Colados por amistad

En muchas situaciones de la vida diaria se ven colas, y en cada cola siempre existen los "colados", aquellos que llegan para ponerse delante de otros que ya se encontraban en la cola. El problema que se quiere resolver es limitar la acción de colarse, implementando para ello una **Cola Amigable** que controle esta acción como se describirá más adelante.

Deberá implementar una clase **`ColaAmigable`** que implemente la interfaz `IColaAmigable`:

```
interface IColaAmigable<T> : IEnumerable
{
    void Colarse(T x);
    void Enqueue(T x);
    T Dequeue();
    int Count { get; }
}
```

La clase `ColaAmigable` deberá tener además, el siguiente constructor:

```
public class ColaAmigable<T> : IColaAmigable<T>
{
      public ColaAmigable(IAmistad<T> criterioAmistad)
      {
          // Implementación del constructor
      }
}
```

Donde la interfaz `IAmistad`, cuyo código se muestra a continuación, nos debe ofrecer un método para determinar si dos elementos de la cola son o no amigos:

```
interface IAmistad<T>
{
    bool SonAmigos(T x, T y);
}
```

El método `void Colarse(T x)` deberá "colar" el elemento `x` en la cola. Para determinar cómo colarse, deberá usar el método `bool SonAmigos(T x, T y)` de la instancia pasada como parámetro en el momento de crear una `ColaAmigable`. El elemento se insertará delante del primer elemento "amigo" que se encuentre en la cola, si y solo si el elemento de la cola hallado no fue previamente colado o hubo colado a alguien antes. En caso de no encontrar ningún elemento delante del cual colarse, el elemento debe insertarse al final de la cola.

La interfaz `IColaAmigable<T>` y la interface `IAmistad<T>` se proporcionarán en una DLL llamada **`ColaAmigos.Utiles.dll`** para su uso en la implementación de la solución del problema. Usted deberá implementar la clase `ColaAmigable` dentro de un proyecto de tipo *Class Library* (Biblioteca de Clases) de nombre **`ProblemaColaAmigable`**.

Para simplificar solo se le pide garantizar que `IColaAmigable<T>` y por tanto `ColaAmigable<T>` implemente a `IEnumerable` y no a `IEnumerable<T>`.

IMPORTANTE

En su implementación **NO PODRÁ USAR** las estructuras de datos existentes para almacenar los elementos de la cola, es decir no podrá usar tipos como *arrays*, `Queue`, `ArrayList`, `List`, ni sus variantes genéricas. Todo lo que necesite deberá implementarlo totalmente. Las respuestas que incumplan con esto serán anuladas.

Ejemplo 1. StringAmistad

Dada la siguiente clase que implementa la interfaz `IAmistad<T>`:

```
class StringAmistad : IAmistad<string>

{
    public bool SonAmigos(string x, string y)
    {
        return x[0] == y[0]; // Son amigos si el primer carácter es el mismo
    }
}
```

Y el siguiente fragmento de código:

```
ColaAmigable<string> queue = new ColaAmigable<string>(new StringAmistad());

queue.Enqueue("horacio");
queue.Colarse("sonia");
queue.Enqueue("gladys");
queue.Colarse("saul");
queue.Enqueue("carlos");
queue.Colarse("carmen");
queue.Enqueue("ulises");
queue.Colarse("caridad");
queue.Colarse("homero");
queue.Colarse("hanna");
queue.Dequeue();
queue.Enqueue("henry");

foreach(string s in queue)
// Este cast no dará excepción porque todos los elementos guardados
// en la cola son string
{
    Console.WriteLine(s);
}
```

Deberá imprimir la siguiente secuencia (revise bien el ejemplo para que se familiarice con el comportamiento de la cola):

```
horacio
saul
sonia
gladys
carmen
carlos
ulises
```

```
caridad
hanna
henry
```

Ejemplo 2. FechaAmistad

Dada la siguiente clase que implementa la interfaz `IAmistad<T>`:

```
class FechaAmistad : IAmistad<DateTime>

{
    public bool SonAmigos(DateTime x, DateTime y)
    {
        // Son amigos si son fechas del mismo mes y
        // corresponden al mismo día de la semana

        return (x.DayOfWeek == y.DayOfWeek) && (x.Month == y.Month);
    }
}
```

Y el siguiente fragmento de código:

```
ColaAmigable<DateTime> q = new ColaAmigable<DateTime>(new FechaAmistad());

q.Enqueue(DateTime.Parse("31/10/1978"));     // Martes
q.Enqueue(new DateTime(2003, 1, 6));         // Lunes
q.Colarse(DateTime.Parse("02/12/1956"));     // Domingo
q.Colarse(DateTime.Parse("09/10/1956"));     // Martes
q.Dequeue();
q.Colarse(DateTime.Parse("28/01/2003"));     // Martes
q.Enqueue(DateTime.Parse("16/08/1963"));     // Viernes
q.Colarse(DateTime.Parse("25/10/2000"));     // Miércoles
q.Dequeue();
q.Colarse(DateTime.Parse("31/10/2008"));     // Viernes

foreach(DateTime date in q)
{
    Console.WriteLine(date.ToShortDateString());
}
```

Deberá imprimir la siguiente secuencia:

```
06/01/2003
02/12/1956
28/01/2003
16/08/1963
25/10/2000
31/10/2008
```