Collections : BitArray

- In simple terms BitArray is 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).
- BitArray is used when we want to deal with the bits of data.
- It is generally used when we need to store the bits but do not know the number of bits in advance.
- In BitArray we can access specific element like normal array means by using index.

Property of BitArray class

Count
Gets the number of elements contained in the BitArray.

IsReadOnly
Gets a value indicating whether the BitArray is read-only.

Item
Gets or sets the value of the bit at a specific position in the BitArray.

Length
Gets or sets the number of elements in the BitArray.

Methods of HashTable

public BitArray And(BitArray value);
Performs the bitwise AND operation on the elements in the current BitArray against the corresponding elements in the specified BitArray.

public bool Get(int index);
Gets the value of the bit at a specific position in the BitArray.

public BitArray Not();
Inverts all the bit values in the current BitArray, so that elements set to true are changed to false, and elements set to false are changed to true.

public BitArray Or(BitArray value);
Performs the bitwise OR operation on the elements in the current BitArray against the corresponding elements in the specified BitArray.

public void Set(int index, bool value);
Sets the bit at a specific position in the BitArray to the specified value.

public void SetAll(bool value);
Sets all bits in the BitArray to the specified value.

public BitArray Xor(BitArray value);
Performs the bitwise eXclusive OR operation on the elements in the current BitArray against the corresponding elements in the specified BitArray.

Application Program which demonstrate use of BitArray methods and properties.

```
using System;
using System.Collections;

 public class Marvellous
 {
  public static void Main(string[] args)
  {
        byte[] a = { 10 };
         byte[] b = { 64 };
         byte[] c = { 25 };

         BitArray ba1 = new BitArray(8);
         BitArray ba2 = new BitArray(8);
         BitArray ba3 = new BitArray(8);

         ba1 = new BitArray(a);
         ba2 = new BitArray(b);
         ba3 = new BitArray(c);

         Console.WriteLine("First Bit array whose value is 10");
         for (int i = 0; i < ba1.Count; i++)
        {
                 Console.Write("{0, -6} ", ba1[i]);
         }
         Console.WriteLine();

        Console.WriteLine("Second Bit array whose value is 64");
         for (int i = 0; i < ba2.Count; i++)
        {
                 Console.Write("{0, -6} ", ba2[i]);
         }
         Console.WriteLine();

        Console.WriteLine("Third Bit array whose value is 25");
         for (int i = 0; i < ba3.Count; i++)
        {
                 Console.Write("{0, -6} ", ba3[i]);
         }
         Console.WriteLine();

         Console.WriteLine("Bit array ba2: 13”);
         for (int i = 0; i < ba2.Count; i++)
        {
                 Console.Write("{0, -6} ", ba2[i]);
         }
         Console.WriteLine();
         BitArray ba4 = new BitArray(8);

        ba4 = ba1.And(ba2);
        Console.WriteLine("After AND operation");
        for (int i = 0; i < ba4.Count; i++)
        {
                 Console.Write("{0, -6} ", ba3[i]);
         }
         Console.WriteLine();

         ba4.Set(3,true);
        Console.WriteLine("After Set method");
         for (int i = 0; i < ba4.Count; i++)
        {
                 Console.Write("{0, -6} ", ba4[i]);
         }
        Console.WriteLine();

        ba4 = ba1.Or(ba2);
        Console.WriteLine("After OR operation");
        for (int i = 0; i < ba4.Count; i++)
        {
                 Console.Write("{0, -6} ", ba4[i]);
         }
         Console.WriteLine();

        ba4 = ba1.Xor(ba2);
         Console.WriteLine("After XOR operation");
         for (int i = 0; i < ba4.Count; i++)
        {
                 Console.Write("{0, -6} ", ba4[i]);
         }
  }
}
```