Rebelión en la Despensa

> Examen Mundial de Programación - Curso 2022-2023

NOTA: *Antes de comenzar asegúrese de descompactar el archivo* `RebelionDespensa.zip`*. Asegúrese también de que su código compila, y la aplicación de consola ejecuta (debe lanzar una excepción). Recuerde que todo el código a evaluar debe ir en el archivo* `Exam.cs`

En la cocina de una cafetería todas las noches los ingredientes se reúnen y realizan combates para establecer de entre ellos cual son los más sabrosos. Los alimentos, interesados en conocer cuales de estos tienen mejor sabor, le han asignado a Roberto la croqueta la tarea de diseñar un sistema de clasificación tal que todos los alimentos puedan verse reflejados de acuerdo a cuan deliciosos se consideran.

Roberto al no ser graduado de MATCOM (es una croqueta) te pide que construyas un programa que le permita, dada una clasificación inicial de los alimentos construida por él, realizar ciertas operaciones sobre estos.

Alimentos

Roberto ha decidido utilizar un árbol binario para almacenar las clasificaciones de los alimentos. Cada nodo del árbol representa un alimento de la despensa. El árbol cumple las siguientes propiedades:

- Dado un nodo del árbol, siempre se cumple que sus hijos tienen peor sabor que él. Cada alimento se encuentra posicionado en el árbol de acuerdo a cuan sabroso se determine este; el alimento más exquisito representa la raíz de dicho árbol.
- Además la cantidad de alimentos que participan de los combates nocturnos es una potencia de dos, lo que garantiza que el árbol binario sea completo.

Los nodos definidos por Roberto utilizan la siguiente interfaz:

```
public interface INode
{
    public string Name { get; }
    public int Tastiness {get;}

    public INode? Parent {get; set; }
    public INode? Left { get; set; }
    public INode? Right { get; set; }

    public INode Tasty {get; }
    public INode Bland {get; }
}
```

- `Name` representa el nombre del alimento
- `Tastiness` cuan sabroso se le considera
- `Parent`, `Left` y `Right` representan el nodo padre y los nodos hijos respectivamente.
- `Tasty` retorna dado los nodos hijos, el más sabroso de los dos y `Bland` el más ínspido. Si el nodo no tiene dos hijos para comparar, estas propiedades lanzan una excepción.

Sistema de Clasificación

Para implementar las funcionalidades necesarias sobre el sistema de clasificación debe apoyarse sobre la interfaz `IRankings`:

```
public interface IRankings
{
    public INode Root {get; }

    public void Remove(string name);
    public void Insert(INode node);
    public INode Find(Func<INode, bool> query);
    public IEnumerable<INode> GetByLevel(int level);
}
```

La propiedad `Root` simboliza la raíz del árbol o el alimento más delicioso.

Remove

Es posible que de una noche a la siguiente un alimento sea utilizado durante el almuerzo y por tanto sea necesario eliminarlo del sistema de clasificación. Para esto se define el método `Remove` que dado el nombre del alimento lo elimina del árbol.

```
public void Remove(string name);
```

Al eliminar un alimento pueden darse los siguientes casos:

- El alimento es un nodo hoja y por tanto se puede eliminar trivialmente.
- El alimento contiene hijos y por tanto su eliminación significa la necesidad de reordenar el árbol. Cuando esto ocurre, existe un vacío en el árbol que pasa a ser ocupado por el nodo hijo de mayor sabor (**S**). No obstante esto implica que el hijo de menor sabor (**I**) queda desconectado del árbol.

  Es necesario añadir nuevamente **I** sin violar las propiedades del árbol. Para esto:

  1. Se desconecta **I'**, el hijo de menor sabor de **S**, y se conecta **I** en su lugar.
  2. Luego **I'** queda desconectado y es necesario reubicarlo. Se repite el paso 1, reconectando **I'** como hijo de **S'** , el nodo de mayor sabor de **S**.

> En el archivo *remover.pdf* se muestra el funcionamiento general del algoritmo.

Al finalizar el proceso de eliminación, quedan $2^n - 1$ nodos, lo que implica que el árbol deja de estar completo. Roberto dice que no es de preocupación pues nunca va a realizar dos eliminaciones consecutivas.

Insert

Roberto desea que, dada la eliminación de un elemento del árbol, tener la posibilidad de añadir uno nuevo. Para esto se define el método `Insert` que recibe como parámetro el nodo nuevo a añadir.

La inserción de un elemento en el árbol solo puede realizarse cuando existe una posición vacía en este. Solo es posible insertar cuando exista un nodo que tenga un sólo hijo. Si el árbol se encuentra completo (posee 2^n nodos) se espera que este método lance una excepción.

```
public void Insert(INode node);
```

Si el nuevo alimento añadido tiene mejor sabor que su padre, se espera que este y su padre intercambien posiciones tal que las propiedades del árbol se mantengan. Este proceso de intercambio se realiza tantas veces sea necesario hasta que el nodo nuevo llegue a uno posición donde no tenga mejor sabor que su padre.

> En el archivo *insertar.pdf* se muestra un ejemplo de inserción.

Find

Roberto está interesado en ocasiones en buscar nodos que cumplan determinadas condiciones. Espera que tras utilizar el método `Find` obtener un nodo que cumpla con la condición(es) especificada en `query`.

```
public INode Find(Func<INode, bool> query);
```

En caso de no exisitir ningún elemento, se debe lanzar una exepción.

GetByLevel

Finalmente puede ser interesante agrupar los alimentos de acuerdo a su nivel dentro del árbol. Para esto Roberto te pide que definas el método `GetByLevel`:

```
public IEnumerable<INode> GetByLevel(int level);
```

Dado un nivel especificado se espera que se devuelva una lista con todos alimentos que pertenecen a ese nivel. La raíz se encuentra en el nivel 1. Si el nivel especificado es 0, o mayor que la altura del árbol se espera que el programa lance una excepción.

Implementando el Sistema de Clasificación

Usted debe dar una implementación de `IRanking`. Las clase que implementa esta interfaz, y todo el código adicional que haga falta para su funcionamiento, deben estar en el archivo `Exam/Exam.cs`, que será **el único archivo evaluado**.

Para evaluar su código, se ejecutará el método `CreateRanking` en la clase `Exam`. En este método usted debe devolver una instancia de su implementación de la interfaz `IRankings`, utilizando como raíz del árbol al parámetro `root`.

En el proyecto `Exam`, archivo `Program.cs`, que es una aplicación de consola, usted puede adicionar todo el código que considere necesario para probar su implementación. Ese código no será evaluado. Además en `Program.cs` cuenta con una implementación de `INode` y un método auxiliar `TreeRepresentaion` que obtiene una representación del árbol en texto.

Para todos los llamados a todos los métodos que no sean correctos, usted debe lanzar una excepción correspondiente.

Observaciones Adicionales

1. El sabor de dos pares de alimentos nunca coincide.
2. El nombre de cada alimentos es único.

Ejemplo

A continuación mostramos un ejemplo sencillo. El código de este ejemplo está en el método `Main` de la clase `Program`.

Primero se crea la instancia del Sistema de Clasificación utilizando un árbol previamente construido.

```
        // Se crea el nodo de nivel 1
        Node root = new Node("bacon", 100);
        // Se crean los nodos de nivel 2
        Node a = new Node("queso", 80);
        Node b = new Node("leche", 60);
        // Se crean los nodos de nivel 3
        Node aa = new Node("brocolí", 1);
        Node ab = new Node("hígado", 2);
        Node ba = new Node("mango", 50);
        Node bb = new Node("tomate", 45);

        // Se conforma el árbol
        root.SetLeft(a);
        root.SetRight(b);
        a.SetLeft(aa);
        a.SetRight(ab);
        b.SetLeft(ba);
        b.SetRight(bb);

        // Se crea el Sistema de Clasificación sobre el árbol
        IRankings rankings = Exam.CreateRankings(root);
```

El árbol construido tiene la siguiente forma:

```
bacon-100:
        L: queso-80:
                L: brocolí-1
                R: hígado-2
        R: leche-60:
                L: mango-50
                R: tomate-45
```

Luego se pasa a remover la raíz de este:

```
        rankings.Remove("bacon");
```

Note que en el árbol resultante, debido a la eliminación un nodo solo contiene un hijo.

```
queso-80:
        L: leche-60:
                L: mango-50
                R: tomate-45
        R: hígado-2:
                L: brocolí-1
                R: Null
```

El siguiente paso es añadir un nuevo nodo utilizando `Insert`. Se introduce en la posición vacía y debe ascender en el árbol hasta su posición correspondiente.

```
        Node friedBacon = new Node("bacon frito", 150);
```

El árbol resultante debe ser similar a:

```
bacon frito-150:
        L: leche-60:
                L: mango-50
                R: tomate-45
        R: queso-80:
                L: brocolí-1
                R: hígado-2
```

Se comprueba que los elementos del nivel 2 sean `mango`, `tomate`, `brocolí` e `hígado`.

```
        IEnumerable<INode> lvl2Food = rankings.GetByLevel(2);
        Debug.Assert(lvl2Food.Count() == 4);
        Debug.Assert(lvl2Food.Contains(aa));
        Debug.Assert(lvl2Food.Contains(boiledInsect));
        Debug.Assert(lvl2Food.Contains(ba));
        Debug.Assert(lvl2Food.Contains(bb));
```

Se elimina y se añade un nodo hoja al sistema de clasificación.

```
        // se elimina el nodo que representa al mangp
        rankings.Remove(aa.Name);
        lvl2Food = rankings.GetByLevel(2);
        Debug.Assert(lvl2Food.Count() == 3);
        Debug.Assert(lvl2Food.Contains(ab));
        Debug.Assert(lvl2Food.Contains(ba));
        Debug.Assert(lvl2Food.Contains(bb));
        rankings.Remove(aa.Name);

        // se añade de vuelta
        rankings.Insert(aa);
        lvl2Food = rankings.GetByLevel(2);
        Debug.Assert(lvl2Food.Count() == 4);
        Debug.Assert(lvl2Food.Contains(aa));
        Debug.Assert(lvl2Food.Contains(ab));
        Debug.Assert(lvl2Food.Contains(ba));
        Debug.Assert(lvl2Food.Contains(bb));
```

Finalmente se remueve el nodo que representa al queso y se añade la pizza hawaina.

```
        rankings.Remove(a.Name);
        Node pizza = new Node("Pizza Hawaiana", 120);
        rankings.Insert(pizza);
```

El arbol resultante debe tener la siguiente estructura.

```
bacon frito-150:
        L: leche-60:
                L: mango-50
                R: tomate-45
        R: Pizza Hawaiana-120:
                L: brocolí-1
                R: hígado-2
```