```
1  using System;
2  using System.Collections.Generic;
3  using System.IO;
4  using System.Data;
5  using System.Text;
6
7  /*
8   * Title:   Lists
9   * Author:  Paul McKillop
10  * Date:    March 2020
11  * Purpose: Create lists and structures for manipulation in other classes
12  */
13
14 namespace SignatureGeneratorV
15 {
16     public class Lists
17     {
18         /// <summary>
19         /// All character data
20         /// </summary>
21         /// <returns></returns>
22         public static List<Character> Characters()
23         {
24             string path = (@"E:\ascii.txt");
25
26             List<Character> characters = new List<Character>();
27
28             DataTable characterData = new DataTable();
29
30             characterData = UtilityCharacterDb.GetCharacterData(path);
31
32             foreach (DataRow row in characterData.Rows)
33             {
34                 var currentCharacter = new Character
35                 {
36                     Code = row.Field<string>(0),
37                     Score = row.Field<string>(1)
38                 };
39
40                 characters.Add(currentCharacter);
41             }
42             //-- Return the list of codes
43             return characters;
44         }
45
46         #region ValidCharacters as Characters
47         /// <summary>
48         /// ValidCharacters as list<Character></Character>
49         /// </summary>
50         /// <returns></returns>
51         public static List<Character> ValidCharacters()
52         {
53             string path = (@"E:\ascii.txt");
54
55             List<Character> characters = new List<Character>();
56
```

```
57              DataTable characterData = new DataTable();
58
59              characterData = UtilityCharacterDb.GetCharacterData(path);
60
61              foreach (DataRow row in characterData.Rows)
62              {
63                  var currentCharacter = new Character
64                  {
65                      Code = row.Field<string>(0),
66                      Score = row.Field<string>(1)
67                  };
68
69                  if (currentCharacter.Score != "9")
70                  {
71                      characters.Add(currentCharacter);
72                  }
73
74              }
75              //-- Return the list of codes
76              return characters;
77          }
78          #endregion
79
80          #region ValidCharacterCodes (Codes only)
81          /// <summary>
82          /// Valid codes as List<string></string>
83          /// </summary>
84          /// <returns></returns>
85          public static List<string> ValidCharacterCodes()
86          {
87              string path = (@"E:\ascii.txt");
88
89
90              List<string> codes = new List<string>();
91              //-- Implement using statement to provide memory management
92              using (StreamReader reader = new StreamReader(path))
93              {
94                  //-- Loop through all and harvest first column into the list
95                  while (true)
96                  {
97                      //-- read line
98                      string line = reader.ReadLine();
99                      //-- Drop if no line data
100                     if (line == null)
101                     {
102                         break;
103                     }
104
105                     //-- split fields with comma separator
106                     string[] fields = line.Split(',');
107
108                     //-- Initialise a Character oblject to hold data
109                     //-- Could use a simple string but demo of OOP process
110                     Character character = new Character();
111
112                     if (fields[1] != "9")
```

```
                    {
                        //-- Read code field
                        character.Code = fields[0];
                        //-- Add the code to the list
                        codes.Add(character.Code);
                    }
                }
            }

            //-- Return the list of codes
            return codes;
        }
        #endregion

        /// <summary>
        /// Just Character codes
        /// </summary>
        /// <returns></returns>
        public static List<string> CharacterCodes()
        {
            string path = (@"E:\ascii.txt");


            List<string> codes = new List<string>();
            //-- Implement using statement to provide memory management
            using (StreamReader reader = new StreamReader(path))
            {
                //-- Loop through all and harvest first column into the list
                while (true)
                {
                    //-- read line
                    string line = reader.ReadLine();
                    //-- Drop if no line data
                    if (line == null)
                    {
                        break;
                    }

                    //-- split fields with comma separator
                    string[] fields = line.Split(',');

                    //-- Initialise a Character oblject to hold data
                    //-- Could use a simple string but demo of OOP process
                    Character character = new Character();
                    //-- Read code field
                    character.Code = fields[0];
                    //-- Add the code to the list
                    codes.Add(character.Code);
                }
            }

            //-- Return the list of codes
            return codes;
        }


```

```
169         #region String is found in list
170
171         /// <summary>
172         /// Check if a string is already in a list
173         /// </summary>
174         /// <param name="listToSearch"></param>
175         /// <param name="stringToFind"></param>
176         /// <returns>Boolean</returns>
177         public static bool StringFound(List<string> listToSearch, string
              stringToFind)
178         {
179             //-- tracker variable
180             bool stringFound = false;
181             //-- Loop through all
182             foreach (string value in listToSearch)
183             {
184                 if (value == stringToFind)
185                 {
186                     stringFound = true;
187                     return stringFound;
188                 }
189             }
190
191             //-- return true or false: in list, or not
192             return stringFound;
193         }
194
195         #endregion
196     }
197 }
198
```