```
/*-----------------------------------------------------------------------------------------------
 * Author:      Dan Cassidy and Dr. Raman Adaikkalavan
 * Date:        2015-06-17
 * Assignment:  cView-P3
 * Source File: Item.cs
 * Language:    C#
 * Course:      CSCI-C 490, C# Programming, MoWe 08:00
 * Purpose:     Provides the base abstract class for data items along with some supporting methods.
-----------------------------------------------------------------------------------------------*/

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

namespace Ph3
{
    public abstract class Item
    {
        /*-------------------------------------------------------------------------------------------
         * Type:    Helper Constants
        -------------------------------------------------------------------------------------------*/
        public const FieldMenuHelper FieldCommonMin = FieldMenuHelper.Name;
        public const FieldMenuHelper FieldCommonMax = FieldMenuHelper.Phone;
        public const FieldMenuHelper FieldMin = FieldMenuHelper.Name;
        public const FieldMenuHelper FieldMax = FieldMenuHelper.BackPark;

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

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

            Name = fromItem.Name;
            Type = fromItem.Type;
            StreetAddress = fromItem.StreetAddress;
            City = fromItem.City;
            State = fromItem.State;
            Zip = fromItem.Zip;
            Latitude = fromItem.Latitude;
            Longitude = fromItem.Longitude;
            Phone = fromItem.Phone;
        }

        /*-------------------------------------------------------------------------------------------
         * Type:    Constructor
         * Purpose: Constructor that will fill all the properties except ItemID and ItemType.
         * Input:   string name, contains the desired Name for the object.
         * Input:   string type, contains the desired Type for the object.
         * Input:   string streetAddress, contains the desired StreetAddress for the object.
         * Input:   string city, contains the desired City for the object.
```

```
67          * Input:   string state, contains the desired State for the object.
68          * Input:   string zip, contains the desired Zip for the object.
69          * Input:   string latitude, contains the desired Latitude for the object.
70          * Input:   string longitude, contains the desired Longitude for the object.
71          * Input:   string phone, contains the desired Phone for the object.
72         ----------------------------------------------------------------------------------------*/
73         public Item(string name, string type, string streetAddress, string city, string state,
74             string zip, string latitude, string longitude, string phone)
75         {
76             Name = name;
77             Type = type;
78             StreetAddress = streetAddress;
79             City = city;
80             State = state;
81             Zip = zip;
82             Latitude = latitude;
83             Longitude = longitude;
84             Phone = phone;
85         }
86
87         /*----------------------------------------------------------------------------------------
88          * Name:    FieldMenuHelper
89          * Type:    Enum
90          * Purpose: Represents the fields in use in this class, with additions for its derived
91          *          classes.
92         ----------------------------------------------------------------------------------------*/
93         public enum FieldMenuHelper
94         {
95             // Common Fields
96             Name = 1,
97             Type,
98             StreetAddress,
99             City,
100            State,
101            Zip,
102            Latitude,
103            Longitude,
104            Phone,
105            Back,
106
107            // Business Fields
108            LicenseFiscalYear,
109            LicenseNumber,
110            LicenseIssueDate,
111            LicenseExpirDate,
112            LicenseStatus,
113            CouncilDistrict,
114            BackBusiness,
115
116            // Park Fields
117            FeatureBaseball,
118            FeatureBasketball,
119            FeatureGolf,
120            FeatureLargeMPField,
121            FeatureTennis,
122            FeatureVolleyball,
123            BackPark
124        }
125
126        /*----------------------------------------------------------------------------------------
127         * BEGIN UNTOUCHABLE CODE -->
128        ----------------------------------------------------------------------------------------*/
129        // Create an ID for each item. So if you have 10 parks and 5 businesses, ID will be 1 to 15.
130        public abstract int ItemID { get; set; }
131
132        // Value will be "business", "park", or "publicfacility".
```

```
133         public abstract string ItemType { get; set; }
134
135         // Populate from CSV
136         public abstract string Name { get; set; }
137         public abstract string Type { get; set; }
138         public abstract string StreetAddress { get; set; }
139         public abstract string City { get; set; }
140         public abstract string State { get; set; }
141         public abstract string Zip { get; set; }
142         public abstract string Latitude { get; set; }
143         public abstract string Longitude { get; set; }
144         public abstract string Phone { get; set; }
145         /*-------------------------------------------------------------------------------------------
146          * <-- END UNTOUCHABLE CODE
147         -------------------------------------------------------------------------------------------*/
148
149         /*-------------------------------------------------------------------------------------------
150          * Name:    this[]
151          * Type:    Indexer
152          * Purpose: Provides easy access to the properties of the class. Need to change the indexer
153          *          name because its default is "Item" and the compiler throws a fit because the
154          *          class is already named that.
155          * Input:   FieldMenuHelper fiendNum, represents the desired property.
156          * Output:  object, contains whichever property was desired, or 0 if the property was not
157          *          found.
158         -------------------------------------------------------------------------------------------*/
159         [IndexerName("Index")]
160         public virtual object this[FieldMenuHelper fieldNum]
161         {
162             get
163             {
164                 switch (fieldNum)
165                 {
166                     case FieldMenuHelper.Name:
167                         return Name;
168                     case FieldMenuHelper.Type:
169                         return Type;
170                     case FieldMenuHelper.StreetAddress:
171                         return StreetAddress;
172                     case FieldMenuHelper.City:
173                         return City;
174                     case FieldMenuHelper.State:
175                         return State;
176                     case FieldMenuHelper.Zip:
177                         return Zip;
178                     case FieldMenuHelper.Latitude:
179                         return Latitude;
180                     case FieldMenuHelper.Longitude:
181                         return Longitude;
182                     case FieldMenuHelper.Phone:
183                         return Phone;
184                     default:
185                         return 0;
186                 }
187             }
188         }
189
190         /*-------------------------------------------------------------------------------------------
191          * Name:    ToStringCSV
192          * Type:    Method
193          * Purpose: Serializes the data contained in the object into a comma-separated value string.
194          * Input:   Nothing.
195          * Output:  string, representing the data of this object as serialized to a CSV string.
196         -------------------------------------------------------------------------------------------*/
197         public virtual string ToStringCSV()
198         {
```

```
199             char separator = ',';
200             return Name + separator + Type + separator + StreetAddress + separator + City +
201                 separator + State +separator + Zip + separator + Latitude + separator + Longitude +
202                 separator + Phone;
203         }
204 
205         /*-----------------------------------------------------------------------------------------
206          * Name:    ToStringSimple
207          * Type:    Method
208          * Purpose: Formats the data contained in the object into a simplified string containing
209          *          only the ItemID, ItemType, and Name properties.
210          * Input:   Nothing.
211          * Output:  string, representing a simplified view of this object.
212         -----------------------------------------------------------------------------------------*/
213         public virtual string ToStringSimple()
214         {
215             // Returns a string formatted as follows:
216             //   Item ID: <ItemID>
217             // Item Type: <ItemType>
218             //      Name: <Name>
219             return string.Format(
220                 "  Item ID: {0}\n" +
221                 "Item Type: {1}\n" +
222                 "     Name: {2}",
223                 ItemID,
224                 ItemType,
225                 Name);
226         }
227     }
228 }
229 
```