

```
1 namespace DelegadosConclusion
2 {
3     internal class Program
4     {
5         static void Main(string[] args)
6         {
7             // TEST FILTER
8
9             FilterClass<string> test1 = new FilterClass<string>();
10
11             FilterClass<string>.DelegateFilter del1;
12             del1 = new FilterClass<string>.DelegateFilter(element =>
13                 element.Length > 0);
14
15             test1.Filter(del1);
16             test1.Filter(element => element.Length > 0);
17             test1.Filter(element => { return element.Length > 0; });
18             test1.Filter(element =>
19             {
20                 return element.Length > 0;
21             });
22
23             // TEST SORT
24
25             SortClass<string> test2 = new SortClass<string>();
26
27             SortClass<string>.DelegateSort del2;
28             del2 = new SortClass<string>.DelegateSort((item1, item2) =>
29                 item1.Length < item2.Length ? 1 : 0);
30
31             test2.Sort(del2);
32             test2.Sort((item1, item2) => item1.Length < item2.Length ?
33                 1 : 0);
34             test2.Sort((item1, item2) => { return item1.Length <
35                 item2.Length ? 1 : 0; });
36             test2.Sort((item1, item2) =>
37             {
38                 return item1.Length < item2.Length ? 1 : 0;
39             });
40
41             // TEST VISIT
42
43             List<string> list = new List<string>();
44
45             VisitClass<string> test3 = new VisitClass<string>();
46
47             VisitClass<string>.DelegateVisit del3;
48             del3 = new VisitClass<string>.DelegateVisit(element =>
49                 element = "Elemento: " + element);
50
51             test3.Visit(del3);
```

```
49         test3.Visit(element => element = "Elemento: " + element);
50         test3.Visit(element => {element = "Elemento: " + element;});
51         test3.Visit(element =>
52         {
53             element = "Elemento: " + element;
54         });
55     }
56 }
57 }
```

```
1 namespace DelegadosConclusion
2 {
3
4     public class FilterClass<T>
5     {
6         private List<T> _list = new List<T>();
7
8         public delegate bool DelegateFilter(T element);
9
10        public List<T> Filter(DelegateFilter del)
11        {
12            if (del == null)
13                return null;
14
15            List<T> filterList = new List<T>();
16
17            for (int i = 0; i < _list.Count; i++)
18            {
19                if (del(_list[i]))
20                    filterList.Add(_list[i]);
21            }
22
23            return filterList;
24        }
25    }
26 }
```



```
1 namespace DelegadosConclusion
2 {
3     public class SortClass<T>
4     {
5         private List<T> _list = new List<T>();
6
7         public delegate int DelegateSort(T element1, T element2);
8
9         public List<T> Sort(DelegateSort comparison)
10        {
11            if (comparison == null)
12                return null;
13
14            List<T> listSort = new List<T>();
15
16            for(int i = 0; i < _list.Count - 1; i++)
17            {
18                for(int j = i + 1; i < _list.Count; j++)
19                {
20                    if (comparison(_list[i], _list[j]) >= 1)
21                    {
22                        T aux;
23                        aux = _list[i];
24                        _list[j] = _list[i];
25                        _list[i] = aux;
26                    }
27                }
28            }
29            return listSort;
30        }
31    }
32 }
```



```
1 namespace DelegadosConclusion
2 {
3     public class VisitClass<T>
4     {
5         private List<T> _list = new List<T>();
6
7         public delegate void DelegateVisit(T visitor);
8
9         public void Visit(DelegateVisit visitor)
10        {
11            if (visitor == null)
12                return;
13
14            foreach(T element in _list)
15            {
16                visitor(element);
17            }
18        }
19    }
20 }
21
```