Universidad del Istmo de Guatemala
Facultad de Ingenieria
Ing. en Sistemas
Informatica 2
Prof. Ernesto Rodriguez - erodriguez@unis.edu.gt

Hoja de trabajo #8

Fecha de entrega: 27 de Abril, 2018 - 11:59pm

Instrucciones: Realizar cada uno de los ejercicios siguiendo sus respectivas instrucciones. El trabajo debe ser entregado a traves de Github, en su repositorio del curso, colocado en una carpeta llamada "Hoja de trabajo 8". Al menos que la pregunta indique diferente, todas las respuestas a preguntas escritas deben presentarse en un documento formato pdf, el cual haya sido generado mediante Latex. Los ejercicios de programación deben ser colocados en una carpeta llamada "Programas", la cual debe colocarse dentro de la carpeta correspondiente a esta hoja de trabajo.

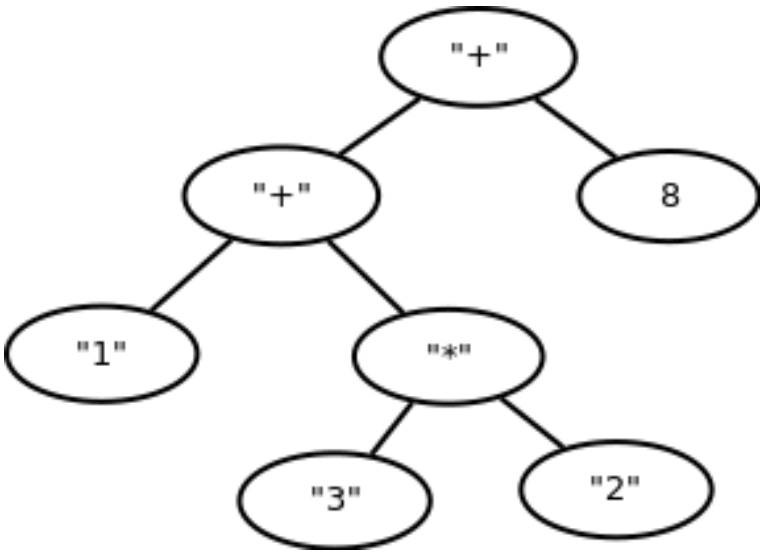
Iniciación

Crear una solución llamada *ParseTree*. Dentro de esta solución crear:

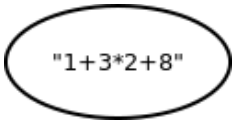
- un proyecto llamado *ParseTree* de tipo console
- un proyecto llamado *ParseTreeTests* de tipo xunit

Parse tree (50%)

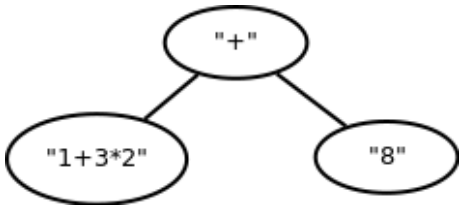
Un *parse tree* o arbol de parseo es un arbol (en este caso un arbol binario) que corresponde a una representación de todas las partes de un lenguaje. Consideremos el lenguaje de la aritmetica, el cual contiene sumas ($+$) y multiplicaciones ($*$), podemos construir la expresión $1 + 3 * 2 + 8$. Recordemos que el orden en que se deben realizar las operaciones (según las reglas de procedencia de la aritmentica) es el siguiente: "$(1 + (3 * 2)) + 8$". A partir de este orden, podemos construir un arbol para representar dicha operación y orden. El arbol seria el siguiente:



Este arbol es facil construir recursivamente a partir de una expresión matematica como "$1 + 3 * 2 + 8$". Por simplicidad, se consideraran dos recorridos al arbol: (1) expandir las sumas y (2) expandir las multiplicaciones. Consideremos de primero la primera pasada. Se comienza con la expresión entera colocada en un arbol con un solo nodo (es un arbol que contiene `strings`):



Luego se selecciona la suma de *menor* procedencia (en otras palabras, la suma más a la derecha) y se divide la operación en ese punto. Por lo cual, a partir de la operación "$1 + 3 * 2 + 8$", obtenemos "$1 + 3 * 2$", "+" y "8". Luego modificamos el nodo actual de tal forma que la operación queda en el nodo central y las sub-operaciones a la derecha e izquierda quedan como nodos hijos. El arbol ahora se ve asi:



Luego, se ejecuta esta operación recursivamente en cada uno de los nodos nuevos. Tomar en cuenta que al llegar a un nodo "final", como "8", este nodo ya no se dividira en más nodos debido a que no tiene una suma. Esto indica que la recursión debe parar ya que no hay más nodos que expandir.

Luego de haber expandido las sumas del arbol, se puede utilizar el mismo metodo para expandir las multiplicaciones del arbol aplicando dicha operación en cada uno de los nodos del arbol nuevo.

Tarea:

1. Crear una clase llamada "ParseTree" que hereda de la clase "BinaryTree" especializando su tipo generico a `string`

2. definir los metodos “ExpandirSumas”, “ExpandirMultiplicaciones” y “Expandir” en la clase “ParseTree” que expandan los nodos de dicho arbol como se menciono anteriormente. El metodo “Expandir” simplemente aplica “ExpandirSumas” seguido de “ExpandirMultiplicaciones”. Si lo desea, puede implementar toda la expansión en un solo metodo.

Metodo *Evaluar* (30%)

Luego de construir un arbol de parseo, es facil calcular cual seria el resultado de la operación expresada por el arbol de parseo utilizando la función *Reduce*. Como recordatorio, *Reduce* toma dos parametros: (1) una función que indica como combinar el valor acutal del arbol con el derecho y el izquierdo (función conocida como *reducer*) (2) el valor que se le pasara al *reducer* en caso que el lado derecho o izquierdo sea nulo.

Para calcular el resultado, se definira un *reducer* que lleva a cabo el trabajo. Como primer paso, definir un metodo en la clase “ParseTree” llamado “ReducerAritmetico” con tipo “`ReducerAritmetico : int` $\otimes$ `int` $\otimes$ `string` $\rightarrow$ `int`”. Este metodo debe hacer lo siguiente:

1. Si el tercer parametro (el valor) es “+”, retornar: izquierdo + derecho
2. Si el tercer parametro (el valor) es “∗”, retornar: izquierdo * derecho
3. De lo contrario, convertir el valor a entero utilizando la funcion *Parse* de la clase *int*: `int.Parse(valor)`. Retornar dicho valor.

A partir de esto, definir el metodo evaluar, el cual simplemente debe llamar al metodo *Reduce* utilizando cero (0) como valor inicial y “ReducerAritmetico” como su *reducer* y retornar dicho valor.

Unit tests (20%)

En el proyecto “ParseTreeTests”, definir un unit test que verifique que un arbol de parseo inicializado con una expresión aritmetica arbitraria, puede evaluarla correctamente llamando primero al metodo “Expandir” para obtener el parse tree correspondiente y luego llamando al metodo “Evaluar”.