```
using System;
using System.Collections.Generic;
using System.Text;

namespace CECS_475_Lab1
{
    /// <summary>
    ///     IntegerSet Class object
    ///         - bool[] set: boolean array that hold true/false of that
    ///           integer is present in the integer set
    ///         + IntegerSet()
    ///         + IntegerSet(bool[])
    ///         + IntegerSet(int[])
    ///         + getSet() => bool[]
    ///         + ToString() => string
    ///         + Union(Integerset) => IntegerSet
    ///         + Intersect(IntegerSet) => IntegerSet
    ///         + InsertElement(int) => bool
    ///         + DeleteElement(int) => bool
    ///         + IsEqualTo(IntegerSet) => bool
    /// </summary>
    /// <remarks>
    ///     An IntegerSet object can be instantiated as an all false array, or by passing in
    ///     a boolean array, or by passing in an integer array
    /// </remarks>
    class IntegerSet
    {
        /// <summary>
        ///    Boolean set
        /// </summary>
        /// <value>
        ///    Each element represents an integer from range (0-100)
        /// </value>
        private bool[] set;

         /// <summary>
         ///    Default class constructor
         /// </summary>
        public IntegerSet()
        {
            this.set = new bool[101];
        }

         /// <summary>
         ///    Class constructor with passing parameter boolean array
         /// </summary>
         /// <param name="set">
         ///    Boolean array passed in to set object's 'set' value
         /// </param>
        public IntegerSet(bool[] set)
        {
            this.set = set;
        }

        /// <summary>
        ///     Class constructor with passing parameter integer array
        /// </summary>
        /// <param name="intSet">
        ///     Integer array passed in
        /// </param>
        public IntegerSet(int[] intSet)
        {
            set = new bool[101];
            for (int i = 0; i < intSet.Length; i++)
            {
                if (!(intSet[i] >= 0 && intSet[i] <= 100))
                {
                    Console.WriteLine("Integer " + intSet[i] + " is not within
                     range (0-100)");
                    continue;
                }
                this.set[intSet[i]] = true;
            }
        }

        /// <summary>
        ///     Object value accessor, get objects 'set' value
        /// </summary>
        /// <returns>
        ///     The objects 'set' boolean array
        /// </returns>
        public bool[] getSet()
        {
            return this.set;
        }

        /// <summary>
        ///     Method Override: ToString
        /// </summary>
        /// <returns>
        ///     String of the objects 'set' value
        /// </returns>
        public override string ToString()
        {
            string list = "";
            for (int i = 0; i < this.set.Length; i++)
            {
                if (this.set[i] == true)
                    list = list + i + " ";
            }
            // empty set
            if (list.Length < 1)
                return "[ --- ]";
            return "[ " + list + "]";
        }

        /// <summary>
        ///     Perform a union between this object and passed in object
        /// </summary>
        /// <param name="b">
        ///     IntegerSet object passed in to perform
        ///     union function with
        /// </param>
        /// <value>
        ///     unionArray: holds the result of the union function
        /// </value>
        /// <returns>
        ///     IntegerSet object with the result array
        ///     passed into it
        /// </returns>
        public IntegerSet Union(IntegerSet b)
        {
            if (this.set.Length != b.getSet().Length)
            {
                Console.WriteLine("Integer Set sizes are not the same.");
                return new IntegerSet();
            }
            bool[] unionArray = new bool[this.set.Length];
            for (int i = 0; i < this.set.Length; i++)
            {
                unionArray[i] = this.set[i] || b.getSet()[i];
            }
            return new IntegerSet(unionArray);
        }

        /// <summary>
        ///     Perform intersect between this object and passed
        ///     in object
        /// </summary>
        /// <param name="b">
        ///     IntegerSet object passed in to perform
        ///     union function with
        /// </param>
        /// <value>
        ///     intersectArray: holds the result of the intersection
        ///     function
        /// </value>
        /// <returns>
        ///     IntegerSet object with the result array
        ///     passed into it
        /// </returns>
        public IntegerSet Intersect(IntegerSet b)
        {
            if (this.set.Length != b.getSet().Length)
            {
                Console.WriteLine("Integer Set sizes are not the same.");
                return new IntegerSet();
            }
            bool[] intersectArray = new bool[this.set.Length];
            for (int i = 0; i < this.set.Length; i++)
            {
                intersectArray[i] = this.set[i] && b.getSet()[i];
            }
            return new IntegerSet(intersectArray);
        }

        /// <summary>
        ///     Insert an element to the objects boolean array 'set'
        /// </summary>
        /// <param name="k">
        ///     Integer to be added to the objects boolean array 'set'
        /// </param>
        /// <returns>
        ///     Boolean if element was within range
        /// </returns>
        public bool InsertElement(int k)
        {
            // check to see if k is within range of 0-100
            if (k >= 0 && k <= this.set.Length)
            {
                this.set[k] = true;
                return true;
            }
            Console.WriteLine("Can't insert " + k + ", Integer is out of range 
              (0-100)");
            return false;
        }

        /// <summary>
        ///     Delete an element to the object boolean array 'set'
        /// </summary>
        /// <param name="k">
        ///     Integer to be added to the objects boolean array 'set'
        /// </param>
        /// <returns>
        ///     Boolean
        ///     True: if element was within range
        ///     False: if the element is not within range
        /// </returns>
        public bool DeleteElement(int k)
        {
            // check to see if k is within range 0-100
            if (k >= 0 && k <= this.set.Length)
            {
                this.set[k] = false;
                return true;
            }
```

```
                Console.WriteLine("Can't delete " + k + ", Integer is out of range 
                  (0-100)");
                return false;
            }

            /// <summary>
            ///     Check if this object is equal to object b
            /// </summary>
            /// <param name="b">
            ///     IntegerObject passed in to see if it is equal
            /// </param>
            /// <returns>
            ///     Boolean
            ///     True: if they are equal
            ///     False: if they are unequal
            /// </returns>
            public bool IsEqualTo(IntegerSet b)
            {
                // if lengths are not the same, then cant fully compare
                if (this.set.Length != b.getSet().Length)
                    return false;
                bool areEqual = true;
                for (int i = 0; i < this.set.Length; i++)
                {
                    if (this.set[i] != b.getSet()[i])
                    {
                        areEqual = false;
                        break ;
                    }
                }
                return areEqual;
            }
        }
    }
```