Other Non-Generic Collections in C#

BitArray

`BitArray` collection manages a compact array of bit values, which are represented as Booleans, where true indicates that the bit is on (1) and false indicates the bit is off (0).

Example:

```
using System;
using System.Collections;
public class SamplesBitArray  {
   public static void Main()  {
      // Creates and initializes several BitArrays.
      BitArray myBA1 = new BitArray( 5 );
      BitArray myBA2 = new BitArray( 5, false );
      byte[] myBytes = new byte[5] { 1, 2, 3, 4, 5 };
      BitArray myBA3 = new BitArray( myBytes );
      bool[] myBools = new bool[5] { true, false, true, true, false };
      BitArray myBA4 = new BitArray( myBools );
      // Displays the properties and values of the BitArrays.
      Console.WriteLine( "myBA1" );
      PrintValues( myBA1, 8 );
      Console.WriteLine( "myBA2" );
      PrintValues( myBA2, 8 );
      Console.WriteLine( "myBA3" );
      PrintValues( myBA3, 8 );
      Console.WriteLine( "myBA4" );
      PrintValues( myBA4, 8 );
   }
   public static void PrintValues( IEnumerable myList, int myWidth )  {
      Console.WriteLine( "   Count:    {0}", myList.Count );
      Console.WriteLine( "   Length:    {0}", myList.Length );
      Console.WriteLine( "   Values:" );
      int i = myWidth;
      foreach ( Object obj in myList ) {
         if ( i <= 0 )  {
            i = myWidth;
            Console.WriteLine();
         }
         i--;
         Console.Write( "{0,8}", obj );
      }
      Console.WriteLine();
   }
}
/*
```

```
This code produces the following output.

myBA1
   Count:    5
   Length:   5
   Values:
   False   False   False   False   False
myBA2
   Count:    5
   Length:   5
   Values:
   False   False   False   False   False
myBA3
   Count:    40
   Length:   40
   Values:
    True   False   False   False   False   False   False   False
   False    True   False   False   False   False   False   False
    True    True   False   False   False   False   False   False
   False   False    True   False   False   False   False   False
    True   False    True   False   False   False   False   False
myBA4
   Count:    5
   Length:   5
   Values:
    True   False    True    True   False
*/
```

SortedList

The `SortedList` in C# represents a collection of key/value pairs that are sorted by the keys and are accessible by key and by index.

Example:

```
using System;
using System.Collections;
public class SamplesSortedList
{
   public static void Main()
   {
      // Creates and initializes a new SortedList.
      SortedList mySL = new SortedList();
      mySL.Add("Third", "!");
      mySL.Add("Second", "World");
      mySL.Add("First", "Hello");
      // Displays the properties and values of the SortedList.
      Console.WriteLine( "mySL" );
      Console.WriteLine( "  Count:    {0}", mySL.Count );
      Console.WriteLine( "  Capacity: {0}", mySL.Capacity );
      Console.WriteLine( "  Keys and Values:" );
      PrintKeysAndValues( mySL );
   }
   public static void PrintKeysAndValues( SortedList myList )
   {
      Console.WriteLine( "\t-KEY-\t-VALUE-" );
      for ( int i = 0; i < myList.Count; i++ )
      {
        Console.WriteLine("\t{0}:\t{1}", myList.GetKey(i), myList.GetByIndex(i));
      }
      Console.WriteLine();
   }
}
/*
This code produces the following output.
mySL
  Count:    3
  Capacity: 16
  Keys and Values:
    -KEY-    -VALUE-
    First:    Hello
    Second:    World
    Third:    !
*/
```

Stack

Stack represents a simple last-in-first-out (LIFO) non-generic collection of objects.

Example:

```
using System;
using System.Collections;
namespace StackCollectionDemo
{
    class Program
    {
        static void Main(string[] args)
        {
            //Creating a stack collection
            Stack s = new Stack();
            //Adding item to the stack using the push method
            s.Push(10);
            s.Push("hello");
            s.Push(3.14f);
            s.Push(true);
            s.Push(67.8);
            s.Push('A');
            //Printing the stack items using foreach loop
            foreach (object obj in s)
            {
                Console.Write(obj + " ");
            }
            Console.WriteLine();
            //Removing annd returning an item from the stack
            //using the pop method
            Console.WriteLine(s.Pop());
            Console.WriteLine();
            //Printing item after removing the last added item
            foreach (object obj in s)
            {
                Console.Write(obj + " ");
            }
            Console.WriteLine();
            //Returning the last item from the stack without removing it
            //by using the peek method
            Console.WriteLine(s.Peek());
            Console.WriteLine();
            //Printing the items after using the Peek method
            foreach (object obj in s)
            {
                Console.Write(obj + " ");
            }
            Console.ReadKey();
        }
    }
}
```

Queue

`Queue` represents a first-in, first-out collection of objects.

Example:

```
using System;
using System.Collections;
namespace QueueCollectionDemo
{
    class Program
    {
        static void Main(string[] args)
        {
            //Creating a queue collection
            Queue q = new Queue();
            //Adding item to the queue using the Enqueue method
            q.Enqueue(10);
            q.Enqueue("hello");
            q.Enqueue(3.14f);
            q.Enqueue(true);
            q.Enqueue(67.8);
            q.Enqueue('A');
            //Printing the queue items using foreach loop
            foreach (object obj in q)
            {
                Console.Write(obj + " ");
            }
            Console.WriteLine();
            //Removing and returning an item from the queue
            //using the Dequeue method
            Console.WriteLine(q.Dequeue());
            Console.WriteLine();
            //Printing item after removing the first added item
            foreach (object obj in q)
            {
                Console.Write(obj + " ");
            }
            Console.WriteLine();
            //Returning the first item from the queue without removing it
            //by using the peek method
            Console.WriteLine(q.Peek());
            Console.WriteLine();
            //Printing the items after using the Peek method
            foreach (object obj in q)
            {
                Console.Write(obj + " ");
            }
            Console.ReadKey();
        }
    }
}
```