

La cola ideal

La cola del Coppelia tiene ciertas irregularidades por lo que los directivos de ese centro han decidido crear nuevas reglas para evitar conflictos entre los clientes. A continuación se describen estas reglas.

1. Un cliente en la cola tendrá derecho a una mesa de k plazas. La cantidad k es la misma para toda la cola (es decir todas las mesas de una cola son de la misma capacidad).
2. Los clientes serán llamados a su mesa por orden de llegada a la cola. Además no compartirán esa mesa con otros clientes con los que no tengan relación de amistad o con los que teniendo esta relación hayan decidido ponerse en la cola sin colarse.
3. Un cliente tiene la oportunidad de “colarse” si un amigo suyo (no importa si este, a su vez, ha sido colado también) ya se encuentra en la cola y las k plazas de la mesa correspondiente no están ocupadas.

Se desea simular el comportamiento de la cola del Coppelia mediante una estructura de datos. A continuación se describen cada uno de los métodos de esta estructura.

```
/// <summary>
/// Se usa para determinar si dos objetos cumplen con cierto criterio de
/// amistad
/// </summary>
/// <typeparam name="T">Tipo de los objetos</typeparam>
/// <param name="a1">objeto 1</param>
/// <param name="a2">objeto 2</param>
/// <returns>true si son amigos segun este criterio, false en otro
/// caso</returns>
public delegate bool SonAmigos<T>(T a1, T a2);

/// <summary>
/// Simula el comportamiento de la nueva cola del Coppelia
/// </summary>
/// <typeparam name="T">El tipo de clientes en la cola</typeparam>
public class ColaDelCoppelia<T>:IColaCoppelia<T>
{
    /// <summary>
    /// Constructor
    /// </summary>
    /// <param name="sonAmigos">Objeto que define la relación de
    /// amistad</param>
    /// <param name="capacidadPorMesas">Máxima cantidad de clientes por
    /// mesa</param>
    public ColaDelCoppelia(SonAmigos<T> sonAmigos, int capacidadPorMesa)

    /// <summary>
    /// Inserta un cliente al final de la cola otorgándole el derecho a
    /// una mesa con la cantidad de plazas especificadas en el
    /// constructor.
    /// </summary>
```

```

/// <param name="cliente">Objeto que representa al cliente</param>
public void PonerEnCola(T cliente)

/// <summary>
/// Cuela un cliente en la cola. Inserta este cliente junto al amigo
/// más adelantado en la cola en cuya mesa aun quede espacio. De no
/// existir en la cola ningún amigo con espacio en su mesa entonces el
/// comportamiento de este método es igual al del método
/// PonerEnCola.
/// </summary>
/// <param name="cliente">Objeto que representa al cliente</param>
public void Colarse (T cliente)

/// <summary>
/// Extrae de la cola todos los clientes que pertenecen a la mesa en
/// turno y los devuelve en forma de Enumerable. Note que puede que no
/// llenen una mesa completa. Si la cola se encuentra vacía lanza
/// InvalidOperationException.
/// </summary>
/// <returns>
/// Enumerable de los clientes correspondientes a la primera mesa.
/// </returns>
public IEnumerable<T> PasaMesa()

/// <summary>
/// Enumera todos los clientes presentes en la cola.
/// </summary>
public IEnumerable<T> Clientes

/// <summary>
/// Enumera los clientes en la cola por mesas, en el orden que
/// deberian ser atendidos.
/// Cada mesa se representa a su vez por un Enumerable de clientes.
/// </summary>
public IEnumerable<IEnumerable<T>> Mesas

/// <summary>
/// Indica el total de clientes en la cola
/// </summary>
public int TotalClientes

/// <summary>
/// Indica el total de grupos de mesas en la cola.
/// </summary>
public int TotalMesas
}

```

Note que además de los métodos de la *interface* usted **deberá implementar el constructor especificado anteriormente**. El *delegate* [SonAmigos](#) así como la definición de la *interface* estará en una DLL que se encuentra referenciada como parte de la plantilla-proyecto que recibirá junto a esta orientación.

Ejemplo

```
ColaDelCoppelia<FutbolFan> cola =  
    new ColaDelCoppelia<FutbolFan>(  
        (x, y) => x.Club == y.Club || x.Seleccion == y.Seleccion,  
        3  
    );
```

Crea una cola de fans de futbol donde dos personas se consideran amigos si siguen a la misma selección-país o al mismo club, y en la que por cada mesa se admiten sólo 3 personas.

```
cola.PonerEnCola(new FutbolFan("Brasil", "Real Madrid"));  
cola.PonerEnCola(new FutbolFan("Argentina", "Boca Juniors"));  
cola.PonerEnCola(new FutbolFan("Brasil", "Barcelona"));  
cola.PonerEnCola(new FutbolFan("Argentina", "Riverplate"));  
cola.PonerEnCola(new FutbolFan("Holanda", "Manchester Utd"));
```

Después de poner a los clientes anteriores en la cola si se hace

```
Console.WriteLine("Total de mesas: {0}",cola.TotalMesas);  
Console.WriteLine("Total de clientes: {0}",cola.TotalClientes);  
Console.WriteLine("*****");
```

se obtiene el resultado a continuación. Note que aunque el primer y tercer cliente son amigos ocupan distintas mesas ya que el segundo en la cola **no se puso** mediante el método `Colarse`.



```
Total de mesas: 5  
Total de clientes: 5  
*****
```

1 Resultado parcial de la ejecución del código de ejemplo.

Si ahora se hiciera

```
cola.Colarse(new FutbolFan("Brasil", "Ajax"));  
cola.Colarse(new FutbolFan("Mexico", "Ajax"));
```

ambas personas encuentran amigos en la primera mesa por lo que no tienen que ponerse al final de la cola. El primero en colarse porque el primero en la cola es fan de la selección de Brasil y el segundo en colarse porque es amigo del que se acaba de colar al compartir su afición por el club Ajax.

Si ahora se hiciera

```
cola.Colarse(new FutbolFan("Brasil", "Oporto"));
```

esta persona encontraría amigos en la primera mesa pero ya no hay capacidad en esa mesa por lo que tiene que sentarse con el amigo de la tercera mesa que comparte su afición por la selección de Brasil.

El resultado de hacer nuevamente

```
Console.WriteLine("Total de mesas: {0}",cola.TotalMesas);
```

```
Console.WriteLine("Total de clientes: {0}", cola.TotalClientes);
Console.WriteLine("*****");
```

sería ahora:



```
Total de mesas: 5
Total de clientes: 8
*****
```

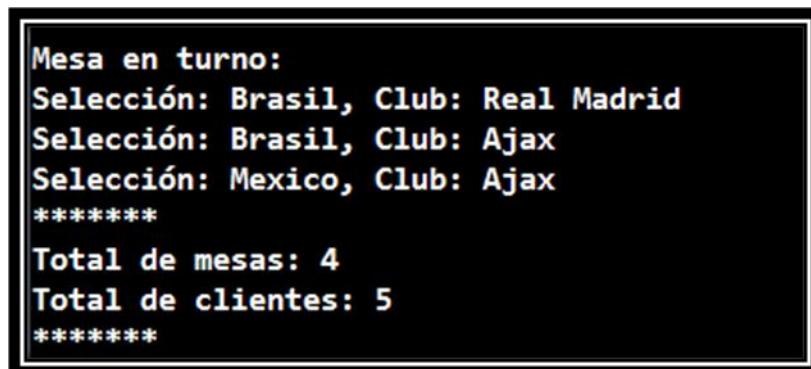
2 Resultado parcial de la ejecución del código de ejemplo.

La ejecución de las siguientes líneas

```
IEnumerable<FutbolFan> personasAPasar = cola.PasaMesa();
Console.WriteLine("Mesa en turno:");
foreach (FutbolFan fan in personasAPasar)
    Console.WriteLine(fan.ToString());

Console.WriteLine("*****");
Console.WriteLine("Total de mesas: {0}", cola.TotalMesas);
Console.WriteLine("Total de clientes: {0}", cola.TotalClientes);
Console.WriteLine("*****");
```

Provocaría la siguiente salida. Note que han salido de la cola todos los de la primera mesa



```
Mesa en turno:
Selección: Brasil, Club: Real Madrid
Selección: Brasil, Club: Ajax
Selección: Mexico, Club: Ajax
*****
Total de mesas: 4
Total de clientes: 5
*****
```

3 Resultado parcial de la ejecución del código de ejemplo.

Las siguientes personas encuentran amigos en la cola y se “cuelan”

```
cola.Colarse(new FutbolFan("Argentina", "Milan AC"));
cola.Colarse(new FutbolFan("Uruguay", "Boca Juniors"));
```

Note como el *enumerable* *Mesas* permite conocer el estado exacto de la cola

```
int mesaActual = 1;
foreach (IEnumerable<FutbolFan> mesa in cola.Mesas)
{
    Console.WriteLine("Mesa {0}", mesaActual++);
    foreach (FutbolFan fan in mesa)
        Console.WriteLine(fan.ToString());
}
Console.WriteLine("*****");
```

```

Mesa 1
Selección: Argentina, Club: Boca Juniors
Selección: Argentina, Club: Milan AC
Selección: Uruguay, Club: Boca Juniors
Mesa 2
Selección: Brasil, Club: Barcelona
Selección: Brasil, Club: Oporto
Mesa 3
Selección: Argentina, Club: Riverplate
Mesa 4
Selección: Holanda, Club: Manchester Utd
*****

```

4 Resultado parcial de la ejecución del código de ejemplo.

Si ahora hacemos

```
cola.Colarse(new FutbolFan("Uruguay", "Liverpool"));
```

esta persona encuentra amigos adelantados en la cola pero como la mesa correspondiente está completa, y no tiene más amigos en la cola, tiene que ponerse al final

```

Console.WriteLine("Total de mesas: {0}", cola.TotalMesas);
Console.WriteLine("Total de clientes: {0}", cola.TotalClientes);
Console.WriteLine("*****");

```

```

Total de mesas: 5
Total de clientes: 8
*****

```

5 Resultado parcial de la ejecución del código de ejemplo.

Note que aunque este ejemplo se ha ilustrado para el tipo `FutbolFan` y para la función de amistad

```
(x, y) => x.Club == y.Club || x.Seleccion == y.Seleccion
```

su solución debe servir para cualquiera sea el tipo `T` de los elementos que se pongan en cola y para cualquier función de tipo

```
delegate bool SonAmigos<T>(T a1, T a2);
```

que se le dé cómo función de amistad.