```
 1  namespace DAMLib
 2  {
 3      public class Queue<T>
 4      {
 5          private T[] _queue;
 6
 7
 8          public T First => _queue[0];
 9          public T Last => _queue[Count - 1];
10          public bool IsEmpty => _queue.Length == 0;
11          public int Count
12          {
13              get
14              {
15                  if (_queue == null)
16                      return 0;
17                  else
18                      return _queue.Length;
19              }
20
21          }
22
23
24          public Queue()
25          {
26              _queue = new T[0];
27          }
28
29
30          // Funcion que introduce un elemento generico en el Queue.
31          public void Enqueue(T element)
32          {
33              int count = _queue.Length;
34              T[] _arrayResult = new T[count + 1];
35
36              for(int i = 0; i < count; i++)
37              {
38                  _arrayResult[i] = _queue[i];
39              }
40
41              _arrayResult[count] = element;
42
43              _queue = _arrayResult;
44          }
45
46          // Funcion que extrae un elemento del Queue.
47          public T Dequeue()
48          {
49              int count = _queue.Length;
50              T result = _queue[0];
51              T[] _arrayResult = new T[count - 1];
52
53              for(int i = 0; i < count - 1; i++)
```

```
54              {
55                  _arrayResult[i] = _queue[i + 1];
56              }
57
58              _queue = _arrayResult;
59
60              return result;
61          }
62
63          public void Clear()
64          {
65              _queue = new T[0];
66          }
67
68          public override string ToString()
69          {
70              string result = "";
71              int count = 0;
72
73              foreach(T element in _queue)
74              {
75                  result += $"El elemento {count} de la Queue es:
                      {element}\n";
76                  count++;
77              }
78
79              return result;
80          }
81
82          // Funcion que clona una Queue.
83          public T[] CloneQueue(T[] queue)
84          {
85              T[] result = new T[queue.Length];
86
87              for(int i = 0; i < result.Length; i++)
88              {
89                  result[i] = queue[i];
90              }
91
92              return result;
93          }
94
95          // Funcion que introduce un array de elementos genericos en la
              Queue.
96          public void QueueMultipleElements(T[] elements)
97          {
98              int newElementsCount = elements.Length;
99              int oldElementsCount = _queue.Length;
100
101             T[] newQueue = new T[newElementsCount + oldElementsCount];
102
103             for(int i = 0; i < oldElementsCount - 1; i++)
104             {
```

```
105                 newQueue[i] = _queue[i];
106             }
107 
108             for(int i = 0; i < newElementsCount - 1; i++)
109             {
110                 newQueue[i + oldElementsCount] = elements[i];
111             }
112 
113             _queue = newQueue;
114         }
115     }
116 }
117 
```