```
/*-----------------------------------------------------------------------------------------------
 * Author:      Dan Cassidy and Dr. Raman Adaikkalavan
 * Date:        2015-06-17
 * Assignment:  cView-P3
 * Source File: Park.cs
 * Language:    C#
 * Course:      CSCI-C 490, C# Programming, MoWe 08:00
 * Purpose:     Contains the Park class, derived from the Item abstract class, and supporting
 *              methods.
-----------------------------------------------------------------------------------------------*/

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Ph3
{
    public class Park : Item
    {
        /*-------------------------------------------------------------------------------------------
         * Type:    Helper Constants
        -------------------------------------------------------------------------------------------*/
        public new const FieldMenuHelper FieldMin = FieldMenuHelper.FeatureBaseball;
        public new const FieldMenuHelper FieldMax = FieldMenuHelper.BackPark;
        public const int FieldOffset = 8;

        /*-------------------------------------------------------------------------------------------
         * Type:    Private Fields
        -------------------------------------------------------------------------------------------*/
        private string itemType = "park";
        private int featureBaseball;
        private float featureBasketball;
        private float featureGolf;
        private int featureLargeMPField;
        private int featureTennis;
        private int featureVolleyball;

        /*-------------------------------------------------------------------------------------------
         * Type:    Constructor
         * Purpose: Basic no-parameter constructor.
         * Input:   Nothing.
        -------------------------------------------------------------------------------------------*/
        public Park()
        {
            // Nothing else to do.
        }

        /*-------------------------------------------------------------------------------------------
         * Type:    Constructor
         * Purpose: Copy constructor.
         * Input:   Park fromItem, reference to the other Park from which fields should be copied.
        -------------------------------------------------------------------------------------------*/
        public Park(Park fromItem)
            : base(fromItem)
        {
            itemType = fromItem.itemType;

            FeatureBaseball = fromItem.FeatureBaseball;
            FeatureBasketball = fromItem.FeatureBasketball;
            FeatureGolf = fromItem.FeatureGolf;
            FeatureLargeMPField = fromItem.FeatureLargeMPField;
            FeatureTennis = fromItem.FeatureTennis;
            FeatureVolleyball = fromItem.FeatureVolleyball;
        }
```

```
67
68          /*---------------------------------------------------------------------------------------
69           * Type:    Constructor
70           * Purpose: Constructor that will fill all the properties except ItemID and ItemType.
71           * Input:   string name, contains the desired Name for the object.
72           * Input:   string type, contains the desired Type for the object.
73           * Input:   string streetAddress, contains the desired StreetAddress for the object.
74           * Input:   string city, contains the desired City for the object.
75           * Input:   string state, contains the desired State for the object.
76           * Input:   string zip, contains the desired Zip for the object.
77           * Input:   string latitude, contains the desired Latitude for the object.
78           * Input:   string longitude, contains the desired Longitude for the object.
79           * Input:   string phone, contains the desired Phone for the object.
80           * Input:   int featureBaseball, contains the desired FeatureBaseball for the object.
81           * Input:   float featureBasketball, contains the desired FeatureBasketball for the object.
82           * Input:   float featureGolf, contains the desired FeatureGolf for the object.
83           * Input:   int featureLargeMPField, contains the desired FeatureLargeMPField for the
84           *          object.
85           * Input:   int featureTennis, contains the desired FeatureTennis for the object.
86           * Input:   int featureVolleyball, contains the desired FeatureVolleyball for the object.
87           * Output:  Nothing.
88          ---------------------------------------------------------------------------------------*/
89          public Park(string name, string type, string streetAddress, string city, string state,
90              string zip, string latitude, string longitude, string phone, int featureBaseball,
91              float featureBasketball, float featureGolf, int featureLargeMPField,
92              int featureTennis, int featureVolleyball)
93              : base(name, type, streetAddress, city, state, zip, latitude, longitude, phone)
94          {
95              FeatureBaseball = featureBaseball;
96              FeatureBasketball = featureBasketball;
97              FeatureGolf = featureGolf;
98              FeatureLargeMPField = featureLargeMPField;
99              FeatureTennis = featureTennis;
100             FeatureVolleyball = featureVolleyball;
101         }
102
103         /*---------------------------------------------------------------------------------------
104          * Type:    Auto-implemented Properties
105         ---------------------------------------------------------------------------------------*/
106         public override int ItemID { get; set; }            // Item ID
107
108         public override string Name { get; set; }           // Park Name
109         public override string Type { get; set; }           // Park Type
110         public override string StreetAddress { get; set; }  // Street Address
111         public override string City { get; set; }           // City
112         public override string State { get; set; }          // State
113         public override string Zip { get; set; }            // Zip Code
114         public override string Latitude { get; set; }       // Latitude
115         public override string Longitude { get; set; }      // Longitude
116         public override string Phone { get; set; }          // Phone Number
117
118         /*---------------------------------------------------------------------------------------
119          * Name:    ItemType
120          * Type:    Property
121          * Purpose: Provides access to the itemType field.
122         ---------------------------------------------------------------------------------------*/
123         public override string ItemType
124         {
125             get
126             {
127                 return itemType;
128             }
129             set
130             {
131                 // Do nothing.
132             }
```

```
133         }
134
135         /*---------------------------------------------------------------------------------------
136          * Name:    FeatureBaseball
137          * Type:    Property
138          * Purpose: Provides access to the featureBaseball field, and validation for the same.
139         ---------------------------------------------------------------------------------------*/
140         public int FeatureBaseball
141         {
142             get
143             {
144                 return featureBaseball;
145             }
146             set
147             {
148                 if (value >= 0)
149                     featureBaseball = value;
150             }
151         }
152
153         /*---------------------------------------------------------------------------------------
154          * Name:    FeatureBasketball
155          * Type:    Property
156          * Purpose: Provides access to the featureBasketball field, and validation for the same.
157         ---------------------------------------------------------------------------------------*/
158         public float FeatureBasketball
159         {
160             get
161             {
162                 return featureBasketball;
163             }
164             set
165             {
166                 if (value >= 0 && value % 0.5 == 0)
167                     featureBasketball = value;
168             }
169         }
170
171         /*---------------------------------------------------------------------------------------
172          * Name:    FeatureGolf
173          * Type:    Property
174          * Purpose: Provides access to the featureGolf field, and validation for the same.
175         ---------------------------------------------------------------------------------------*/
176         public float FeatureGolf
177         {
178             get
179             {
180                 return featureGolf;
181             }
182             set
183             {
184                 if (value >= 0 && value % 0.5 == 0)
185                     featureGolf = value;
186             }
187         }
188
189         /*---------------------------------------------------------------------------------------
190          * Name:    FeatureLargeMPField
191          * Type:    Property
192          * Purpose: Provides access to the featureLargeMPField field, and validation for the same.
193         ---------------------------------------------------------------------------------------*/
194         public int FeatureLargeMPField
195         {
196             get
197             {
198                 return featureLargeMPField;
```

```
199             }
200             set
201             {
202                 if (value >= 0)
203                     featureLargeMPField = value;
204             }
205         }
206 
207         /*-------------------------------------------------------------------------------------------
208          * Name:    FeatureTennis
209          * Type:    Property
210          * Purpose: Provides access to the featureTennis field, and validation for the same.
211         -------------------------------------------------------------------------------------------*/
212         public int FeatureTennis
213         {
214             get
215             {
216                 return featureTennis;
217             }
218             set
219             {
220                 if (value >= 0)
221                     featureTennis = value;
222             }
223         }
224 
225         /*-------------------------------------------------------------------------------------------
226          * Name:    FeatureVolleyball
227          * Type:    Property
228          * Purpose: Provides access to the featureVolleyball field, and validation for the same.
229         -------------------------------------------------------------------------------------------*/
230         public int FeatureVolleyball
231         {
232             get
233             {
234                 return featureVolleyball;
235             }
236             set
237             {
238                 if (value >= 0)
239                     featureVolleyball = value;
240             }
241         }
242 
243         /*-------------------------------------------------------------------------------------------
244          * Name:    this[]
245          * Type:    Indexer
246          * Purpose: Provides easy access to the properties of the class.
247          * Input:   FieldMenuHelper fiendNum, represents the desired property.
248          * Output:  object, contains whichever property was desired, or 0 if the property was not
249          *          found.
250         -------------------------------------------------------------------------------------------*/
251         public override object this[FieldMenuHelper fieldNum]
252         {
253             get
254             {
255                 switch (fieldNum)
256                 {
257                     case FieldMenuHelper.FeatureBaseball:
258                         return FeatureBaseball;
259                     case FieldMenuHelper.FeatureBasketball:
260                         return FeatureBasketball;
261                     case FieldMenuHelper.FeatureGolf:
262                         return FeatureGolf;
263                     case FieldMenuHelper.FeatureLargeMPField:
264                         return FeatureLargeMPField;
```

```
265                     case FieldMenuHelper.FeatureTennis:
266                         return FeatureTennis;
267                     case FieldMenuHelper.FeatureVolleyball:
268                         return FeatureVolleyball;
269                     default:
270                         return base[fieldNum];
271                 }
272             }
273         }
274 
275         /*-------------------------------------------------------------------------------------------
276          * Name:    ToStringCSV
277          * Type:    Method
278          * Purpose: Serializes the data contained in the object into a comma-separated value string.
279          * Input:   Nothing.
280          * Output:  string, representing the data of this object as serialized to a CSV string.
281         -------------------------------------------------------------------------------------------*/
282         public override string ToStringCSV()
283         {
284             char separator = ',';
285             return base.ToStringCSV() + separator + FeatureBaseball + separator +
286                 FeatureBasketball + separator + FeatureGolf + separator + FeatureLargeMPField +
287                 separator + FeatureTennis + separator + FeatureVolleyball;
288         }
289 
290         /*-------------------------------------------------------------------------------------------
291          * Name:    ToString
292          * Type:    Method
293          * Purpose: Override of ToString() method. Formats the data contained in this object so it
294          *          looks pretty.
295          * Input:   Nothing.
296          * Output:  string, containing serialized object data.
297         -------------------------------------------------------------------------------------------*/
298         public override string ToString()
299         {
300             // Returns a the base a string formatted as follows:
301             //  Item ID (Item Type): <ItemID> (<ItemType>)
302             //     Park Name (Type): <Name> (<Type>)
303             //              Address: <StreetAddress>, <City>, <State> <Zip>
304             //      GPS Coordinates: (<Latitude>, <Longitude>)
305             //         Phone Number: <Phone>
306             //    Baseball Diamonds: <FeatureBaseball>
307             //    Basketball Courts: <FeatureBasketball>
308             //         Golf Courses: <FeatureGolf>
309             //      Large MP Fields: <FeatureLargeMPField>
310             //        Tennis Courts: <FeatureTennis>
311             //    Volleyball Courts: <FeatureVolleyball>
312             return string.Format(
313                 " Item ID (Item Type): {0} ({1})\n" +
314                 "    Park Name (Type): {2} ({3})\n" +
315                 "             Address: {4}, {5}, {6} {7}\n" +
316                 "     GPS Coordinates: ({8}, {9})\n" +
317                 "        Phone Number: {10}\n" +
318                 "   Baseball Diamonds: {11}\n" +
319                 "   Basketball Courts: {12}\n" +
320                 "        Golf Courses: {13}\n" +
321                 "     Large MP Fields: {14}\n" +
322                 "       Tennis Courts: {15}\n" +
323                 "   Volleyball Courts: {16}",
324                 ItemID, ItemType,
325                 Name, Type,
326                 StreetAddress, City, State, Zip,
327                 Latitude, Longitude,
328                 Phone,
329                 FeatureBaseball,
330                 FeatureBasketball,
```

```
331                 FeatureGolf,
332                 FeatureLargeMPField,
333                 FeatureTennis,
334                 FeatureVolleyball);
335         }
336     }
337 }
338
```