```
    public class SkiRunDataManagerXml : IDisposable
    {
        private List<SkiRun> _skiRuns;

        public SkiRunDataManagerXml(List<SkiRun> skiRuns)
        {
            _skiRuns = skiRuns;
        }

        /// <summary>
        /// method to return a list of ski run objects
        /// </summary>
        /// <returns>list of ski run objects</returns>
        public List<SkiRun> GetAllSkiRuns()
        {
            return _skiRuns;
        }

        /// <summary>
        /// method to return the index of a given ski run
        /// <param name="skiRun"></param>
        /// <returns>int ID</returns>
        private int GetSkiRunByIndex(int ID)
        {
            int skiRunIndex = 0;

            for (int index = 0; index < _skiRuns.Count(); index++)
            {
                if (_skiRuns[index].ID == ID)
                {
                    skiRunIndex = index;
                }
            }

            return skiRunIndex;
        }

        /// <summary>
        /// method to add a new ski run
        /// </summary>
        /// <param name="_skiRun"></param>
        public async void InsertSkiRun(SkiRun skiRun)
        {
            _skiRuns.Add(skiRun);

            await SkiRunsDataServiceXml.SaveObjectToXml(_skiRuns, "SkiRuns.xml");
        }

        /// <summary>
        /// method to delete a ski run by ski run ID
        /// </summary>
        /// <param name="ID"></param>
        public async void DeleteSkiRun(int ID)
        {
            _skiRuns.RemoveAt(GetSkiRunByIndex(ID));

            await SkiRunsDataServiceXml.SaveObjectToXml(_skiRuns, "SkiRuns.xml");
        }

        /// <summary>
        /// method to update an existing ski run
        /// </summary>
        /// <param name="skiRun">ski run object</param>
        public async void UpdateSkiRun(SkiRun skiRun)
        {
            DeleteSkiRun(skiRun.ID);
            InsertSkiRun(skiRun);

            await SkiRunsDataServiceXml.SaveObjectToXml(_skiRuns, "SkiRuns.xml");
        }

        /// <summary>
        /// method to return a ski run object given the ID
        /// </summary>
        /// <param name="ID">int ID</param>
        /// <returns>ski run object</returns>
        public SkiRun GetSkiRunByID(int ID)
        {
            SkiRun skiRun = null;

            skiRun = _skiRuns[GetSkiRunByIndex(ID)];

            return skiRun;
        }
```

```csharp

        /// <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;
        }
    }
```