Problema 3 de Programación Curso 2013-2014

Cola Amigable

NOTA: Si usted está leyendo este documento sin haber extraído el compactado que descargó del sitio, ciérrelo ahora, extraiga todos los archivos en el escritorio, y siga trabajando desde ahí. Es un error común trabajar en la solución dentro del compactado, lo cual provoca que los cambios **no se guarden**. Si usted comete este error y entrega una solución vacía, no tendrá oportunidad de reclamar.

La siguiente `interface` ha sido diseñada para describir cuando dos elementos son conocidos:

```
public interface IConocido<T>
{
    /// <summary>
    /// Verifica si la instancia actual conoce al elemento que se pasa como
    /// parámetro. Devuelve true en caso de que se conozcan y false en caso
    /// contrario.
    /// </summary>
    /// <param name="aquien">Elemento para el que se va a verificar que sea o no
    /// conocido de la instancia.
    /// </param>
    bool Conoce(T aquien);
}
```

La `interface` que se muestra a continuación ha sido diseñada para encapsular las operaciones básicas de una Cola Amigable.

```
public interface IColaAmigable<T> where T:IConocido<T>
{
    /// <summary>
    /// Añade el elemento a la cola amigable. El elemento deberá insertarse
    /// delante del primer conocido que encuentre en la cola, si y solo si este
    /// conocido no es a su vez un colado.
    /// En caso de no encontrar ningún conocido que no se haya
    /// colado antes, se deberá incorporar el nuevo elemento al final de la cola.
    /// </summary>
    /// <param name="elemento">Elemento que deberá añadirse a la cola amigable.
    /// </param>
    void Enqueue(T elemento);

    /// <summary>
    /// Quita el primer elemento de la cola, devolviendo su valor.
    /// </summary>
    T Dequeue();

    /// <summary>
    /// Devuelve el valor del primer elemento de la cola, sin quitarlo.
    /// </summary>
    T Peek();

    /// <summary>
    /// Devuelve la cantidad de elementos almacenados en la cola.
    /// </summary>
    int Count
    {
        get;
    }
```

```
    /// <summary>
    /// Devuelve un iterador que permite iterar por los elementos que fueron
    /// colados por un elemento específico.
    /// </summary>
    /// <param name="quien">Elemento delante del que se colaron
    /// todos los valores por los que se itera.</param>
    IEnumerable<T> ColadosPor(T quien);
}
```

NOTA: Las interfaces descritas anteriormente ya están definidas como parte de los ensamblados que se proveen para la realización del examen. Estas interfaces han sido definidas para usarse tal y como se ha descrito, usted no debe definir ninguna nueva interface.

Problema

Usted debe proveer una implementación de la `interface IColaAmigable` descrita anteriormente, con el nombre `ColaAmigable`. Esta clase deberá simular el comportamiento de una cola en la que los elementos intentan colarse siempre delante del primer elemento conocido que, a su vez, no haya sido colado anteriormente. El primer elemento de la cola podrá ser obtenido o eliminado de igual manera que en una cola normal. Los elementos que se han colado delante de otro podrán ser recuperados usando el enumerador que devuelve el método `ColadosPor`. Este método deberá hacer uso de la implementación del método Equals para buscar al elemento en la cola.

ACLARACIONES:

- Su implementación deberá contener un único constructor, sin parámetros, que será usado en la evaluación automática para crear instancias de esta clase. En caso de no contener este constructor todos los casos de prueba se considerarán fallidos.
- En su implementación no podrá emplear ninguna estructura de los namespaces `System.Collections` ni `System.Collections.Generic`. Tampoco podrá hacer uso de arrays. Por tanto, se recomienda el uso de estructuras enlazables.

Ejemplos

A continuación se muestran algunos ejemplos del uso de esta clase. Estos ejemplos hacen uso de la clase `Estudiante`, que forma parte de los ensamblados que se proveen para la realización del examen.

```
public class Estudiante : IConocido<Estudiante>
{
    public string Nombre { get; set; }

    public string Grupo { get; set; }

    public Estudiante(string nombre, string grupo)
    {
        Nombre = nombre;
        Grupo = grupo;
    }

    public bool Conoce(Estudiante aquien)
    {
        return Grupo == aquien.Grupo;
    }
```

```
    public override bool Equals(object obj)
    {
        if (obj is Estudiante)
        {
            var estudiante = obj as Estudiante;
            return Grupo == estudiante.Grupo && Nombre == estudiante.Nombre;
        }
        return false;
    }
}
```

En esta implementación dos estudiantes son conocidos si pertenecen al mismo grupo.

Recuerde que aunque el ejemplo usa el tipo `Estudiante` para evaluar su trabajo se podrá usar otro tipo `T` que implemente la `IConocido<T>` para probar su cola amigable.

Note que el método `Equals` ha sido redefinido para que sea utilizado correctamente por el iterador de la cola amigable.

```
ColaAmigable<Estudiante> colaAmigable = new ColaAmigable<Estudiante>();

colaAmigable.Enqueue(new Estudiante("Jose", "C312"));    // Jose
colaAmigable.Enqueue(new Estudiante("Maria", "C312"));   // Maria-Jose

colaAmigable.Peek(); // Maria

colaAmigable.Enqueue(new Estudiante("Juan", "C513"));    // Maria-Jose-Juan
colaAmigable.Enqueue(new Estudiante("Amanda", "C312"));  // Maria-Amanda-Jose-Juan

int count1 = colaAmigable.Count; // 4

colaAmigable.ColadosPor(new Estudiante("Jose", "C312")); // Maria-Amanda

colaAmigable.ColadosPor(new Estudiante("Amanda", "C312")); // ningún elemento

colaAmigable.ColadosPor(new Estudiante("Juan", "C513")); // ningún elemento

colaAmigable.Dequeue();  // Amanda-Jose-Juan

int count2 = colaAmigable.Count; // 3

colaAmigable.ColadosPor(new Estudiante("Jose", "C312")); // Amanda
```