```
-- This is the Stack example used earlier to
-- illustrate order-sorted features

module! STACK-INT {
  protecting(INT)
  [NeStack < Stack]     -- sort/subsort declaration

  op new : -> Stack
  op push : Stack Int -> NeStack
  op pop_ : NeStack -> Stack
  op top_ : NeStack -> Int

  vars S : Stack
  vars I : Int

  eq pop(push(S,I)) = S .
  eq top(push(S,I)) = I .
}
```

```
Script started on Sun Feb 20 16:26:29 2000
fleck@tornado [101]% cafeobj
                -- CafeOBJ system Version 1.4.2 --

CafeOBJ> in errstk
processing input : ./errstk.mod
-- defining module! STACK-INT........_..* done.
CafeOBJ> select STACK-INT
STACK-INT> red pop (push(pop(push(push(new,1),2)),3)) .
-- reduce in STACK-INT :
                    pop push(pop push(push(new,1),2),3)
push(new,1) : NeStack
(0.010 sec for parse, 2 rewrites(0.030 sec), 2 matches)
STACK-INT> red top (pop push(pop
                                        push(push(new,1),2),3)) .
-- reduce in STACK-INT :
             top (pop push(pop push(push(new,1),2),3))
1 : NzNat
(0.010 sec for parse, 3 rewrites(0.000 sec), 3 matches)
```

```
STACK-INT> red top (pop push(new,1)) .
-- reduce in STACK-INT : top (pop push(new,1))
top new : ?Int
(0.000 sec for parse, 1 rewrites(0.000 sec), 1 matches)
STACK-INT> red pop new .
-- reduce in STACK-INT : pop new
pop new : ?Stack
(0.000 sec for parse, 0 rewrites(0.000 sec), 0 matches)
STACK-INT> red pop (pop(push(push(new,1),2))) .
-- reduce in STACK-INT : pop (pop push(push(new,1),2))
new : Stack
(0.010 sec for parse, 2 rewrites(0.010 sec), 2 matches)
STACK-INT> q
[Leaving CafeOBJ]
```