```java
// $Id: jrpn.java,v 1.22 2013-10-11 19:19:01-07 - - $

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);
   }


   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/test1.rpn
::::::::::::::::::::::::::::::::
     1  # $Id: test1.rpn,v 1.2 2015-01-26 13:26:15-08 - - $
     2  # tests for simple operators
     3  # Note that # starts a comment to end of line.
     4
     5  34 .3 + 23 8 * - ; @
     6  44 0 / ; @
     7  2 4 / ; @
::::::::::::::::::::::::::::::::
jtest1.output
::::::::::::::::::::::::::::::::
     1  -149.7
     2  Infinity
     3  0.5
::::::::::::::::::::::::::::::::
jtest1.status
::::::::::::::::::::::::::::::::
     1  STATUS = 0
```

```
::::::::::::::::::::::::::::::::
../.score/test2.rpn
::::::::::::::::::::::::::::::::
     1  # $Id: test2.rpn,v 1.1 2013-09-25 13:09:38-07 - - $
     2  # test for generation of errors
     3  3 + ; # stack underflow error
     4  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 #stack overflow
     5  error bad operator
::::::::::::::::::::::::::::::::
jtest2.output
::::::::::::::::::::::::::::::::
     1  '+': stack underflow3.0
     2  1.0: stack overflow
     3  1.0: stack overflow
     4  1.0: stack overflow
     5  1.0: stack overflow
     6  1.0: stack overflow
     7  "error": invalid operator
     8  "bad": invalid operator
     9  "operator": invalid operator
::::::::::::::::::::::::::::::::
jtest2.status
::::::::::::::::::::::::::::::::
     1  STATUS = 1
```

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

```c
// $Id: crpn.c,v 1.28 2014-04-08 15:23:19-07 - - $

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

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

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 (struct stack *the_stack, double number) {
   printf ("the_stack=%p, top=%d, number=%.15g\n",
           the_stack, the_stack->top, number);
}

void do_binop (struct stack *the_stack, char oper) {
   printf ("the_stack=%p, top=%d, oper='%c'\n",
           the_stack, the_stack->top, oper);
}

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

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

void do_operator (struct stack *the_stack, const char *oper) {
   printf ("the_stack=%p, top=%d, oper=\"%s\"\n",
           the_stack, the_stack->top, oper);
}

```

```

int main (int argc, char **argv) {
   if (argc != 1) {
      fprintf (stderr, "Usage: %s\n", basename (argv[0]));
      fflush (NULL);
      exit (EXIT_FAILURE);
   }
   struct 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
::::::::::::::::::::::::::::::::
     1  # $Id: test1.rpn,v 1.2 2015-01-26 13:26:15-08 - - $
     2  # tests for simple operators
     3  # Note that # starts a comment to end of line.
     4
     5  34 .3 + 23 8 * - ; @
     6  44 0 / ; @
     7  2 4 / ; @
::::::::::::::::::::::::::::::::
ctest1.output
::::::::::::::::::::::::::::::::
     1  the_stack=0x7ffee0fb57f0, top=-1, number=34
     2  the_stack=0x7ffee0fb57f0, top=-1, number=0.3
     3  the_stack=0x7ffee0fb57f0, top=-1, oper="+"
     4  the_stack=0x7ffee0fb57f0, top=-1, number=23
     5  the_stack=0x7ffee0fb57f0, top=-1, number=8
     6  the_stack=0x7ffee0fb57f0, top=-1, oper="*"
     7  the_stack=0x7ffee0fb57f0, top=-1, oper="-"
     8  the_stack=0x7ffee0fb57f0, top=-1, oper=";"
     9  the_stack=0x7ffee0fb57f0, top=-1, oper="@"
    10  the_stack=0x7ffee0fb57f0, top=-1, number=44
    11  the_stack=0x7ffee0fb57f0, top=-1, number=0
    12  the_stack=0x7ffee0fb57f0, top=-1, oper="/"
    13  the_stack=0x7ffee0fb57f0, top=-1, oper=";"
    14  the_stack=0x7ffee0fb57f0, top=-1, oper="@"
    15  the_stack=0x7ffee0fb57f0, top=-1, number=2
    16  the_stack=0x7ffee0fb57f0, top=-1, number=4
    17  the_stack=0x7ffee0fb57f0, top=-1, oper="/"
    18  the_stack=0x7ffee0fb57f0, top=-1, oper=";"
    19  the_stack=0x7ffee0fb57f0, top=-1, oper="@"
::::::::::::::::::::::::::::::::
ctest1.status
::::::::::::::::::::::::::::::::
     1  STATUS = 0
```

```
#!/bin/sh
# $Id: mk,v 1.9 2016-01-11 21:51:08-08 - - $

cid $0 jrpn.java crpn.c
mkc jrpn.java

for test in ../.score/test*.rpn
do
   basename=${test##.*/}
   prefix=j${basename%.*}
   java -jar jrpn <$test >$prefix.output 2>&1
   echo STATUS = $? >$prefix.status
   catnv $test $prefix.output $prefix.status >$prefix.lis
   rm $prefix.output $prefix.status
done

mkc crpn.c
./crpn <../.score/test1.rpn >ctest1.output 2>&1
echo STATUS = $? >ctest1.status

catnv ../.score/test1.rpn ctest1.output ctest1.status >ctest1.lis
rm ctest1.output ctest1.status

mkpspdf -s12 Listing.ps jrpn.java* j*.lis crpn.c* c*.lis $0
```