```
1: // $Id: jrpn.java,v 1.1 2013-10-22 20:38:39-07 - - $
 3: import java.util.Scanner;
 4: import static java.lang.System.*;
 5:
 6: class jrpn {
7:
       static int exit status = 0;
       static final int EMPTY = -1;
8:
 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();
          exit_status = 1;
19:
20:
       }
21:
22:
       static void bad_operator (String oper) {
          error ("\"%s\": invalid operator%n", oper);
23:
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:
30:
             stack.numbers[++stack.top] = number;
31:
          }
32:
       }
33:
34:
       static void do_binop (stack_t stack, char oper) {
35:
          if (stack.top < 1) {</pre>
36:
             out.printf ("'%s': stack underflow", oper);
37:
          }else {
38:
             double right = stack.numbers[stack.top--];
39:
             double left = stack.numbers[stack.top--];
             switch (oper) {
40:
41:
                case '+': push (stack, left + right); break;
                case '-': push (stack, left - right); break;
42:
                case '*': push (stack, left * right); break;
43:
                case '/': push (stack, left / right); break;
44:
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:
54:
             for (int pos = 0; pos <= stack.top; ++pos) {</pre>
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)) {
             case '+': do_binop (stack, '+'); break;
66:
             case '-': do_binop (stack, '-'); break;
67:
             case '*': do_binop (stack, '*'); break;
68:
             case '/': do_binop (stack, '/'); break;
69:
             case ';': do_print (stack);
                                              break;
70:
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");
          if (jarname.equals (".")) jarname = "jrpn";
78:
79:
          return jarname.substring (jarname.lastIndexOf ("/") + 1);
80:
       }
81:
```

```
82:
 83:
        public static void main (String[] args) {
 84:
           if (args.length != 0) {
              err.printf ("Usage: %s%n", argv_0());
 85:
 86:
              exit (1);
 87:
           }
           Scanner stdin = new Scanner (in);
 88:
           stack_t stack = new stack_t();
 89:
 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);
              }catch (NumberFormatException error) {
 99:
                  if (token.length() != 1) {
100:
101:
                     bad_operator (token);
102:
                  }else {
103:
                     do_operator (stack, token);
104:
                  }
105:
              }
106:
           exit (exit_status);
107:
108:
        }
109: }
```

/afs/cats.ucsc.edu/users/g/asilva3/private/cmps012b/lab3/
jtest*.lis

1: ::::::::::::: 2: ../.score/test*.rpn 4: :::::::::::::: 5: jtest*.output 6: :::::::::::::: 7: :::::::::::::::: 8: jtest*.status 9: ::::::::::::::: $1 \quad \text{STATUS} = 1$

10/22/13 20:56:40

10:

```
1: // $Id: crpn.c, v 1.11 2013-10-22 20:56:39-07 - - $
 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) {
       fflush (NULL);
19:
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 {the_stack->numbers[++the_stack->top] = number;}
29:
       printf("the_stack=%p, top=%d, number=%.15g\n",
30:
               the_stack, the_stack->top, number);
31: }
32:
33: void do_binop (stack *the_stack, char oper) {
34:
      if (the_stack->top < 1) {</pre>
35:
             printf("'%d': stack underflow", 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;
                case '*': push (the_stack, left * right); break;
42:
                case '/': push (the_stack, left / right); break;
43:
44:
             }
45:
       printf("the_stack=%p, top=%d, oper='%c'\n",
46:
47:
               the_stack, the_stack->top, oper);
48: }
49:
50: void do_print (stack *the_stack) {
51:
        if(the_stack->top == EMPTY) {
52:
          printf("stack is empty\n");
       }else{for(int pos = 0; pos <= the_stack->top; ++pos) {
53:
54:
                printf ("%lf\n", the_stack->numbers[pos]);
```

```
55:
              }
56:
       }
        printf("the_stack=%p, top=%d\n", the_stack, the_stack->top);
57:
58: }
59:
60: void do_clear (stack *the_stack) {
61:
       the_stack->top = EMPTY;
62:
       printf("the_stack=%p, top=%d\n", the_stack, the_stack->top);
63: }
64:
65: void do_operator (stack *the_stack, const char *oper) {
       switch(oper[0]){
66:
       case '+': do_binop(the_stack, '+'); break;
67:
       case '-': do_binop(the_stack, '-'); break;
case '*': do_binop(the_stack, '*'); break;
68:
69:
70:
       case '/': do_binop (the_stack, '/'); break;
       case ';': do_print (the_stack);
71:
72:
       case '@': do_clear (the_stack);
73:
       default : bad_operator (oper);
                                           break;
74:
       printf("the_stack=%p, top=%d, oper=\"%s\"\n",
                the_stack, the_stack->top, oper);
75:
76:
       }
77: }
```

```
78:
 79: int main (int argc, char **argv) {
        if (argc != 1) {
           fprintf (stderr, "Usage: %s\n", basename (argv[0]));
 81:
 82:
           fflush (NULL);
 83:
           exit (EXIT_FAILURE);
 84:
        }
        stack the_stack;
 85:
 86:
        the_stack.top = EMPTY;
        char buffer[1024];
 87:
 88:
        for (;;) {
 89:
           int scanrc = scanf ("%1023s", buffer);
 90:
           if (scanrc == EOF) break;
 91:
           assert (scanrc == 1);
 92:
           if (buffer[0] == '#') {
 93:
              scanrc = scanf ("%1023[^\n]", buffer);
 94:
              continue;
 95:
           }
 96:
           char *endptr;
 97:
           double number = strtod (buffer, &endptr);
           if (*endptr == ' \setminus 0') {
 98:
 99:
              push (&the_stack, number);
100:
           }else if (buffer[1] != '\0') {
101:
              bad_operator (buffer);
102:
           }else {
103:
              do_operator (&the_stack, buffer);
104:
           }
105:
        }
106:
        return exit_status;
107: }
108:
```

/afs/cats.ucsc.edu/users/g/asilva3/private/cmps012b/lab3/ctest1.lis

10/22/13 20:56:40