Problema 3 de Programación Curso 2013-2014

Cola con Prioridad

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 `interface` que se muestra a continuación ha sido diseñada para encapsular las operaciones básicas de una cola con prioridad.

```
public interface IColaConPrioridad<T>
{
    /// <summary>
    /// Añade el elemento a la cola con prioridad. El elemento deberá insertarse
    /// delante del primer elemento cuya prioridad sea menor estricta que la del
    /// elemento a insertar.
    /// En caso de no encontrar ningún elemento que cumpla la anterior condición
    /// se deberá incorporar el nuevo elemento al final de la cola.
    /// </summary>
    /// <param name="elemento">Elemento que deberá añadirse a la cola.</param>
    /// <param name="p">Prioridad del elemento a insertar.</param>
    void Enqueue(T elemento, int p);

    /// <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 están en la
    /// cola y que tienen la misma prioridad con que se insertó el elemento pasado
    /// como parámetro.
    /// Si el elemento pasado como parámetro no está en la cola, este método
    /// devuelve un iterador vacío.
    /// </summary>
    IEnumerable<T> ElementosConIgualPrioridad(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` `IColaConPrioridad` descrita anteriormente, con el nombre `ColaConPrioridad`. Esta clase deberá simular el comportamiento de una cola en la que los elementos tienen asociada una prioridad que determina un orden en que se ubican con respecto a los otros elementos de la cola. La prioridad se representa con un valor numérico entero y mientras mayor sea el mismo mayor será la prioridad. El primer elemento de la cola es el de mayor prioridad y pude ser obtenido (`Peek`) o eliminado (`Dequeue`) de igual manera que en una cola normal respetando el orden definido por las prioridades con las que se insertaron los elementos.

Dado un elemento, se puede obtener un enumerador de aquellos elementos que tienen igual prioridad que éste en la cola mediante el método `ElementosConIgualPrioridad`. Estos elementos (con igual prioridad) se devuelven en el mismo orden en que fueron insertados. El elemento buscado forma parte del enumerador. Si el elemento no se encuentra originalmente en la cola, el enumerador devuelto será vacío. El método `ElementosConIgualPrioridad` deberá hacer uso de la implementación del método Equals para buscar el 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.

```
ColaConPrioridad<string> cola = new ColaConPrioridad<string>();

cola.Enqueue("azul", 3);    // azul
cola.Enqueue("rojo", 4);   // rojo, azul

cola.Peek(); // El resultado debe ser rojo'.

cola.Enqueue("verde", 3);    // rojo, azul, verde
cola.Enqueue("blanco", 8);  // blanco, rojo, azul, verde

int count1 = cola.Count; // El resultado debe ser 4.

cola.ElementosConIgualPrioridad("verde"); // El resultado debe ser 'azul, verde'.

cola.ElementosConIgualPrioridad("cyan"); // El resultado debe ser un iterador vacío.

string s = cola.Dequeue(); // El resultado debe ser 'blanco'.

int count2 = cola.Count; // El resultado debe ser 3 (rojo, azul, verde).

cola.ElementosConIgualPrioridad("rojo"); // La respuesta debe ser 'rojo'.
```