Кравцов Дмитрий Александрович

КОМПОНЕНТ "ВЫСКАЗЫВАНИЕ" ОБУЧАЮЩЕЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ ПО АЛГЕБРЕ

Адрес статьи: www.gramota.net/materials/1/2012/4/39.html

Источник

Альманах современной науки и образования

Тамбов: Грамота, 2012. № 4 (59). С. 130-132. ISSN 1993-5552.

Адрес журнала: www.gramota.net/editions/1.html

Содержание данного номера журнала: www.gramota.net/materials/1/2012/4/

УДК 377+004:[004.02+004.588]
Технические науки

Дмитрий Александрович Кравцов
Сибирская автомобильно-дорожная академия

КОМПОНЕНТ «ВЫСКАЗЫВАНИЕ» ОБУЧАЮЩЕЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ ПО АЛГЕБРЕ©

Одной из важнейших задач при проектировании и создании любой обучающей информационной системы является формализация объектов предметной области. В компьютерной обучающей системе по алгебре ключевым понятием является *математическое выражение*. В более широком понятии мы используем термин «*математическое высказывание»*. Под высказыванием в целом мы понимаем *единицу сообщения, которая обладает смысловой целостностью в конкретной предметной области*. Это могут быть алгебраические выражения, химические уравнения, логические высказывания, формулы и т.д. Высказывание в математике - это очень широкое понятие. Практически все, с чем мы имеем дело в математике - это набор математических выражений. Любые примеры, формулы, дроби, уравнения состоят из математических выражений. Очевидно, что подобные высказывания в рамках компьютерной обучающей системы должны быть представлены определенным образом в памяти компьютера для удобной работы с ними программно и использования в различных областях обучения.

На наш взгляд, наиболее универсальным и подходящим форматом описания высказываний является *древовидная структура* или *дерево* - связный ациклический граф, то есть граф, не содержащий циклов, между любой парой вершин которого существует ровно один путь [2] . На Рисунке 1 представлен пример древовидной структуры, демонстрирующий возможную иерархическую организацию математического высказывания.

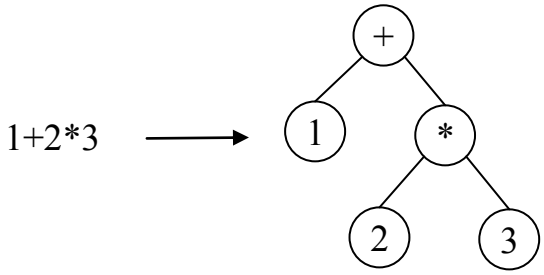

Рис. 1. *Пример разбора математического высказывания в дерево*

Существует множество различных способов представления деревьев. Наиболее общий способ представления изображает узлы как записи, расположенные в динамически выделяемой памяти с указателями на своих потомков, предков (или и тех и других). Подобный подход мы используем при построении так называемого «*дерева высказываний*» - универсальный шаблон проектирования для работы с различного рода высказываниями, которые могут использоваться в компьютерных обучающих программах. Нами разработана и реализована объектно-ориентированная модель дерева высказываний - набор базовых классов и интерфейсов, которые описывают функциональность универсальной древовидной структуры для работы с математическими выражениями. Каждый узел в подобной структуре реализует базовый интерфейс *IMathStatement*, который содержит такие основные операции как: вставка нового элемента в определённую позицию; вставка поддерева; добавление ветви дерева (называется прививкой); нахождение корневого элемента для любого узла; нахождение наименьшего общего предка двух вершин; перебор всех элементов дерева; перебор элементов ветви дерева; поиск элемента; удаление ветви дерева (называется обрезкой); удаление поддерева; удаление элемента (Рисунок 2). Кроме универсальных методов для работы со структурой дерева математические высказывания могут содержать такие специфические методы как *Simplify* (упростить) и *Calculate* (вычислить). Общий принцип работы каждого методы следующий: когда для некоторого узла дерева вызывается метод - он в свою очередь вызывает аналогичный метод нижестоящих узлов, таким образом, может осуществляться проход снизу вверх или сверху вниз по узлам дерева, если это функция, то она может опираться на результат вызова аналогичных функций нижестоящих узлов. По такому принципу, например, работает функция *Calculate* - результат вычисления текущего узла опирается на результаты вычисления дочерних узлов.

Структура дерева выражений не должна зависеть от логики её обработки. Условно-постоянное множество объектов, описывающее дерево высказываний, не должна зависеть от специализированных функций предметной области. Если, например, рассматривать алгоритм упрощения математического выражения, то при его изменении не должна затрагиваться исходная структура дерева и его составляющих, так как, с одной стороны, эта структура может использоваться независимо в разных модулях и подсистемах обучения, а, с другой стороны, алгоритм упрощения может разрабатываться программистами, которые не имеют возможности модифицировать исходный код объектов дерева. Очевидно, что все потенциальные операции над

деревом выражений невозможно предусмотреть и включить в структуру дерева высказываний. Поэтому одно из требований к компоненту - это возможность определить новую операцию, не изменяя классы объектов. Для решения этой задачи мы используем механизм динамического обхода дерева с целью его обработки. В настоящий момент реализовано два подхода: *предупорядоченный* или обход в *прямом порядке* - каждый узел-предок просматривается прежде его потомков и *поступорядоченный* обход или обход в *обратном порядке* - просматриваются сначала потомки, а потом предки. Для реализации обхода динамически созданного дерева используется шаблон проектирования «Посетитель» [1]. Диаграмма классов для нашей ситуации представлена на Рисунке 3.

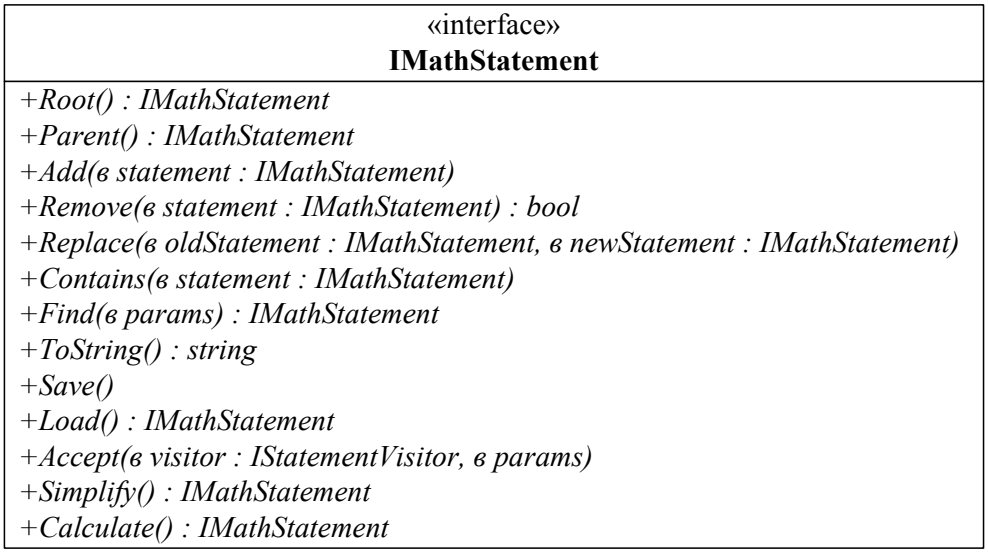

Рис. 2. *Интерфейс IMathStatement*

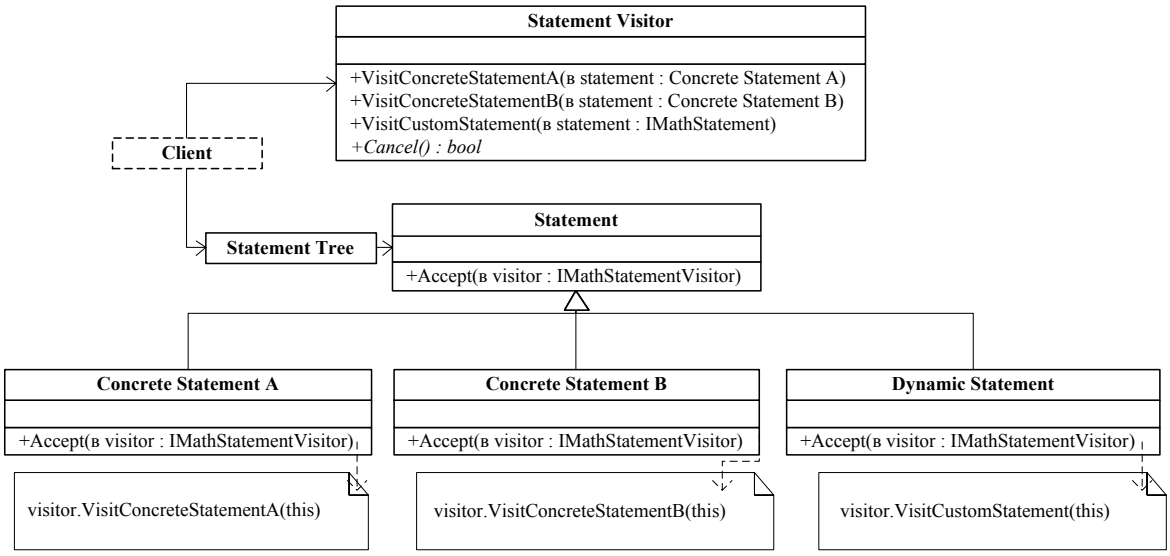

Рис. 3. *Использование шаблона «Посетитель» для обхода дерева высказываний*

Для поддержки данного шаблона в интерфейсе *IMathStatement* существует метод *Accept*, который в качестве параметров принимает конкретного посетителя, реализующего интерфейс *IMathStatementVisitor*, а также параметры обхода дерева, которые в том числе задают прямой или обратный порядок обхода. Таким образом, когда определенное высказывание в дереве принимает некоторого посетителя, оно вызывает метод посетителя, который соответствует типу текущего высказывания, посетитель в свою очередь содержит всю логику обработки посещения конкретного высказывания. В данном подходе есть одно «слабое место», на которое следует обратить внимание. Интерфейс *IMathStatementVisitor* содержит определенный набор методов, соответствующих каждому типу высказываний, но типы высказываний теоретически могут быть динамическими и соответствующие методы в посетителе могут отсутствовать. В этом случае интерфейс *IMathStatementVisitor* содержит общий метод *VisitCustomStatement*, который может быть использован высказыванием для взаимодействия с посетителем. В этом случае посетитель самостоятельно принимает решение о

том, какое это высказывание и как его обрабатывать. Использование шаблона проектирования «Посетитель» имеет целый ряд достоинств:

- упрощается добавление новых операций;
- объединяет родственные операции в классе *Visitor*;
- экземпляр *Visitor* может иметь в себе состояние (например, общую сумму) и накапливать его по ходу обхода дерева высказываний;
- при изменении посетителя нет необходимости изменять обслуживаемые классы.

Кроме интерфейсов дерева высказывания имеется базовая реализация нескольких видов высказываний в виде предопределенного набора классов. Абстрактный класс *StatementContainer* представляет собой реализацию узла-контейнера, используется, когда элемент дерева высказываний содержит коллекцию дочерних узлов как на Рисунке 4.

Рис. 4. *Высказывание - контейнер*

Класс *StatementLeaf* является листовым в дереве высказываний, как правило, он содержит определенное значение - произвольный текст, неизвестное или число. Подобный элемент дерева не может иметь дочерних элементов.

Дерево высказываний легко проецируется в *XML* формат, который обладает многими преимуществами: позволяет стандартизировать вид файлов-данных, используемых компьютерными программами, в виде текста, понятного человеку; представляет собой простой текст, свободный от лицензирования и каких-либо ограничений; не зависит от платформы и многими другими [2]. Интерфейс *IMathStatement* содержит методы *Save* и *Load*, которые предназначены для работы с форматом *XML*. На Рисунке 5 представлена формула, дерево высказываний и его *XML* представление.

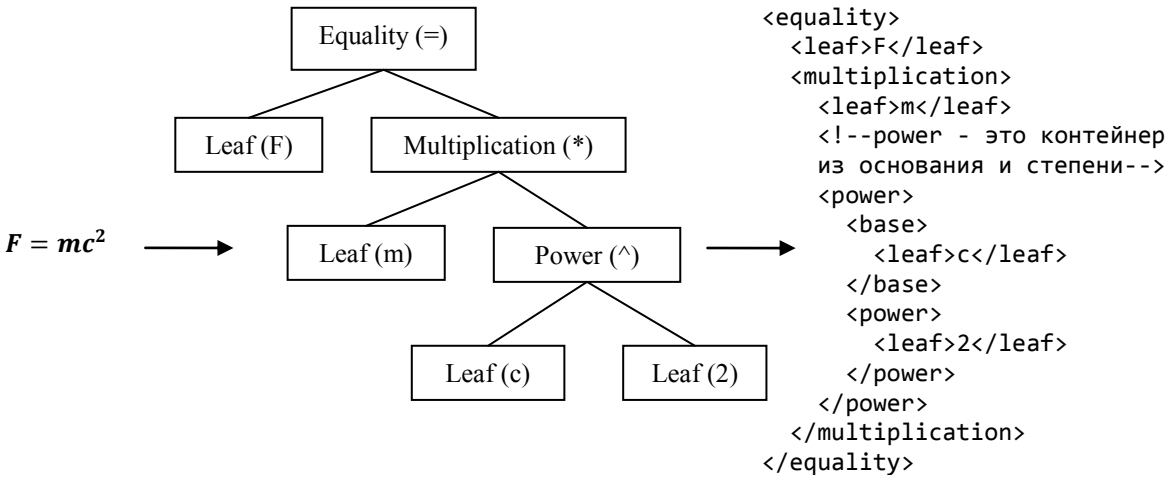

Рис. 5. *Пример представления формулы в разных форматах*

Таким образом, используя вышеописанный подход можно единым образом организовать работу с разного рода математическими выражениями, которые могут быть представлены в виде дерева высказываний для последующей обработки.

Список литературы

1. **Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж.** Приемы объектно-ориентированного проектирования. Паттерны проектирования. СПб.: Питер, 2007. 366 с.
2. **Кнут Д. Э.** Искусство программирования. 3-е изд. М.: Вильямс, 2000. Т. 2. Основные алгоритмы. 832 с.
3. **Хантер Д.** XML: базовый курс. М.: Вильямс, 2009. 1344 с.