```java
// $Id: jrpn.java,v 1.1 2013-10-22 20:38:39-07 - - $

import java.util.Scanner;
import static java.lang.System.*;

class jrpn {
   static int exit_status = 0;
   static final int EMPTY = -1;
   static final int SIZE = 16;
   static class stack_t {
      int top = EMPTY;
      double[] numbers = new double[SIZE];
   }

   static void error (String format, Object... args) {
      out.flush();
      err.printf (format, args);
      err.flush();
      exit_status = 1;
   }

   static void bad_operator (String oper) {
      error ("\"%s\": invalid operator%n", oper);
   }

   static void push (stack_t stack, double number) {
      if (stack.top >= SIZE - 1) {
         out.printf ("%s: stack overflow%n", number);
      }else {
         stack.numbers[++stack.top] = number;
      }
   }

   static void do_binop (stack_t stack, char oper) {
      if (stack.top < 1) {
         out.printf ("'%s': stack underflow", oper);
      }else {
         double right = stack.numbers[stack.top--];
         double left = stack.numbers[stack.top--];
         switch (oper) {
            case '+': push (stack, left + right); break;
            case '-': push (stack, left - right); break;
            case '*': push (stack, left * right); break;
            case '/': push (stack, left / right); break;
         }
      }
   }


   static void do_print (stack_t stack) {
      if (stack.top == EMPTY) {
         out.printf ("stack is empty%n");
      }else {
         for (int pos = 0; pos <= stack.top; ++pos) {
            out.printf ("%s%n", stack.numbers[pos]);
         }
      }
   }

   static void do_clear (stack_t stack) {
      stack.top = EMPTY;
   }

   static void do_operator (stack_t stack, String oper) {
      switch (oper.charAt(0)) {
         case '+': do_binop (stack, '+'); break;
         case '-': do_binop (stack, '-'); break;
         case '*': do_binop (stack, '*'); break;
         case '/': do_binop (stack, '/'); break;
         case ';': do_print (stack);      break;
         case '@': do_clear (stack);      break;
         default : bad_operator (oper);   break;
      }
   }

   static String argv_0() {
      String jarname = getProperty ("java.class.path");
      if (jarname.equals (".")) jarname = "jrpn";
      return jarname.substring (jarname.lastIndexOf ("/") + 1);
   }

```

```java

   public static void main (String[] args) {
      if (args.length != 0) {
         err.printf ("Usage: %s%n", argv_0());
         exit (1);
      }
      Scanner stdin = new Scanner (in);
      stack_t stack = new stack_t();
      while (stdin.hasNext()) {
         String token = stdin.next();
         if (token.startsWith("#")) {
            stdin.nextLine();
            continue;
         }
         try {
            double number = Double.parseDouble (token);
            push (stack, number);
         }catch (NumberFormatException error) {
            if (token.length() != 1) {
               bad_operator (token);
            }else {
               do_operator (stack, token);
            }
         }
      }
      exit (exit_status);
   }
}
```

```
::::::::::::::::
../.score/test*.rpn
::::::::::::::::
::::::::::::::::
jtest*.output
::::::::::::::::
::::::::::::::::
jtest*.status
::::::::::::::::
     1  STATUS = 1
```

```c
// $Id: crpn.c,v 1.11 2013-10-22 20:56:39-07 - - $

#include <assert.h>
#include <libgen.h>
#include <stdio.h>
#include <stdlib.h>

int exit_status = EXIT_SUCCESS;
#define EMPTY (-1)
#define SIZE 16

typedef struct stack stack;
struct stack {
   int top;
   double numbers[SIZE];
};

void bad_operator (const char *oper) {
   fflush (NULL);
   fprintf (stderr, "oper=\"%s\"\n", oper);
   fflush (NULL);
   exit_status = EXIT_FAILURE;
}

void push (stack *the_stack, double number) {
  if(the_stack->top >= SIZE - 1){
      printf("%f: stack overflow\n", number);
   }else {the_stack->numbers[++the_stack->top] = number;}
   printf("the_stack=%p, top=%d, number=%.15g\n",
           the_stack, the_stack->top, number);
}

void do_binop (stack *the_stack, char oper){
  if (the_stack->top < 1){
         printf("'%d': stack underflow", oper);
      }else{
         double right = the_stack->numbers[the_stack->top--];
         double left = the_stack->numbers[the_stack->top--];
         switch (oper) {
            case '+': push (the_stack, left + right); break;
            case '-': push (the_stack, left - right); break;
            case '*': push (the_stack, left * right); break;
            case '/': push (the_stack, left / right); break;
         }
      }
   printf("the_stack=%p, top=%d, oper='%c'\n",
           the_stack, the_stack->top, oper);
}

void do_print (stack *the_stack) {
    if(the_stack->top == EMPTY){
      printf("stack is empty\n");
   }else{for(int pos = 0; pos <= the_stack->top; ++pos) {
            printf ("%lf\n", the_stack->numbers[pos]);
         }
   }
    printf("the_stack=%p, top=%d\n", the_stack, the_stack->top);
}

void do_clear (stack *the_stack) {
   the_stack->top = EMPTY;
   printf("the_stack=%p, top=%d\n", the_stack, the_stack->top);
}

void do_operator (stack *the_stack, const char *oper) {
   switch(oper[0]){
   case '+': do_binop(the_stack, '+'); break;
   case '-': do_binop(the_stack, '-'); break;
   case '*': do_binop(the_stack, '*'); break;
   case '/': do_binop (the_stack, '/'); break;
   case ';': do_print (the_stack);      break;
   case '@': do_clear (the_stack);      break;
   default : bad_operator (oper);   break;
   printf("the_stack=%p, top=%d, oper=\"%s\"\n",
           the_stack, the_stack->top, oper);
   }
}
```

```c

int main (int argc, char **argv) {
   if (argc != 1) {
      fprintf (stderr, "Usage: %s\n", basename (argv[0]));
      fflush (NULL);
      exit (EXIT_FAILURE);
   }
   stack the_stack;
   the_stack.top = EMPTY;
   char buffer[1024];
   for (;;) {
      int scanrc = scanf ("%1023s", buffer);
      if (scanrc == EOF) break;
      assert (scanrc == 1);
      if (buffer[0] == '#') {
         scanrc = scanf ("%1023[^\n]", buffer);
         continue;
      }
      char *endptr;
      double number = strtod (buffer, &endptr);
      if (*endptr == '\0') {
         push (&the_stack, number);
      }else if (buffer[1] != '\0') {
         bad_operator (buffer);
      }else {
         do_operator (&the_stack, buffer);
      }
   }
   return exit_status;
}

```

```
::::::::::::::::
../.score/test1.rpn
::::::::::::::::
::::::::::::::::
ctest1.output
::::::::::::::::
::::::::::::::::
ctest1.status
::::::::::::::::
     1  STATUS = 1
```