COLA PARTICIONADA

Tercer Problema de Programación

Curso 2010-2011

La cola particionada es una estructura de datos que permite agregar y extraer datos con un comportamiento similar al de una cola. No obstante, tiene algunas diferencias, una cola particionada tiene además conocimiento de un conjunto de **categorías** que permite tener "agrupados" los elementos que son encolados.

Para cada elemento a encolar se define a qué categoría pertenece.

El proceso de **desencolar** en este tipo de cola se define de la siguiente manera:

- Se comienza a extraer de las categorías según se fueron añadiendo éstas.
- De cada categoría se extrae una cantidad de elementos determinada (que depende de la categoría) de forma consecutiva.
- Cuando se haya extraído esa cantidad de elementos de la categoría (o no haya más elementos) se pasa a la categoría siguiente. Las categorías se recorren de modo circular una vez que termine el turno de la última categoría se vuelve a empezar con la primera (Ver ejemplo 1).
- Si se terminan los elementos de una categoría estando en el turno de esta y quedando aún con derecho a sacar más elementos de dicha categoría entonces si se encola un elemento en dicha categoría, este sería el próximo a ser desencolado (Ver ejemplo 2).
- El paso a la siguiente se realiza cuando se requiere desencolar un nuevo elemento. Es decir, si antes de desencolar un elemento (que produciría el paso a la siguiente categoría) se agregara una nueva categoría que fuese la siguiente, le correspondería a esta (Ver ejemplo 3).

A continuación se definen las funcionalidades de este tipo de estructura.

DEFINICIÓN DE LA CLASE

```
public class ColaConCategorias<T> : IEnumerable<T>
```

CONSTRUCTOR

```
public ColaConCategorias(){…}
```
Crear una cola con categorías. Inicialmente la cola está vacía y sin categorías.

MÉTODOS Y PROPIEDADES

```
public void AgregaCategoria(string categoria, int cantidad){…}
```
Agrega una nueva categoría en la cola. La categoría se describe con un nombre (`string`). Al extraer elementos de la cola se van atendiendo las categorías según se fueron añadiendo.

Si el valor del parámetro `categoria` es `null` se deberá lanzar la excepción `ArgumentNullException`. Si el valor ya se encuentra entre las categorías agregadas en la cola actual, se deberá lanzar la excepción `ArgumentException`.

El parámetro `cantidad` indica cuántos elementos son desencolados de forma consecutiva cuando corresponda extraer de dicha categoría. Si el valor de `cantidad` es menor que 1 se deberá lanzar la excepción `ArgumentOutOfRangeException`.

`public IEnumerable<string> Categorias {get {…}}`
Propiedad que devuelve un IEnumerable con todas las categorías en el orden en que se fueron añadiendo a la cola.

`public void Enqueue(string categoria, T valor){…}`
Encola un determinado valor en una de las categorías de la cola. El parámetro `categoria` representa un texto que identifica el nombre de la categoría en la que se encolará el elemento. Si este parámetro es `null` se debe lanzar la excepción `ArgumentNullException`. Si el parámetro tiene un valor que no se encuentra entre las categorías de la cola actual se debe lanzar la excepción `ArgumentOutOfRangeException`.

No hay problemas en que `valor` sea `null` o que anteriormente se haya encolado otro igual.

`public T Dequeue(){…}`
Se extrae un elemento de la cola según el criterio de extracción explicado anteriormente, teniendo en cuenta el orden en que se incluyeron las categorías, la cantidad de elementos consecutivos que se pueden extraer de una categoría y la categoría de la que se esté extrayendo. Quita el elemento de la cola. Si la cola no contiene elementos este método deberá lanzar `InvalidOperationException`.

`public T Peek(){…}`
Hace lo mismo que `Dequeue` pero sin quitar el elemento de la cola. Si la cola no contiene elementos este método deberá lanzar `InvalidOperationException`.

`public int Count {get{…}}`
Devuelve la cantidad de elementos de la cola.

`public IEnumerator<T> GetEnumerator(){…}`
Devuelve un objeto `IEnumerator` que permite recorrer los elementos de la cola en el orden en que estos serían desencolados si no se produjese ninguna operación de `Enqueue`. Se garantiza que durante la iteración sobre este iterador no se le harán operaciones `Enqueue` ni `Dequeue` a la cola.

Ejemplos:

Ejemplo 1. Cola con 3 categorías a la que se le realiza varias inserciones, extracciones y recorridos.

```
ColaConCategorias<string> q = new ColaConCategorias<string>();

string catPlanJaba = "Plan Jaba";
string catNormal = "Normal";
string catEmbarazadas = "Embarazadas";

q.AddCategory(catPlanJaba, 2);
q.AddCategory(catNormal, 1);
q.AddCategory(catEmbarazadas, 4);

q.Enqueue(catPlanJaba, "Juan");
q.Enqueue(catPlanJaba, "Pedro");
q.Enqueue(catNormal, "Jose");
q.Enqueue(catEmbarazadas, "Ana");
q.Enqueue(catNormal, "Maria");
q.Enqueue(catNormal, "Carlos");
q.Enqueue(catEmbarazadas, "Arnold");
q.Enqueue(catPlanJaba, "Jesus");

ImprimeCola(q); // este método enumera los elementos de la cola y los imprime…
// Imprime: Juan, Pedro, Jose, Ana, Arnold, Jesus, Maria, Carlos

q.Dequeue();
q.Dequeue();
q.Dequeue();

q.Enqueue(catEmbarazadas, "Lola");
q.Enqueue(catPlanJaba, "Raul");
ImprimeCola(q);
// Imprime: [Ana, Arnold, Lola, Jesus, Raul, Maria, Carlos]
```

EJEMPLO 2. EL DE LA CATEGORÍA “ROJO” AÚN TIENE DERECHO.

```
ColaConCategorias<int> q = new ColaConCategorias<int>();

q.AgregaCategoria("rojo", 10);
q.AgregaCategoria("verde", 2);
q.AgregaCategoria("azul", 1);

q.Enqueue("rojo", 1);
q.Enqueue("rojo", 2);
q.Enqueue("rojo", 3);
q.Dequeue();
q.Dequeue();
q.Dequeue();
q.Enqueue("verde", 4);
q.Enqueue("verde", 5);
q.Enqueue("verde", 6);
q.Enqueue("azul", 7);
q.Enqueue("rojo", 8);
ImprimeCola(q);
// Imprime: [8, 4, 5, 7, 6]
```

EJEMPLO 3. LA "NUEVA CATEGORÍA" AÚN TIENE DERECHO.

```
ColaConCategorias<string> q = new ColaConCategorias<string>();

q.AgregaCategoria("A", 1);
q.Enqueue("A", "A1");
Console.WriteLine(q.Dequeue());

q.AgregaCategoria("B", 2);
q.Enqueue("B","B1");
q.Enqueue("B","B2");
Console.WriteLine(q.Dequeue());
Console.WriteLine(q.Dequeue());

q.Enqueue("A", "A2");
q.Enqueue("B", "B3");
q.Enqueue("B", "B4");

Console.WriteLine("Peek :" + q.Peek());

q.AgregaCategoria("C", 1);
q.Enqueue("C","C1");
ImprimeCola(q);
//Imprime:
A1
B1
B2
Peek :A2
[C1,A2,B3,B4]
```