```
 1: // $Id: jrpn.java,v 1.23 2014-04-20 17:56:29-07 - - $
 2:
 3: import java.util.Scanner;
 4: import static java.lang.System.*;
 5:
 6: class jrpn {
 7:    static int exit_status = 0;
 8:    static final int EMPTY = -1;
 9:    static final int SIZE = 16;
10:    static class stack_t {
11:       int top = EMPTY;
12:       double[] numbers = new double[SIZE];
13:    }
14:
15:    static void error (String format, Object... args) {
16:       out.flush();
17:       err.printf (format, args);
18:       err.flush();
19:       exit_status = 1;
20:    }
21:
22:    static void bad_operator (String oper) {
23:       error ("\"%s\": invalid operator%n", oper);
24:    }
25:
26:    static void push (stack_t stack, double number) {
27:       if (stack.top >= SIZE - 1) {
28:          out.printf ("%s: stack overflow%n", number);
29:       }else {
30:          stack.numbers[++stack.top] = number;
31:       }
32:    }
33:
34:    static void do_binop (stack_t stack, char oper) {
35:       if (stack.top < 1) {
36:          out.printf ("'%s': stack underflow", oper);
37:       }else {
38:          double right = stack.numbers[stack.top--];
39:          double left = stack.numbers[stack.top--];
40:          switch (oper) {
41:             case '+': push (stack, left + right); break;
42:             case '-': push (stack, left - right); break;
43:             case '*': push (stack, left * right); break;
44:             case '/': push (stack, left / right); break;
45:          }
46:       }
47:    }
48:
49:
50:    static void do_print (stack_t stack) {
51:       if (stack.top == EMPTY) {
52:          out.printf ("stack is empty%n");
53:       }else {
54:          for (int pos = 0; pos <= stack.top; ++pos) {
```

```
55:             out.printf ("%s%n", stack.numbers[pos]);
56:          }
57:       }
58:    }
59:
60:    static void do_clear (stack_t stack) {
61:       stack.top = EMPTY;
62:    }
63:
64:    static void do_operator (stack_t stack, String oper) {
65:       switch (oper.charAt(0)) {
66:          case '+': do_binop (stack, '+'); break;
67:          case '-': do_binop (stack, '-'); break;
68:          case '*': do_binop (stack, '*'); break;
69:          case '/': do_binop (stack, '/'); break;
70:          case ';': do_print (stack);      break;
71:          case '@': do_clear (stack);      break;
72:          default : bad_operator (oper);   break;
73:       }
74:    }
75:
76:    static String argv_0() {
77:       String jarname = getProperty ("java.class.path");
78:       if (jarname.equals (".")) jarname = "jrpn";
79:       return jarname.substring (jarname.lastIndexOf ("/") + 1);
80:    }
81:
82:
83:    public static void main (String[] args) {
84:       if (args.length != 0) {
85:          err.printf ("Usage: %s%n", argv_0());
86:          exit (1);
87:       }
88:       Scanner stdin = new Scanner (in);
89:       stack_t stack = new stack_t();
90:       while (stdin.hasNext()) {
91:          String token = stdin.next();
92:          if (token.startsWith("#")) {
93:             stdin.nextLine();
94:             continue;
95:          }
96:          try {
97:             double number = Double.parseDouble (token);
98:             push (stack, number);
99:          }catch (NumberFormatException error) {
100:            if (token.length() != 1) {
101:               bad_operator (token);
102:            }else {
103:               do_operator (stack, token);
104:            }
105:         }
106:      }
107:      exit (exit_status);
108:   }
```

```
109: }
```

```
  1: ::::::::::::::::
  2: ../.score/test1.rpn
  3: ::::::::::::::::
  4:      1  # $Id: test1.rpn,v 1.1 2013-09-25 13:09:38-07 - - $
  5:      2  # tests for simple operators
  6:      3  # Note that # starts a comment to end of line.
  7:      4  34 .3 88 ; # should print 3 numbers
  8:      5  + + ; # should print one sum
  9:      6  8 3 * 4 7 * + ; # should print one sum
 10:      7  3 10 - ; # should print a negative number
 11:      8  4 9 / ; #fraction
 12:      9  7 0 / ; # infinity
 13:     10  1e1000000 ; # infinity
 14: ::::::::::::::::
 15: jtest1.output
 16: ::::::::::::::::
 17:      1  34.0
 18:      2  0.3
 19:      3  88.0
 20:      4  122.3
 21:      5  122.3
 22:      6  52.0
 23:      7  122.3
 24:      8  52.0
 25:      9  -7.0
 26:     10  122.3
 27:     11  52.0
 28:     12  -7.0
 29:     13  0.4444444444444444
 30:     14  122.3
 31:     15  52.0
 32:     16  -7.0
 33:     17  0.4444444444444444
 34:     18  Infinity
 35:     19  122.3
 36:     20  52.0
 37:     21  -7.0
 38:     22  0.4444444444444444
 39:     23  Infinity
 40:     24  Infinity
 41: ::::::::::::::::
 42: jtest1.status
 43: ::::::::::::::::
 44:      1  STATUS = 0
```

```
  1: ::::::::::::::::
  2: ../.score/test2.rpn
  3: ::::::::::::::::
  4:      1  # $Id: test2.rpn,v 1.1 2013-09-25 13:09:38-07 - - $
  5:      2  # test for generation of errors
  6:      3  3 + ; # stack underflow error
  7:      4  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 #stack overflow
  8:      5  error bad operator
  9: ::::::::::::::::
 10: jtest2.output
 11: ::::::::::::::::
 12:      1  '+': stack underflow3.0
 13:      2  1.0: stack overflow
 14:      3  1.0: stack overflow
 15:      4  1.0: stack overflow
 16:      5  1.0: stack overflow
 17:      6  1.0: stack overflow
 18:      7  "error": invalid operator
 19:      8  "bad": invalid operator
 20:      9  "operator": invalid operator
 21: ::::::::::::::::
 22: jtest2.status
 23: ::::::::::::::::
 24:      1  STATUS = 1
```

```
  1: ::::::::::::::::
  2: ../.score/test3.rpn
  3: ::::::::::::::::
  4:      1  # $Id: test3.rpn,v 1.1 2013-09-25 13:09:38-07 - - $
  5:      2  # tests for simple operators
  6:      3  # Note that # starts a comment to end of line.
  7:      4  34 .3 88 ;
  8:      5  + + ; @ # should print one sum
  9:      6  8 3 * 4 7 * + ; @ # should print one sum
 10:      7  3 10 - ; @ # should print a negative number
 11:      8  4 9 / ; @ #fraction
 12:      9  7 0 / ; @ # infinity
 13:     10  1e1000000 ; @ # infinity
 14: ::::::::::::::::
 15: jtest3.output
 16: ::::::::::::::::
 17:      1  34.0
 18:      2  0.3
 19:      3  88.0
 20:      4  122.3
 21:      5  52.0
 22:      6  -7.0
 23:      7  0.4444444444444444
 24:      8  Infinity
 25:      9  Infinity
 26: ::::::::::::::::
 27: jtest3.status
 28: ::::::::::::::::
 29:      1  STATUS = 0
```

```
  1: ::::::::::::::::
  2: ../.score/test1.rpn
  3: ::::::::::::::::
  4:      1  # $Id: test1.rpn,v 1.1 2013-09-25 13:09:38-07 - - $
  5:      2  # tests for simple operators
  6:      3  # Note that # starts a comment to end of line.
  7:      4  34 .3 88 ; # should print 3 numbers
  8:      5  + + ; # should print one sum
  9:      6  8 3 * 4 7 * + ; # should print one sum
 10:      7  3 10 - ; # should print a negative number
 11:      8  4 9 / ; #fraction
 12:      9  7 0 / ; # infinity
 13:     10  1e1000000 ; # infinity
 14: ::::::::::::::::
 15: jtest1.output
 16: ::::::::::::::::
 17:      1  34.0
 18:      2  0.3
 19:      3  88.0
 20:      4  122.3
 21:      5  122.3
 22:      6  52.0
 23:      7  122.3
 24:      8  52.0
 25:      9  -7.0
 26:     10  122.3
 27:     11  52.0
 28:     12  -7.0
 29:     13  0.4444444444444444
 30:     14  122.3
 31:     15  52.0
 32:     16  -7.0
 33:     17  0.4444444444444444
 34:     18  Infinity
 35:     19  122.3
 36:     20  52.0
 37:     21  -7.0
 38:     22  0.4444444444444444
 39:     23  Infinity
 40:     24  Infinity
 41: ::::::::::::::::
 42: jtest1.status
 43: ::::::::::::::::
 44:      1  STATUS = 0
```

```
  1: ::::::::::::::::
  2: ../.score/test2.rpn
  3: ::::::::::::::::
  4:      1  # $Id: test2.rpn,v 1.1 2013-09-25 13:09:38-07 - - $
  5:      2  # test for generation of errors
  6:      3  3 + ; # stack underflow error
  7:      4  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 #stack overflow
  8:      5  error bad operator
  9: ::::::::::::::::
 10: jtest2.output
 11: ::::::::::::::::
 12:      1  '+': stack underflow3.0
 13:      2  1.0: stack overflow
 14:      3  1.0: stack overflow
 15:      4  1.0: stack overflow
 16:      5  1.0: stack overflow
 17:      6  1.0: stack overflow
 18:      7  "error": invalid operator
 19:      8  "bad": invalid operator
 20:      9  "operator": invalid operator
 21: ::::::::::::::::
 22: jtest2.status
 23: ::::::::::::::::
 24:      1  STATUS = 1
```

```
  1: ::::::::::::::::
  2: ../.score/test3.rpn
  3: ::::::::::::::::
  4:      1  # $Id: test3.rpn,v 1.1 2013-09-25 13:09:38-07 - - $
  5:      2  # tests for simple operators
  6:      3  # Note that # starts a comment to end of line.
  7:      4  34 .3 88 ;
  8:      5  + + ; @ # should print one sum
  9:      6  8 3 * 4 7 * + ; @ # should print one sum
 10:      7  3 10 - ; @ # should print a negative number
 11:      8  4 9 / ; @ #fraction
 12:      9  7 0 / ; @ # infinity
 13:     10  1e1000000 ; @ # infinity
 14: ::::::::::::::::
 15: jtest3.output
 16: ::::::::::::::::
 17:      1  34.0
 18:      2  0.3
 19:      3  88.0
 20:      4  122.3
 21:      5  52.0
 22:      6  -7.0
 23:      7  0.4444444444444444
 24:      8  Infinity
 25:      9  Infinity
 26: ::::::::::::::::
 27: jtest3.status
 28: ::::::::::::::::
 29:      1  STATUS = 0
```

```
  1: ::::::::::::::::
  2: ../.score/test*.rpn
  3: ::::::::::::::::
  4: ::::::::::::::::
  5: jtest*.output
  6: ::::::::::::::::
  7: ::::::::::::::::
  8: jtest*.status
  9: ::::::::::::::::
 10:       1  STATUS = 1
```

```
 1: // $Id: crpn.c,v 1.34 2014-04-20 18:35:28-07 - - $
 2:
 3: #include <assert.h>
 4: #include <libgen.h>
 5: #include <stdio.h>
 6: #include <stdlib.h>
 7:
 8: int exit_status = EXIT_SUCCESS;
 9: #define EMPTY (-1)
10: #define SIZE 16
11:
12: struct stack {
13:    int top;
14:    double numbers[SIZE];
15: };
16:
17: void bad_operator (const char *oper) {
18:    fflush (NULL);
19:    fprintf (stderr, "\"%s\": invalid operator\n", oper);
20:    fflush (NULL);
21:    exit_status = EXIT_FAILURE;
22: }
23:
24: void push (struct stack *the_stack, double number) {
25:    if(the_stack->top >= (SIZE - 1)) {
26:       printf("%.15g: stack overflow\n", number);
27:    } else {
28:       the_stack->numbers[++the_stack->top] = number;
29:    }
30: }
31:
32: void do_binop (struct stack *the_stack, char oper) {
33:    if(the_stack->top < 1) {
34:       printf("'%c': stack underflow\n", oper);
35:    } else {
36:       double right = the_stack->numbers[the_stack->top--];
37:       double left = the_stack->numbers[the_stack->top--];
38:       switch (oper) {
39:          case '+': push(the_stack, left + right); break;
40:          case '-': push(the_stack, left - right); break;
41:          case '*': push(the_stack, left * right); break;
42:          case '/': push(the_stack, left / right); break;
43:       }
44:    }
45: }
46:
47: void do_print (struct stack *the_stack) {
48:    if(the_stack->top == EMPTY) {
49:       printf("stack is empty\n");
50:    } else {
51:       for(int pos = 0; pos <= the_stack->top; ++pos) {
52:          printf("%.15g\n", the_stack->numbers[pos]);
53:       }
54:    }
```

```
 55: }
 56:
 57: void do_clear (struct stack *the_stack) {
 58:    the_stack->top = EMPTY;
 59: }
 60:
 61: void do_operator (struct stack *the_stack, const char *oper) {
 62:    switch(oper[0]) {
 63:       case '+': do_binop (the_stack, '+'); break;
 64:       case '-': do_binop (the_stack, '-'); break;
 65:       case '*': do_binop (the_stack, '*'); break;
 66:       case '/': do_binop (the_stack, '/'); break;
 67:       case ';': do_print (the_stack);      break;
 68:       case '@': do_clear (the_stack);      break;
 69:       default: bad_operator (oper);        break;
 70:    }
 71: }
 72:
 73:
 74: int main (int argc, char **argv) {
 75:    if (argc != 1) {
 76:       fprintf (stderr, "Usage: %s\n", basename (argv[0]));
 77:       fflush (NULL);
 78:       exit (EXIT_FAILURE);
 79:    }
 80:    struct stack the_stack;
 81:    the_stack.top = EMPTY;
 82:    char buffer[1024];
 83:    for (;;) {
 84:       int scanrc = scanf ("%1023s", buffer);
 85:       if (scanrc == EOF) break;
 86:       assert (scanrc == 1);
 87:       if (buffer[0] == '#') {
 88:          scanrc = scanf ("%1023[^\n]", buffer);
 89:          continue;
 90:       }
 91:       char *endptr;
 92:       double number = strtod (buffer, &endptr);
 93:       if (*endptr == '\0') {
 94:          push (&the_stack, number);
 95:       }else if (buffer[1] != '\0') {
 96:          bad_operator (buffer);
 97:       }else {
 98:          do_operator (&the_stack, buffer);
 99:       }
100:    }
101:    return exit_status;
102: }
103:
```

```
  1: ::::::::::::::::
  2: ../.score/test1.rpn
  3: ::::::::::::::::
  4:      1  # $Id: test1.rpn,v 1.1 2013-09-25 13:09:38-07 - - $
  5:      2  # tests for simple operators
  6:      3  # Note that # starts a comment to end of line.
  7:      4  34 .3 88 ; # should print 3 numbers
  8:      5  + + ; # should print one sum
  9:      6  8 3 * 4 7 * + ; # should print one sum
 10:      7  3 10 - ; # should print a negative number
 11:      8  4 9 / ; #fraction
 12:      9  7 0 / ; # infinity
 13:     10  1e1000000 ; # infinity
 14: ::::::::::::::::
 15: ctest1.output
 16: ::::::::::::::::
 17:      1  34
 18:      2  0.3
 19:      3  88
 20:      4  122.3
 21:      5  122.3
 22:      6  52
 23:      7  122.3
 24:      8  52
 25:      9  -7
 26:     10  122.3
 27:     11  52
 28:     12  -7
 29:     13  0.444444444444444
 30:     14  122.3
 31:     15  52
 32:     16  -7
 33:     17  0.444444444444444
 34:     18  inf
 35:     19  122.3
 36:     20  52
 37:     21  -7
 38:     22  0.444444444444444
 39:     23  inf
 40:     24  inf
 41: ::::::::::::::::
 42: ctest1.status
 43: ::::::::::::::::
 44:      1  STATUS = 0
```