Harvey Mudd College
Computer Science 65
Fall 2008

Assignment 8
Unicalc, Part II
Due. 11:59 p.m., Wed., 12 November 2008

Construct in Java a parser for free-form Unicalc expressions. Use it to create a user-friendly **CommandLineInterface** class (containing a **main** method). The input grammar is shown below. Each expression must be on a single line of input. The exclamation mark ! is used to designate a special command. Otherwise the expression is a Unicalc expression to be evaluated. The parser should call the evaluator that you constructed in Part I and produce the latter output on the command line. **Note that whitespace is implicitly allowed between major syntactic entities, but not within identifiers, or numerals**.

Production	Meaning
L → ! M | S	A **line L** is either a command (if preceded by !) **M** or an expression **S** to be evaluated.
S → T { { + | - } T }*	An **expression S** is a **term T**, followed by 0 or more + (representing addition) or – (representing subtraction) operators and accompanying terms. Grouping is to the left.
T → U { { <space> | / } U }*	A **term** T is a unit expression **U**, followed by 0 or more additional unit expressions separated by space (representing multiplication) or a / (representing division). Grouping is to the left.
U → P | P ^ I	A **unit expression U** is a **primary expression P**, optionally followed by a ^ (representing exponentiation) and an **integer I**.
P → C | ID | '(' S ')'	A **primary expression P** is a coefficient **C**, an identifier ID, or a parenthesized expression.
I → { + | - | λ} D D*	An **integer numeral I** is one or more digits, possibly preceded by a + or -. λ is the empty string.
D → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9	**D** represents a **digit**.
ID → {Letter | _ }{Letter | _ | D}*	An **identifier I** is a letter or underscore, followed by any number of letters, underscores, or digits.
C	**C** represents a numeric **coefficient**, which can be any unsigned Java floating numeral.
Letter → a | b | c | … | z | A | B | C | … | Z	**L** represents a **letter**.
M → add ID S | remove ID | set ID S	A **command M** is any one of **add**, **remove**, or **set** (see Part I).

Examples, showing input line and both parser and evaluator output:

```
line: 1 foot/hour
parse: (/ (* 1.0 foot) hour)
normalized quantity: 8.46651371E-5 (meter)/(second)

line: 2 mph / (foot/sec)
parse: (/ (* 2.0 mph) (/ foot sec))
normalized quantity: 2.9333333333333336

line: 1 (foot pound_force) /  (newton meter)
parse: (/ (* 1.0 (* foot pound_force)) (* newton meter))
normalized quantity: 1.355793454465969

line: 3.14
parse: 3.14
normalized quantity: 3.14

line: 3.14^2
parse: (* 3.14 3.14)
normalized quantity: 9.8596

line: !set x 10 acre
parse: (set x (* 10.0 acre))
Set x to 40467.102047438835 (meter meter)

line: x
parse: x
normalized quantity: 40467.102047438835 (meter meter)

line: x / yard^2
parse: (/ x (* yard yard))
normalized quantity: 48399.99999999999

line: 3.456 (joule/coulomb) / volt
parse: (/ (* 3.456 (/ joule coulomb)) volt)
normalized quantity: 3.456

line: !add potrzebie 2.263348517438173216473 mm
parse: (add potrzebie (* 2.2633485174381733 mm))
Defined symbol potrzebie to be: 2.2633485174381733
(millimeter)

line: 7 furlong / fortnight
parse: (/ (* 7.0 furlong) fortnight)
normalized quantity: 0.001164145635125 (meter)/(second)

line: !remove potrzebie
parse: (remove potrzebie)
Removed symbol potrzebie was: (* 2.2633485174381733
                               (/ (*  millimeter)(* )))
```