```
 1:  //$Id: jrpn.java,v 1.1 2013-10-17 15:08:09-07 - - $
 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:    public static void main (String[] args) {
 83:       if (args.length != 0) {
 84:          err.printf ("Usage: %s%n", argv_0());
 85:          exit (1);
 86:       }
 87:       Scanner stdin = new Scanner (in);
 88:       stack_t stack = new stack_t();
 89:       while (stdin.hasNext()) {
 90:          String token = stdin.next();
 91:          if (token.startsWith("#")) {
 92:             stdin.nextLine();
 93:             continue;
 94:          }
 95:          try {
 96:             double number = Double.parseDouble (token);
 97:             push (stack, number);
 98:          }catch (NumberFormatException error) {
 99:             if (token.length() != 1) {
100:                bad_operator (token);
101:             }else {
102:                do_operator (stack, token);
```

```
103:                }
104:             }
105:          }
106:          exit (exit_status);
107:       }
108: }
```

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

```
  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: // $Id: crpn.c,v 1.12 2013-10-19 16:15:50-07 - - $
 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: typedef struct stack stack;
13: struct stack {
14:    int top;
15:    double numbers[SIZE];
16: };
17:
18: void bad_operator (const char *oper) {
19:    fflush (NULL);
20:    fprintf (stderr, "oper=\"%s\"\n", oper);
21:    fflush (NULL);
22:    exit_status = EXIT_FAILURE;
23: }
24:
25: void push (stack *the_stack, double number) {
26:    if (the_stack->top >= SIZE - 1) {
27:          printf ("%f: stack overflow\n", number);
28:       }else {
29:          the_stack->numbers[++the_stack->top] = number;
30:       }
31: }
32:
33: void do_binop (stack *the_stack, char oper) {
34:    if (the_stack->top < 1) {
35:          printf ("'%d': stack underflow\n", oper);
36:       }else {
37:          double right = the_stack->numbers[the_stack->top--];
38:          double left = the_stack->numbers[the_stack->top--];
39:          switch (oper) {
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:             case '/': push (the_stack, left / right); break;
44:          }
45:       }
46: }
47:
48: void do_print (stack *the_stack) {
49:    if (the_stack->top == EMPTY) {
50:          printf ("stack is empty\n");
51:       }else {
52:          int pos = 0;
53:          for (;pos <= the_stack->top; ++pos) {
54:             printf ("%f\n", the_stack->numbers[pos]);
```

```
   55:          }
   56:       }
   57: }
   58:
   59: void do_clear (stack *the_stack) {
   60:    the_stack->top = EMPTY;
   61: }
   62:
   63: void do_operator (stack *the_stack, const char *oper) {
   64:    switch (oper[0]) {
   65:          case '+': do_binop (the_stack, '+'); break;
   66:          case '-': do_binop (the_stack, '-'); break;
   67:          case '*': do_binop (the_stack, '*'); break;
   68:          case '/': do_binop (the_stack, '/'); break;
   69:          case ';': do_print (the_stack);      break;
   70:          case '@': do_clear (the_stack);      break;
   71:          default : bad_operator (oper);   break;
   72:       }
   73: }
   74:
   75: int main (int argc, char **argv) {
   76:    if (argc != 1) {
   77:       fprintf (stderr, "Usage: %s\n", basename (argv[0]));
   78:       fflush (NULL);
   79:       exit (EXIT_FAILURE);
   80:    }
   81:    stack the_stack;
   82:    the_stack.top = EMPTY;
   83:    char buffer[1024];
   84:    for (;;) {
   85:       int scanrc = scanf ("%1023s", buffer);
   86:       if (scanrc == EOF) break;
   87:       assert (scanrc == 1);
   88:       if (buffer[0] == '#') {
   89:          scanrc = scanf ("%1023[^\n]", buffer);
   90:          continue;
   91:       }
   92:       char *endptr;
   93:       double number = strtod (buffer, &endptr);
   94:       if (*endptr == '\0') {
   95:          push (&the_stack, number);
   96:       }else if (buffer[1] != '\0') {
   97:          bad_operator (buffer);
   98:       }else {
   99:          do_operator (&the_stack, buffer);
  100:       }
  101:    }
  102:    return exit_status;
  103: }
  104:
```

```
  1: ::::::::::::::::
  2: ../.score/test1.rpn
  3: ::::::::::::::::
  4: ::::::::::::::::
  5: ctest1.output
  6: ::::::::::::::::
  7: ::::::::::::::::
  8: ctest1.status
  9: ::::::::::::::::
 10:       1  STATUS = 1
```