. *SortedList sınıfına eklenen elemanlar anahtar değerleri baz alınarak sıralanırlar.* Burada koleksiyona eklenme sırasının veya ArrayList'teki gibi elemanın değerinin büyüklüğünün bir önemi yoktur, *sıralama sadece anahtar değerine göre yapılır.*

SortedList sınıfı *HashTable* sınıfında kullanılan birçok metodu ortak kullanır ve içerdiği elemanlara anahtar değerinin yanı sıra indeks numarası kullanılarak da erişilebilir. Bu sınıfın *HashTable* sınıfından farklı olarak kullandığı bazı önemli metotlar Tablo 12.4'te verilmiştir.

Tablo12.4. SortedList sınıfına ait bazı metotlar

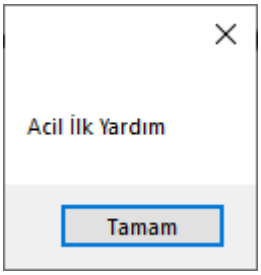
Metot Adı	Açıklama
GetByIndex	Parametre olarak verilen indeks numaralı koleksiyon elemanını geri döndürür.
GetKey	Parametre olarak verilen indeks numaralı koleksiyon anahtarını geri döndürür.
GetKeyList	SortedList içerisindeki anahtarları IList ara yüzü referansı ile geri döndürür.
GetValueList	SortedList içerisindeki değerleri IList ara yüzü referansı ile geri döndürür.
IndexOfKey	Parametre olarak verilen anahtar değerinin koleksiyonda kaçıncı indeks numarasına denk geldiği bilgisini geri döndürür.

SortedList sınıfına eleman ekleme ve eklenen bu elemanlara erişim yöntemine ait kod örneği aşağıda verilmiştir.

```
SortedList dersler = new SortedList();
dersler.Add("BTT", "Temel Bilgi Teknolojileri");
dersler.Add("AİY", "Acil İlk Yardım");
dersler.Add("YD", "Yabancı Dil");
dersler["BTT"] = "Bilişim Teknolojilerinin Temelleri";
MessageBox.Show(dersler.GetByIndex(0).ToString());
```

Yukarıdaki örnekte *dersler* isimli *SortedList* koleksiyonuna "BTT", "AİY" ve "YD" anahtarları ile ders isimleri eklenmiştir. Ardından "BTT" anahtarının değeri "Bilişim Teknolojilerinin Temelleri" şeklinde güncellenmiştir. Güncelleme işlemi yapılırken *köşeli parantezlerin [ ]* içerisine anahtar değerin yazılmasına dikkat edilmelidir. Bu işlem gerçekleştirildikten sonra "BTT" anahtarlı elemanın değeri "Bilişim Teknolojilerinin Temelleri" olacaktır. Son olarak da *MessageBox* kullanılarak dersler isimli *SortedList* koleksiyonundaki 0 indeks numaralı eleman ekranda görüntülenmiştir (Şekil 12.2).

Şekil 12.2. MessageBox ekran görüntüsü



Örnekte *SortedList* koleksiyonundaki elemanların anahtar değerleri *string* ifadelerdir. *SortedList* bu elemanları otomatik olarak sıralamaktadır. Örnekte *MessageBox* kullanılarak 0 indeks numaralı elemanın görüntülenmesi istenmiştir. *Normalde koleksiyona ilk olarak "BTT" anahtarlı eleman eklenmesine rağmen SortedList koleksiyonu otomatik olarak sıralama yaptığından ilk eleman yani 0 indeks numarasına sahip eleman "AİY" olmuştur ve bu şekilde ekranda görüntülenmiştir.* Dolayısıyla *GetByIndex()* metodu kullanıldığında 0 indeks numaralı eleman "AİY", 1 indeks numaralı eleman "BTT" ve 2 indeks numaralı eleman "YD" şeklinde olmuştur.

Bireysel Etkinlik

- Anahtar değeri cep telefonu markası ve elemanları cep telefonu modellerinden oluşan 10 elemanlı bir SortedList koleksiyonu oluşturunuz.
- Oluşturduğunuz koleksiyonun anahtar ve eleman değerlerini GetKey metodunu kullanarak bir Listbox içerisinde alt alta görüntüleyiniz.

Özet

- Koleksiyonlar, programlamada sıklıkla ihtiyaç duyulan yapılardan olup System.Collections isim uzayı altında yer alırlar. Bu yapılar içerisinde aynı türden veriler saklanabildiği gibi farklı türden veriler de saklanabilir. Bunun için Visual Studio.Net'in sunmuş olduğu çeşitli koleksiyon sınıfları bulunmaktadır. C++ dilinde konteyner, Java dilinde Collections Framework olarak adlandırılan bu sınıflar, C# dilinde ise koleksiyon olarak adlandırılmaktır.
- Koleksiyonlar; C++, Java gibi diğer programlama dillerindeki kullanım amacına bakıldığında temel olarak aynı ya da farklı türde birden fazla verinin veya nesnenin gruplanıp saklanmasına ve bunlara erişim sağlanmasına olanak tanıyan bir sınıf olarak tanımlanabilir. Bu tanıma göre dizilerle benzerlik gösterdiği görülebilir ancak koleksiyonların dizilerden ayrıldığı temel farklılıklar bulunmaktadır. Bunlardan en önemlisi dizilerde aynı türden veriler saklanabilir, koleksiyonlarda ise farklı türden verilerin saklanması mümkündür.
- *System.Collections* isim uzayının altında birçok *ara yüz* bulunmaktadır. Bu ara yüzler kullanılarak koleksiyon sınıfları için genel karakteristik tanımlaması yapılmaktadır. Örneğin *ArrayList* sınıfı koleksiyon sınıfları arasında sıklıkla kullanılır ve *IList* ara yüzünü uygulamaktadır. Bu yüzden bazı geliştiriciler tarafından *liste tipi koleksiyon* olarak adlandırılmaktadır.
- *System.Collections* isim uzayının altında yer alan *non generic* koleksiyon sınıflarının temel özelliği farklı türdeki verilerin *boxing* işlemi ile *object* türüne dönüştürülerek saklanabilmesidir. Bu başlık altında *Non Generic* koleksiyonlar arasında en sık kullanılan *ArrayList, Hashtable* ve *SortedList* sınıfları incelenecektir.
- *System.Collection* isim uzayı altında yer alan *ArrayList* sınıfı, içerisine aktarılan değerlerin *boxing* işlemi ile *object* türüne dönüştürüldüğü ve eleman sayısı belirlenmeden tanımlama yapıldığı koleksiyon türüdür. Dizilerde farklı türde eleman barındırılamaması ve dizi boyutunun dinamik bir şekilde düzenlenememesi sebebiyle geliştiriciler tarafından *ArrayList* sınıfı sıklıkla tercih edilir.
- *System.Collections* isim uzayı altında yer alan HashTable sınıfı, barındırdığı elamanlara ait değerlerin indeks numarası yerine benzersiz bir değer kullanılarak saklandığı ve bu şekilde erişim sağlandığı koleksiyon türüdür. Koleksiyon elemanlarına indeks numarası yerine farklı bir değer ile erişilmesi gerekebilir. Aynı şekilde koleksiyon elemanlarının benzersiz bir değere sahip olması ve koleksiyon içerisinde arama yapılırken bu benzersiz değerin kullanılması gerekebilir.
- HashTable sınıfında kullanılan anahtar değerin benzersiz olması zorunludur. Yani bir anahtar değer sadece bir kere kullanılabilir. Bir anahtar değerin HashTable sınıfı içerisinde daha önceden kullanılıp kullanılmadığını öğrenmek için *ContainsKey()* metodu kullanılır.
- *SortedList* sınıfı, elemanlarına erişimin hem HashTable sınıfında olduğu gibi *anahtar* aracılığıyla hem de *indeks numarası* kullanılarak yapıldığı bir koleksiyon türüdür. *System.Collections* isim uzayının altında yer alan *SortedList* sınıfı, elemanlarının *küçüklük/büyüklük değerine göre veya alfabetik olarak sıralanmasına* olanak tanır. Bir başka ifadeyle elemanlarını sıralı bir şekilde saklayabilen ve indeks numarası kullanılarak da erişilebilen HashTable olarak tanımlanabilir. *SortedList sınıfına eklenen elemanlar anahtar değerleri baz alınarak sıralanırlar.*

DEĞERLENDİRME SORULARI

1. Koleksiyonlarla ilgili aşağıdaki ifadelerden hangisi yanlıştır?
   a) System.Collections isim uzayı altında yer alır.
   b) Sadece aynı türde verileri saklayabilir.
   c) İçerdiği elemanlar kutulamaya tabi tutulabilir.
   d) Koleksiyon tanımlanırken new anahtar sözcüğü kullanılmalıdır.
   e) Koleksiyonlar performans avantajı sağlayabilir.

2. ArrayList koleksiyonuna eleman eklemek için aşağıdaki metotlardan hangisi kullanılır?
   a) Copy()
   b) Remove()
   c) IndexOf()
   d) Add()
   e) Clear()

3. ArrayList koleksiyonunda sıralama yapmak için aşağıdaki metotlardan hangisi kullanılır?
   a) Sort()
   b) Remove()
   c) Insert()
   d) Contains()
   e) Add()

4. Aşağıdakilerden hangisi Non Generic koleksiyon sınıfında yer alır?
   a) HashTable
   b) List
   c) Stack
   d) Queue
   e) LinkedList

5. HashTable koleksiyonunda bir anahtar değerini aramak için aşağıdaki metotlardan hangisi kullanılır?
   a) ContainsValue()
   b) ContainsKey()
   c) Add()
   d) GetEnumerator()
   e) Synchronized()

6. Bir anahtar değerine bağlı olarak anahtar/değer çiftlerini sıralı olarak saklayan koleksiyon aşağıdakilerden hangisidir?
   a) ArrayList
   b) Stack
   c) SortedList
   d) HashTable
   e) Stack

7. Hashtable koleksiyonu ile ilgili aşağıdakilerden hangisi yanlıştır?
   a) Anahtar-değer çifti mekanizmasını kullanır.
   b) System.Collections isim alanı altında bulunan bir sınıftır.
   c) Add metodu ile değer eklenebilir.
   d) Anahtar değer kullanılarak içinde arama yapılır.
   e) İndeks numarası ile HashTable nesnesi ögelerine erişim sağlanabilir.

8. HashTable koleksiyonunda bir elemanın değerini aramak için aşağıdaki metotlardan hangisi kullanılır?
   a) ContainsValue()
   b) ContainsKey()
   c) Add()
   d) GetEnumerator()
   e) Synchronized()

9. SortedList koleksiyonu ile ilgili aşağıdakilerden hangisi yanlıştır?
   a) Elemanlarını otomatik olarak sıralar.
   b) Elemanlarını anahtar değerine göre sıralar.
   c) GetByIndex metodundaki parametreye göre ilgili elemana erişilebilir.
   d) Elemanlarına sadece anahtar değeri kullanılarak erişilebilir.
   e) GetKey metodundaki parametreye göre ilgili anahtara erişilebilir.

10. İndeks numarası verilen elemanı ArrayList'ten çıkaran metot aşağıdakilerden hangisidir?
    a) Contains
    b) Remove
    c) RemoveAt
    d) Reverse
    e) Clear

Cevap Anahtarı
1.b,2.d,3.a,4.a,5.b,6.c,7.e,8.a,9.d,10.c

YARARLANILAN KAYNAKLAR

Albaharı, Joseph & ALBAHARI, Ben, (2012), C# 5.0 in a Nutshell Fifth Edition, O'Reilly Media, California.

Aktaş, Volkan, (2021), Her Yönüyle C# 9.0, KODLAB, İstanbul.

Clark, Dan, (2013), Beginning C# Object-Oriented Programming 2nd Edition, Apress, New York.

Grıffıths, Ian, (2013), Programming C# 5.0, O'Reilly Media, California.

Schıldt, Herbert, (2005), The Complete Reference C#, çev. Duygu Arbatlı Yağcı, Alfa Basım Yayım Dağıtım, İstanbul.

Skeet, Jon, (2014), C# in Depth 3rd Edition, Manning Publication Co, New York.