```
1  using System;
2  using System.Collections.Generic;
3  using System.Text;
4
5  namespace CECS_475_Lab1
6  {
7      /// <summary>
8      ///     IntegerSet Class object
9      ///         - bool[] set: boolean array that hold true/false of that
10     ///           integer is present in the integer set
11     ///         + IntegerSet()
12     ///         + IntegerSet(bool[])
13     ///         + IntegerSet(int[])
14     ///         + getSet() => bool[]
15     ///         + ToString() => string
16     ///         + Union(Integerset) => IntegerSet
17     ///         + Intersect(IntegerSet) => IntegerSet
18     ///         + InsertElement(int) => bool
19     ///         + DeleteElement(int) => bool
20     ///         + IsEqualTo(IntegerSet) => bool
21     /// </summary>
22     /// <remarks>
23     ///     An IntegerSet object can be instantiated as an all false array, or by ↵
           passing in
24     ///     a boolean array, or by passing in an integer array
25     /// </remarks>
26     class IntegerSet
27     {
28         /// <summary>
29         ///    Boolean set
30         /// </summary>
31         /// <value>
32         ///    Each element represents an integer from range (0-100)
33         /// </value>
34         private bool[] set;
35
36          /// <summary>
37          ///    Default class constructor
38          /// </summary>
39         public IntegerSet()
40         {
41             this.set = new bool[101];
42         }
43
44          /// <summary>
45          ///    Class constructor with passing parameter boolean array
46          /// </summary>
47          /// <param name="set">
48          ///    Boolean array passed in to set object's 'set' value
49          /// </param>
50         public IntegerSet(bool[] set)
51         {
```

```
52              this.set = set;
53          }
54
55          /// <summary>
56          ///     Class constructor with passing parameter integer array
57          /// </summary>
58          /// <param name="intSet">
59          ///     Integer array passed in
60          /// </param>
61          public IntegerSet(int[] intSet)
62          {
63              set = new bool[101];
64              for (int i = 0; i < intSet.Length; i++)
65              {
66                  if (!(intSet[i] >= 0 && intSet[i] <= 100))
67                  {
68                      Console.WriteLine("Integer " + intSet[i] + " is not within
                         range (0-100)");
69                      continue;
70                  }
71                  this.set[intSet[i]] = true;
72              }
73          }
74
75          /// <summary>
76          ///     Object value accessor, get objects 'set' value
77          /// </summary>
78          /// <returns>
79          ///     The objects 'set' boolean array
80          /// </returns>
81          public bool[] getSet()
82          {
83              return this.set;
84          }
85
86          /// <summary>
87          ///     Method Override: ToString
88          /// </summary>
89          /// <returns>
90          ///     String of the objects 'set' value
91          /// </returns>
92          public override string ToString()
93          {
94              string list = "";
95              for (int i = 0; i < this.set.Length; i++)
96              {
97                  if (this.set[i] == true)
98                      list = list + i + " ";
99              }
100             // empty set
101             if (list.Length < 1)
102                 return "[ --- ]";
```

```
103             return "[ " + list + "]";
104         }
105
106         /// <summary>
107         ///     Perform a union between this object and passed in object
108         /// </summary>
109         /// <param name="b">
110         ///     IntegerSet object passed in to perform
111         ///     union function with
112         /// </param>
113         /// <value>
114         ///     unionArray: holds the result of the union function
115         /// </value>
116         /// <returns>
117         ///     IntegerSet object with the result array
118         ///     passed into it
119         /// </returns>
120         public IntegerSet Union(IntegerSet b)
121         {
122             if (this.set.Length != b.getSet().Length)
123             {
124                 Console.WriteLine("Integer Set sizes are not the same.");
125                 return new IntegerSet();
126             }
127             bool[] unionArray = new bool[this.set.Length];
128             for (int i = 0; i < this.set.Length; i++)
129             {
130                 unionArray[i] = this.set[i] || b.getSet()[i];
131             }
132             return new IntegerSet(unionArray);
133         }
134
135         /// <summary>
136         ///     Perform intersect between this object and passed
137         ///     in object
138         /// </summary>
139         /// <param name="b">
140         ///     IntegerSet object passed in to perform
141         ///     union function with
142         /// </param>
143         /// <value>
144         ///     intersectArray: holds the result of the intersection
145         ///     function
146         /// </value>
147         /// <returns>
148         ///     IntegerSet object with the result array
149         ///     passed into it
150         /// </returns>
151         public IntegerSet Intersect(IntegerSet b)
152         {
153             if (this.set.Length != b.getSet().Length)
154             {
```

```
155                 Console.WriteLine("Integer Set sizes are not the same.");
156                 return new IntegerSet();
157             }
158             bool[] intersectArray = new bool[this.set.Length];
159             for (int i = 0; i < this.set.Length; i++)
160             {
161                 intersectArray[i] = this.set[i] && b.getSet()[i];
162             }
163             return new IntegerSet(intersectArray);
164         }
165 
166         /// <summary>
167         ///     Insert an element to the objects boolean array 'set'
168         /// </summary>
169         /// <param name="k">
170         ///     Integer to be added to the objects boolean array 'set'
171         /// </param>
172         /// <returns>
173         ///     Boolean if element was within range
174         /// </returns>
175         public bool InsertElement(int k)
176         {
177             // check to see if k is within range of 0-100
178             if (k >= 0 && k <= this.set.Length)
179             {
180                 this.set[k] = true;
181                 return true;
182             }
183             Console.WriteLine("Can't insert " + k + ", Integer is out of range
                  (0-100)");
184             return false;
185         }
186 
187         /// <summary>
188         ///     Delete an element to the object boolean array 'set'
189         /// </summary>
190         /// <param name="k">
191         ///     Integer to be added to the objects boolean array 'set'
192         /// </param>
193         /// <returns>
194         ///     Boolean
195         ///     True: if element was within range
196         ///     False: if the element is not within range
197         /// </returns>
198         public bool DeleteElement(int k)
199         {
200             // check to see if k is within range 0-100
201             if (k >= 0 && k <= this.set.Length)
202             {
203                 this.set[k] = false;
204                 return true;
205             }
```

```
206             Console.WriteLine("Can't delete " + k + ", Integer is out of range
                  (0-100)");
207             return false;
208         }
209
210         /// <summary>
211         ///     Check if this object is equal to object b
212         /// </summary>
213         /// <param name="b">
214         ///     IntegerObject passed in to see if it is equal
215         /// </param>
216         /// <returns>
217         ///     Boolean
218         ///     True: if they are equal
219         ///     False: if they are unequal
220         /// </returns>
221         public bool IsEqualTo(IntegerSet b)
222         {
223             // if lengths are not the same, then cant fully compare
224             if (this.set.Length != b.getSet().Length)
225                 return false;
226             bool areEqual = true;
227             for (int i = 0; i < this.set.Length; i++)
228             {
229                 if (this.set[i] != b.getSet()[i])
230                 {
231                     areEqual = false;
232                     break ;
233                 }
234             }
235             return areEqual;
236         }
237     }
238 }
239
```