```
 1  namespace DelegadosResumen
 2  {
 3      public class Country
 4      {
 5          private List<City> _citiesList = new List<City>();
 6
 7          public void CreateCitiesList()
 8          {
 9              City c1 = new City("Alicante", 350000);
10              City c2 = new City("Valencia", 850000);
11              City c3 = new City("Madrid", 1250000);
12              City c4 = new City("Sevilla", 950000);
13              City c5 = new City("Barcelona", 1100000);
14              City c6 = new City("Bilbao", 870000);
15              City c7 = new City("Pontevedra", 250000);
16
17              _citiesList = new List<City>() { c1, c2, c3, c4, c5, c6,
                  c7 };
18          }
19
20          // Podemos implementar una funcion 'FILTER' que ordena nuestra
              coleccion de datos
21          // según una 'FUNCION DELEGADA' que introduce el usuario a
              traves de una 'LAMBDA'
22
23          // public List Filter(FUNCION LAMBDA)
24          // la funcion 'Filter' devuelve una lista segun los criterios
              de la 'FUNCION LAMBDA.
25
26
27          #region Delegate Filter
28          // CREAMOS UN CODIGO FUENTE QUE:
29          // FILTRA ->
30          // UNA COLECCION EXISTENTE MEDIANTE UNA FUNCION FILTER ->
31          // A TRAVES DE UNA FUNCION DELEGADA
32          // ======================================================
33
34          // A. Definicion de la FUNCION DELEGADA:
35          // La funcion NO RECIBE PARAMETROS y devuelve un BOOL.
36          public delegate bool DelegateFilterPlain();
37
38
39          // B. Definicion de la funcion que FILTRA:
40          // Si la funcion delegada devuelve 'TRUE' para un elemento, lo
              añade a la lista resultado.
41          public List<City> Filter(DelegateFilterPlain filter)
42          {
43              List<City> result = new List<City>();
44              for (int i = 0; i < _citiesList.Count; i++)
45              {
46                  if (filter())
47                  {
48                      result.Add(_citiesList[i]);
```

```
49                  }
50              }
51              return result;
52          }
53
54
55          // C. Definicion de los parametros de la FUNCION DELEGADA:
56          // Existen distintas sintaxis para la funcion filter.
57          public void FilterTest()
58          {
59              Country countryTest = new Country();
60              List<City> listResult = new List<City>();
61
62              // sintaxis 1
63              DelegateFilterPlain delegado = delegate () { return
                  true; };
64              DelegateFilterPlain filter = new DelegateFilterPlain
                  (delegado);
65              listResult = countryTest.Filter(filter);
66
67              // sintaxis 2
68              DelegateFilterPlain filter2 = new DelegateFilterPlain(() =>
                    { return true; });
69              listResult = countryTest.Filter(filter2);
70
71              // sintaxis 3
72              DelegateFilterPlain filter3 = new DelegateFilterPlain(() =>
                    true );
73              listResult = countryTest.Filter(filter3);
74          }
75
76
77          // A'. Definicion de la FUNCION DELEGADA:
78          // La funcion RECIBE UN INT y devuelve un BOOL.
79          public delegate bool DelegateFilterWithInt(int population);
80          public List<City> Filter(DelegateFilterWithInt filter)
81          {
82              List<City> result = new List<City>();
83              for (int i = 0; i < _citiesList.Count; i++)
84              {
85                  if (filter(_citiesList[i].Population))
86                  {
87                      result.Add(_citiesList[i]);
88                  }
89              }
90              return result;
91          }
92
93
94
95
96
97
```

```
 98
 99          // A''. Definicion de la FUNCION DELEGADA:
100          // La funcion RECIBE UN STRING y devuelve un BOOL.
101          public delegate bool DelegateFilterWithString(string name);
102          public List<City> Filter(DelegateFilterWithString filter)
103          {
104              List<City> result = new List<City>();
105              for (int i = 0; i < _citiesList.Count; i++)
106              {
107                  if (filter(_citiesList[i].Name))
108                  {
109                      result.Add(_citiesList[i]);
110                  }
111              }
112              return result;
113          }
114          #endregion
115
116
117          #region Delegate Sort
118          // CREAMOS UN CODIGO FUENTE QUE:
119          // ORDENA ->
120          // UNA COLECCION EXISTENTE MEDIANTE UNA FUNCION SORT ->
121          // A TRAVES DE UNA FUNCION DELEGADA
122          // ======================================================
123
124          // A. Definicion de la FUNCION DELEGADA:
125          // La funcion recibe DOS ELEMENTOS DE LA COLECCION y devuelve   ↵
               un INT.
126          public delegate int DelegateSort(City city1, City city2);
127
128
129          // B. Definicion de la funcion que ORDENA:
130          // Si la funcion delegada devuelve '-1' se mantiene la          ↵
               posición, si devuelve '1' mueve el elemento a la derecha.
131          public List<City> Sort(DelegateSort filter)
132          {
133              List<City> result = new List<City>();
134              for (int i = 0; i < _citiesList.Count; i++)
135              {
136                  for(int j = i + 1; j < _citiesList.Count; j++)
137                  {
138                      if (filter(_citiesList[i], _citiesList[j]) > 0)
139                      {
140                          City aux;
141                          aux = _citiesList[i];
142                          _citiesList[i] = _citiesList[j];
143                          _citiesList[j] = aux;
144                      }
145                  }
146              }
147              return result;
148          }
```

```
149
150         // C. Definicion de los parametros de la FUNCION DELEGADA:
151         // Existen distintas sintaxis para la funcion filter.
152         public void SortTest()
153         {
154             Country countryTest = new Country();
155             List<City> listResult = new List<City>();
156
157             // sintaxis 1
158             DelegateSort delegado = delegate (City city1, City city2)
                  { return 0; };
159             DelegateSort filter = new DelegateSort(delegado);
160             listResult = countryTest.Sort(filter);
161
162             // sintaxis 2
163             DelegateSort filter2 = new DelegateSort((City city1, City
                  city2) => { return 0; });
164             listResult = countryTest.Sort(filter2);
165
166             // sintaxis 3
167             DelegateSort filter3 = new DelegateSort((City city1, City
                  city2) => 0);
168             listResult = countryTest.Sort(filter3);
169         }
170         #endregion
171
172
173         #region Delegate Visit
174         // CREAMOS UN CODIGO FUENTE QUE:
175         // VISITA ->
176         // TODA LA COLECCION EXISTENTE MEDIANTE UNA FUNCION DEFINIDA
              POR EL USUARIO ->
177         // A TRAVES DE UNA FUNCION DELEGADA
178         // ======================================================
179
180         // A. Definicion de la FUNCION DELEGADA:
181         // La funcion recibe UN ELEMENTO DE LA COLECCION y devuelve
              VOID.
182         public delegate void DelegateVisit(City visitor);
183
184
185         // B. Definicion de la funcion que VISITA:
186         // El usuario definirá la FUNCIÓN que se realiza sobre CADA UNO
              DE LOS ELEMENTOS
187         // de la colección.
188         public void Visit(DelegateVisit visitor)
189         {
190             for(int i = 0; i < _citiesList.Count; i++)
191             {
192                 visitor(_citiesList[i]);
193             }
194         }
195
```

```
196         // C. Definicion de los parametros de la FUNCION DELEGADA:
197         // Existen distintas sintaxis para la funcion filter.
198         public void VisitTest()
199         {
200             Country countryTest = new Country();
201             List<City> listResult = new List<City>();
202 
203             // sintaxis 1
204             DelegateVisit delegado = delegate (City city1)
                  { Console.WriteLine(city1.Name + ": " +
                  city1.Population) };
205             DelegateVisit filter = new DelegateVisit(delegado);
206             countryTest.Visit(filter);
207 
208             // sintaxis 2
209             DelegateVisit filter2 = new DelegateVisit((City city1) =>
                  { Console.WriteLine(city1.Name + ": " +
                  city1.Population) });
210             countryTest.Visit(filter2);
211 
212             // sintaxis 3
213             DelegateVisit filter3 = new DelegateVisit((City city1) =>
                  Console.WriteLine(city1.Name + ": " + city1.Population));
214             countryTest.Visit(filter3);
215         }
216         #endregion
217     }
218 }
```