VB.NET - BITARRAY

The BitArray class 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.

It is used when you need to store the bits but do not know the number of bits in advance. You can access items from the BitArray collection by using an integer index, which starts from zero.

Properties and Methods of the BitArray Class

The following table lists some of the commonly used **properties** of the **BitArray** class:

Property	Description
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.

The following table lists some of the commonly used **methods** of the **BitArray** class:

S.N	Method Name & Purpose
1	**Public Function And *valueAsBitArray* As BitArray** Performs the bitwise AND operation on the elements in the current BitArray against the corresponding elements in the specified BitArray.
2	**Public Function Get *indexAsInteger* As Boolean** Gets the value of the bit at a specific position in the BitArray.
3	**Public Function Not As BitArray** 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.
4	**Public Function Or *valueAsBitArray* As BitArray** Performs the bitwise OR operation on the elements in the current BitArray against the corresponding elements in the specified BitArray.
5	**Public Sub Set *indexAsInteger, valueAsBoolean*** Sets the bit at a specific position in the BitArray to the specified value.
6	**Public Sub SetAll *valueAsBoolean***

Sets all bits in the BitArray to the specified value.

7

Public Function Xor *valueAsBitArray* As BitArray

Performs the bitwise eXclusive OR operation on the elements in the current BitArray against the corresponding elements in the specified BitArray.

Example:

The following example demonstrates the use of BitArray class:

```
Module collections
   Sub Main()
      'creating two  bit arrays of size 8
      Dim ba1 As BitArray = New BitArray(8)
      Dim ba2 As BitArray = New BitArray(8)
      Dim a() As Byte = {60}
      Dim b() As Byte = {13}
      'storing the values 60, and 13 into the bit arrays
      ba1 = New BitArray(a)
      ba2 = New BitArray(b)
      'content of ba1
      Console.WriteLine("Bit array ba1: 60")
      Dim i As Integer
      For i = 0 To ba1.Count
          Console.Write("{0 } ", ba1(i))
      Next i
      Console.WriteLine()
      'content of ba2
      Console.WriteLine("Bit array ba2: 13")
      For i = 0 To ba2.Count
          Console.Write("{0 } ", ba2(i))
      Next i
      Console.WriteLine()
      Dim ba3 As BitArray = New BitArray(8)
      ba3 = ba1.And(ba2)
      'content of ba3
      Console.WriteLine("Bit array ba3 after AND operation: 12")
      For i = 0 To ba3.Count
          Console.Write("{0 } ", ba3(i))
      Next i
      Console.WriteLine()
      ba3 = ba1.Or(ba2)
      'content of ba3
      Console.WriteLine("Bit array ba3 after OR operation: 61")
      For i = 0 To ba3.Count
          Console.Write("{0 } ", ba3(i))
      Next i
      Console.WriteLine()
      Console.ReadKey()
   End Sub
End Module
```

When the above code is compiled and executed, it produces the following result:

```
Bit array ba1: 60
False False True True True True False False
Bit array ba2: 13
True False True True False False False False
Bit array ba3 after AND operation: 12
False False True True False False False False
Bit array ba3 after OR operation: 61
True False True True False False False False
```