

```
7 public class SkiRunDataManagerXml : IDisposable
8 {
9     private List<SkiRun> _skiRuns;
10
11     public SkiRunDataManagerXml(List<SkiRun> skiRuns)
12     {
13         _skiRuns = skiRuns;
14     }
15
16     /// <summary>
17     /// method to return a list of ski run objects
18     /// </summary>
19     /// <returns>list of ski run objects</returns>
20     public List<SkiRun> GetAllSkiRuns()
21     {
22         return _skiRuns;
23     }
24
25     /// <summary>
26     /// method to return the index of a given ski run
27     /// <param name="skiRun"></param>
28     /// <returns>int ID</returns>
29     private int GetSkiRunByIndex(int ID)
30     {
31         int skiRunIndex = 0;
32
33         for (int index = 0; index < _skiRuns.Count(); index++)
34         {
35             if (_skiRuns[index].ID == ID)
36             {
37                 skiRunIndex = index;
38             }
39         }
40
41         return skiRunIndex;
42     }
43
44     /// <summary>
45     /// method to add a new ski run
46     /// </summary>
47     /// <param name="_skiRun"></param>
48     public async void InsertSkiRun(SkiRun skiRun)
```

```
49     {
50         _skiRuns.Add(skiRun);
51
52         await SkiRunsDataServiceXml.SaveObjectToXml(_skiRuns, "SkiRuns.xml");
53     }
54
55     /// <summary>
56     /// method to delete a ski run by ski run ID
57     /// </summary>
58     /// <param name="ID"></param>
59     public async void DeleteSkiRun(int ID)
60     {
61         _skiRuns.RemoveAt(GetSkiRunByIndex(ID));
62
63         await SkiRunsDataServiceXml.SaveObjectToXml(_skiRuns, "SkiRuns.xml");
64     }
65
66     /// <summary>
67     /// method to update an existing ski run
68     /// </summary>
69     /// <param name="skiRun">ski run object</param>
70     public async void UpdateSkiRun(SkiRun skiRun)
71     {
72         DeleteSkiRun(skiRun.ID);
73         InsertSkiRun(skiRun);
74
75         await SkiRunsDataServiceXml.SaveObjectToXml(_skiRuns, "SkiRuns.xml");
76     }
77
78     /// <summary>
79     /// method to return a ski run object given the ID
80     /// </summary>
81     /// <param name="ID">int ID</param>
82     /// <returns>ski run object</returns>
83     public SkiRun GetSkiRunByID(int ID)
84     {
85         SkiRun skiRun = null;
86
87         skiRun = _skiRuns[GetSkiRunByIndex(ID)];
88
89         return skiRun;
90     }
```

```
91 |  
92 |  
93 |  
94 |  
95 |  
96 |  
97 |  
98 |  
99 |  
100 |  
101 |  
102 |  
103 |  
104 |  
105 |  
106 |  
107 |  
108 |  
109 |  
110 |  
111 |  
112 |  
113 |  
114 |  
115 |  
116 |  
117 |  
118 |  
119 |  
120 |
```

```
    /// <summary>  
    /// method to query the data by the vertical of each ski run in feet  
    /// </summary>  
    /// <param name="minimumVertical">int minimum vertical</param>  
    /// <param name="maximumVertical">int maximum vertical</param>  
    /// <returns></returns>  
    public List<SkiRun> QueryByVertical(int minimumVertical, int maximumVertical)  
    {  
        List<SkiRun> matchingSkiRuns = new List<SkiRun>();  
  
        foreach (var skiRun in _skiRuns)  
        {  
            if ((skiRun.Vertical >= minimumVertical) & (skiRun.Vertical <= maximumVertical))  
            {  
                matchingSkiRuns.Add(skiRun);  
            }  
        }  
  
        return matchingSkiRuns;  
    }  
}
```

```
    /// <summary>  
    /// method to handle the IDisposable interface contract  
    /// </summary>  
    public void Dispose()  
    {  
        _skiRuns = null;  
    }  
}
```