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

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

```
105             }
106
107             return result;
108         }
109     }
110 }
```