329-Windows Programlama

Bil. Müh. S. Kıvanç EKİCİ

İleri C# Programlama
Collections, Generics

- System.Collections
  - Nongeneric collections
- System.Collections.Generic
  - Generic collections

Array

```
static void Main(string[] args)
{
        // Make an array of string data.
        string[] strArray = {"First", "Second", "Third" };
        // Show number of items in array using Length property.
        Console.WriteLine("This array has {0} items.", strArray.Length);

        // Display contents using enumerator.
        foreach (string s in strArray)
        {
                Console.WriteLine("Array Entry: {0}", s);
        }

        // Reverse the array and print again.
        Array.Reverse(strArray);
        foreach (string s in strArray)
        {
                Console.WriteLine("Array Entry: {0}", s);
        }
        Console.ReadLine();
}
```

System.Collections

- ArrayList
- BitArray
- Hashtable
- Queue
- SortedList
- Stack

System.Collections Interface

- ICollection
  - size, enumeration, thread safety
- ICloneable
  - objenin kendi kopyasını çıkarması için
- IDictionary
  - key/value ikilileri ile nongeneric bir liste işlemi
- IEnumerable
  - IEnumerator cinsinden obje dönmesini sağlar
- IEnumerator
  - foreach tarzı iterasyonlara izin verir.
- IList
  - sıralı listelerde ekleme, çıkarma ve index elemanlarını sağlar

ArrayList

```
// You must import System.Collections to access the ArrayList.
static void Main(string[] args)
{
	ArrayList strArray = new ArrayList();
	strArray.AddRange(new string[] { "First", "Second", "Third" });
	// Show number of items in ArrayList.
	Console.WriteLine("This collection has {0} items.", strArray.Count);
	Console.WriteLine();
	// Add a new item and display current count.
	strArray.Add("Fourth!");
	Console.WriteLine("This collection has {0} items.", strArray.Count);
	// Display contents.
	foreach (string s in strArray)
	{
		Console.WriteLine("Entry: {0}", s);
	}
	Console.WriteLine();
}
```

Neden Soysallar (Generics)

- Generic olmayan listelerde veriler Object cinsinden tutulur. Ancak tutmak istediğimiz veriler sayı veya kendi yazdığımız class türünden olabilir. Bu nedenle verileri eklemek ve işlemek gereksiz performans kayıplarına sebep olur.
- Generic olmayan listeler **typesafe** değildir. Yani diziye sadece sayı eklenebilmesini istiyor olabiliriz. Ancak dizi ekleme fonksiyonu object olarak parametre aldığı için herşeyi listeye eklemek mümkündür. Bu nedenle listeye eklenmiş olan verilerin istediğimiz veri türünden olup olmadığını bilemeyiz.

```
static void SimpleBoxUnboxOperation()
{
    // Make a ValueType (int) variable.
    int myInt = 25;
    // Box the int into an object reference.
    object boxedInt = myInt;
    // Unbox the reference back into a corresponding int.
    int unboxedInt = (int)boxedInt;
}
```

typesafety

```
static void ArrayListOfRandomObjects()
{
        // The ArrayList can hold anything at all.
        ArrayList allMyObjects = new ArrayList();
        allMyObjects.Add(true);
        allMyObjects.Add(new OperatingSystem(PlatformID.MacOSX, new
        Version(10, 0)));
        allMyObjects.Add(66);
        allMyObjects.Add(3.14);
}
```

Soysallar (Generics)

```
static void UseGenericList()
{
    Console.WriteLine("***** Fun with Generics *****\n");
    // This List<> can hold only Person objects.

    List<Person> morePeople = new List<Person>();
    morePeople.Add(new Person ("Frank", "Black", 50));
    Console.WriteLine(morePeople[0]);

    // This List<> can hold only integers.
    List<int> moreInts = new List<int>();
    moreInts.Add(10);
    moreInts.Add(2);
    int sum = moreInts[0] + moreInts[1];
    // Compile-time error! Can't add Person object
    // to a list of ints!
    // moreInts.Add(new Person());
}
```

Örnek List<T>

```
// A partial listing of the List<T> class.
namespace System.Collections.Generic
{
    public class List<T> :
    IList<T>, ICollection<T>, IEnumerable<T>, IReadOnlyList<T>
    IList, ICollection, IEnumerable
    {...
    public void Add(T item);
    public ReadOnlyCollection<T> AsReadOnly();
    public int BinarySearch(T item);
    public bool Contains(T item);
    public void CopyTo(T[] array);
    public int FindIndex(System.Predicate<T> match);
    public T FindLast(System.Predicate<T> match);
    public bool Remove(T item);
    public int RemoveAll(System.Predicate<T> match);
    public T[] ToArray();
    public bool TrueForAll(System.Predicate<T> match);
```

System.Collections.Generic

- ICollection<T>
- IComparer<T>
- IDictionary<TKey, TValue>
- IEnumerable<T>
- IEnumerator<T>
- IList<T>
- ISet<T>

Okuyunuz : Sayfa 338 --> Table 9-4. Key Interfaces Supported by Classes of System.Collections.Generic

System.Collections.Generic

- Dictionary<TKey, TValue>
- LinkedList<T>
- List<T>
- Queue<T>
- SortedDictionary<TKey,TValue>
- SortedSet<T>
- Stack<T>

Okuyunuz : Sayfa 339 --> Table 9-5. Classes of System.Collections.Generic

Collection Initialization Syntax

```
// Init a standard array.
int[] myArrayOfInts = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };

// Init a generic List<> of ints.
List<int> myGenericList = new List<int> { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };

// Init an ArrayList with numerical data.
ArrayList myList = new ArrayList { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };


List<Point> myListOfPoints = new List<Point>
{
new Point { X = 2, Y = 2 },
new Point { X = 3, Y = 3 },
new Point(PointColor.BloodRed){ X = 4, Y = 4 }
};
```

Generic Methodlar

```
// This method will swap any two items.
// as specified by the type parameter <T>.
static void Swap<T>(ref T a, ref T b)
{
        Console.WriteLine("You sent the Swap() method a {0}",
        typeof(T));
        T temp;
        temp = a;
        a = b;
        b = temp;
}
```

typeof

```
static void DisplayBaseClass<T>()
{
        // BaseType is a method used in reflection,
        // which will be examined in Chapter 15
        Console.WriteLine("Base class of {0} is: {1}.",
        typeof(T), typeof(T).BaseType);
}
```

Generic Struct / Class

```
// A generic Point structure.
public struct Point<T>
{
	// Generic state date.
	private T xPos;
	private T yPos;
	// Generic constructor.
	public Point(T xVal, T yVal)
	{
		xPos = xVal;
		yPos = yVal;
	}
	// Generic properties.
	public T X
	{
		get { return xPos; }
		set { xPos = value; }
	}
	public T Y
	{
		get { return yPos; }
		set { yPos = value; }
	}
```

Generic Struct 2

```
        public override string ToString()
        {
                return string.Format("[{0}, {1}]", xPos, yPos);
        }
        // Reset fields to the default value of the
        // type parameter.
        public void ResetPoint()
        {
                xPos = default(T);
                yPos = default(T);
        }
}
```

default

```
// The "default" keyword is overloaded in C#.
// When used with generics, it represents the default
// value of a type parameter.
public void ResetPoint()
{
        X = default(T);
        Y = default(T);
}
```

- sayısal değerler için 0
- referans değerleri için null

Parametre Türlerini Kısıtlamak (Constraining Type Parameters)

```
// MyGenericClass derives from object, while
// contained items must have a default ctor.
public class MyGenericClass<T> where T : new()
{
    ...
}
```

- Okuyunuz : Sayfa 355 --> Table 9-8. Possible Constraints for Generic Type Parameters

```
// MyGenericClass derives from object, while
// contained items must be a class implementing IDrawable
// and must support a default ctor.
public class MyGenericClass<T> where T : class, IDrawable, new()
{
    ...
}
```

Ödev

- Aşağıdaki bölümler özetlenecek.
  - Chapter 8: Working with Interfaces.....281
- ICloneable ve IComparable arayüzlerini implemente eden sınıflar yazınız.