Интерактивный интерпретатор математических выражений на JAVA.

Сергей Подкорытов.

22 марта 2015 г.

Цель данного задания: написать интерактивный интерпретатор математических выражений в префиксной записи. Перед тем как приступать к выполнению первого пункта, прочитайте задание до конца, чтобы не пришлось переписывать один тот же код несколько раз.

Итак, пусть выражение в префиксной форме следует следующему правилу:

expression :== double |
(expression) |
variableName |
operator expression expression

Где:

double — число в десятичной записи;

variableName — имя переменной, состоящее из латинских букв без пробелов;

operator — один из четырёх арифметический операторов $+, -, *, /$.

1. Напишите класс Tokenizer, разбивающий строку на список токенов для последующего разбора. Для описания типов токенов стоит использовать Enum, а для разбора строки — регулярными выражениями (классы Pattern и Matcher).

2. Напишите класс, реализующий разбор списка токенов в дерево. Для передвижения по списку токенов стоит использовать итератор. Внимательно продумайте общий интерфейс и структуру классов, представляющих узлы дерева. Полученное дерево должно уметь максимально вычислить себя, до тех пор пока не получится число или выражение, содержащее неопределенные переменные (см. пример в конце).

3. Напишите интерактивный интерпретатор. Помимо непосредственного вычисления выражений, интерпретатор должен поддерживать следующий команды:

- set variableName expression

Сопоставляет выражение в соответствующей переменной. Обратите внимание, что в переменной нужно хранить не только константы, а полноценные выражения.

- print variableName

  Выводит значение выражения, хранящегося в переменной, если известны все переменные, в противном случае вывод само выражение.

В случае ввода неправильного выражения интерпретатор должен сообщить пользователю о том, что именно помешало разбору выражения.

Пример работы подобного интерпретатора:

```
>+ 1 2
3
>+ (1 2)
Unexpected token. Expecting ).
>+ x - 3 2
+ x 1
>set x 3
x = 3
>set x
Unexpected end of input.
>set 3
Expecting variable name.
>print x
x = 3
>set y * x x
y = * x x
>print y
9
>print z
z is not initialised.
>set z / a b
z = / ab
>print z
* a b
>set a 4
a = 4
>print z
/ 4 b
```