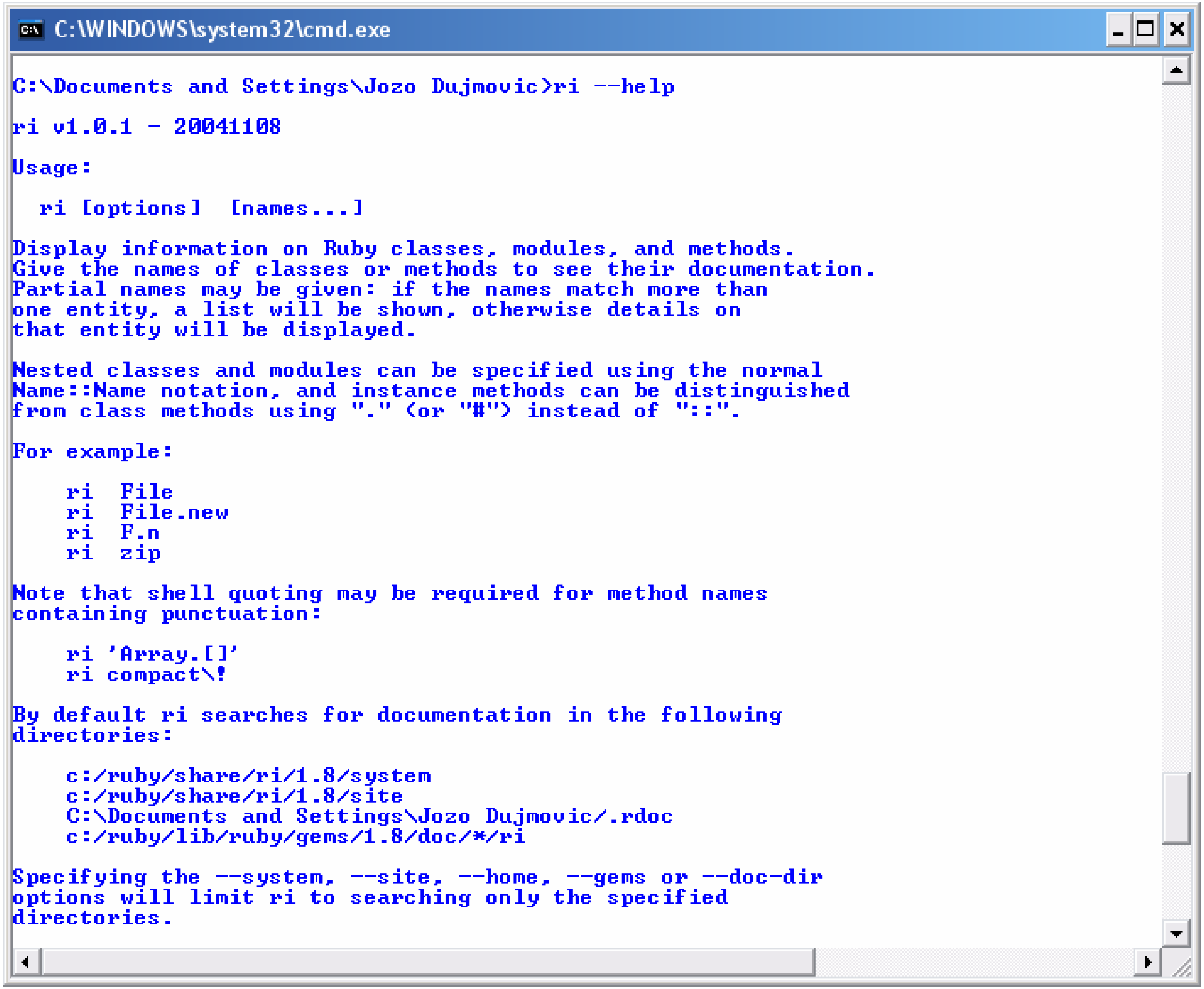
RUBY

#3 Methods and Classes

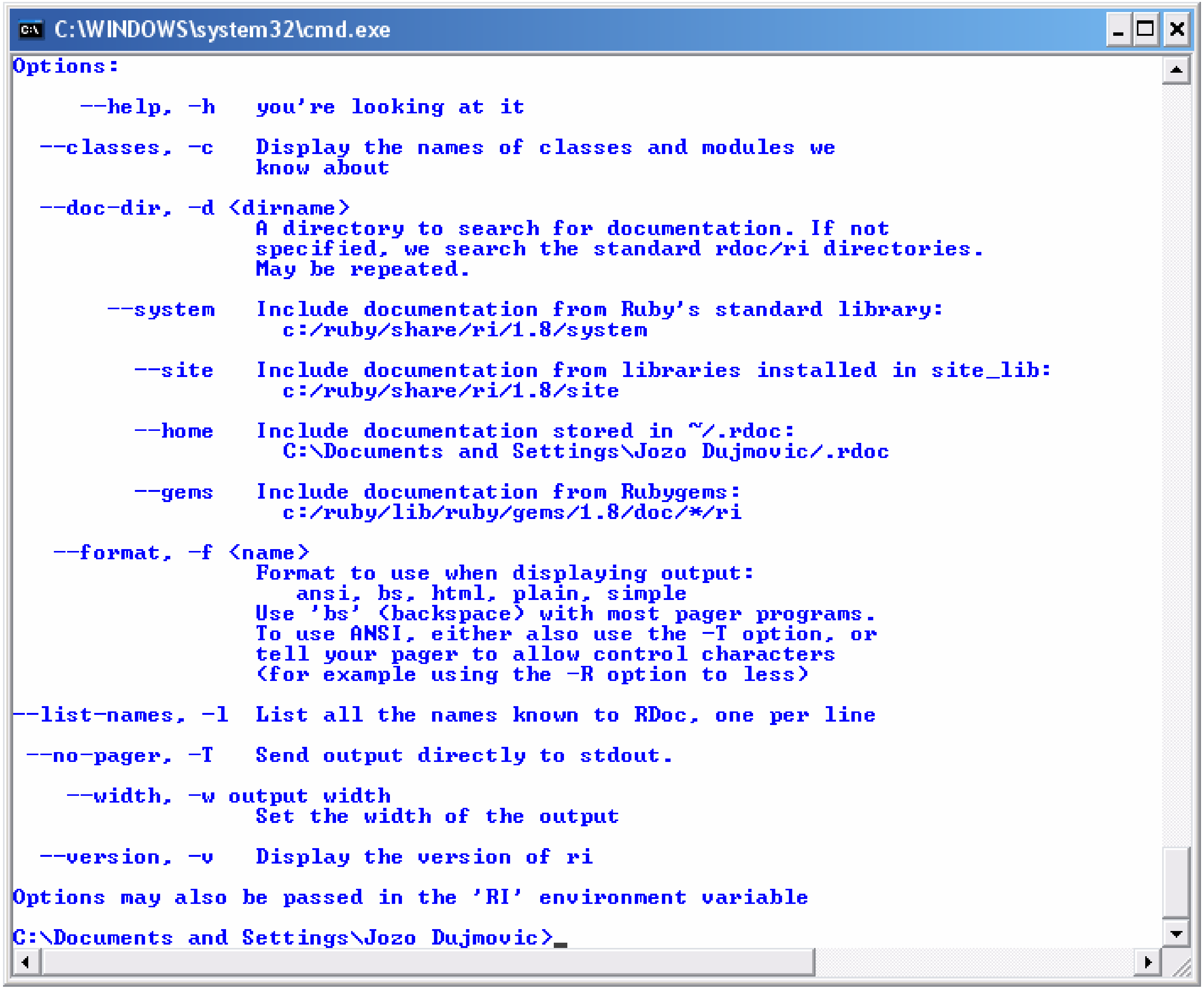
Dr. Jozo Dujmović

Ruby is not a small language

- Methods belong to libraries
- Number of various standard libraries that come with Ruby 1.8 = **98**
- Number of methods that come with Ruby is currently **9000++**
- These methods are documented in RDoc, available at http://www.ruby-doc.org
- **ri** is a local command line viewer of RDoc

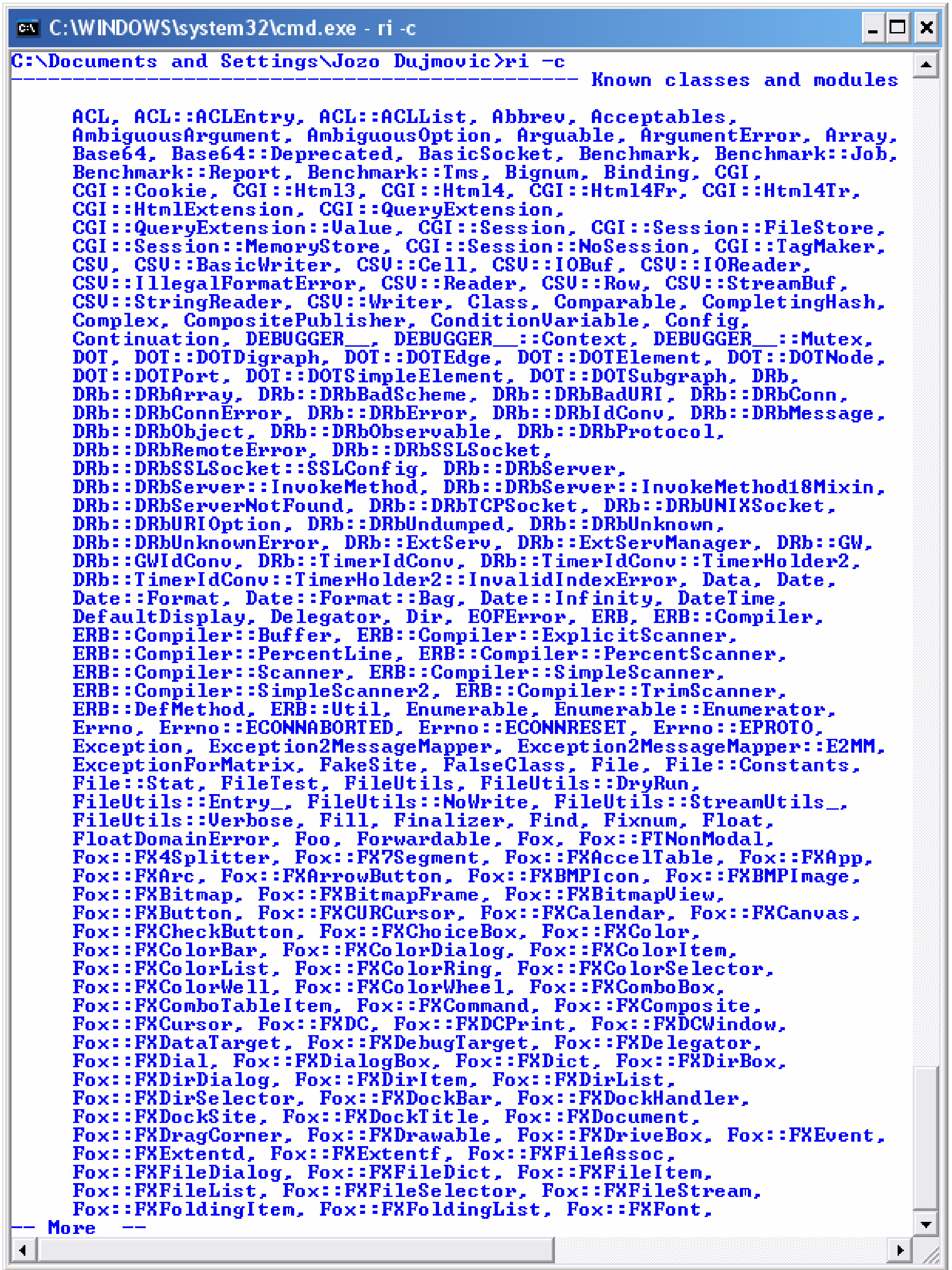


```
C:\Documents and Settings\Jozo Dujmovic>ri --help

ri v1.0.1 - 20041108

Usage:

  ri [options]  [names...]

Display information on Ruby classes, modules, and methods.
Give the names of classes or methods to see their documentation.
Partial names may be given: if the names match more than
one entity, a list will be shown, otherwise details on
that entity will be displayed.

Nested classes and modules can be specified using the normal
Name::Name notation, and instance methods can be distinguished
from class methods using "." (or "#") instead of "::".

For example:

    ri  File
    ri  File.new
    ri  F.n
    ri  zip

Note that shell quoting may be required for method names
containing punctuation:

    ri 'Array.[]'
    ri compact\!

By default ri searches for documentation in the following
directories:

    c:/ruby/share/ri/1.8/system
    c:/ruby/share/ri/1.8/site
    C:\Documents and Settings\Jozo Dujmovic/.rdoc
    c:/ruby/lib/ruby/gems/1.8/doc/*/ri

Specifying the --system, --site, --home, --gems or --doc-dir
options will limit ri to searching only the specified
directories.
```



```
Options:

     --help, -h   you're looking at it

  --classes, -c   Display the names of classes and modules we
                  know about

  --doc-dir, -d <dirname>
                  A directory to search for documentation. If not
                  specified, we search the standard rdoc/ri directories.
                  May be repeated.

       --system   Include documentation from Ruby's standard library:
                    c:/ruby/share/ri/1.8/system

         --site   Include documentation from libraries installed in site_lib:
                    c:/ruby/share/ri/1.8/site

         --home   Include documentation stored in ~/.rdoc:
                    C:\Documents and Settings\Jozo Dujmovic/.rdoc

         --gems   Include documentation from Rubygems:
                    c:/ruby/lib/ruby/gems/1.8/doc/*/ri

   --format, -f <name>
                  Format to use when displaying output:
                     ansi, bs, html, plain, simple
                  Use 'bs' (backspace) with most pager programs.
                  To use ANSI, either also use the -T option, or
                  tell your pager to allow control characters
                  (for example using the -R option to less)

--list-names, -l  List all the names known to RDoc, one per line

 --no-pager, -T   Send output directly to stdout.

    --width, -w output width
                  Set the width of the output

  --version, -v   Display the version of ri

Options may also be passed in the 'RI' environment variable

C:\Documents and Settings\Jozo Dujmovic>_
```

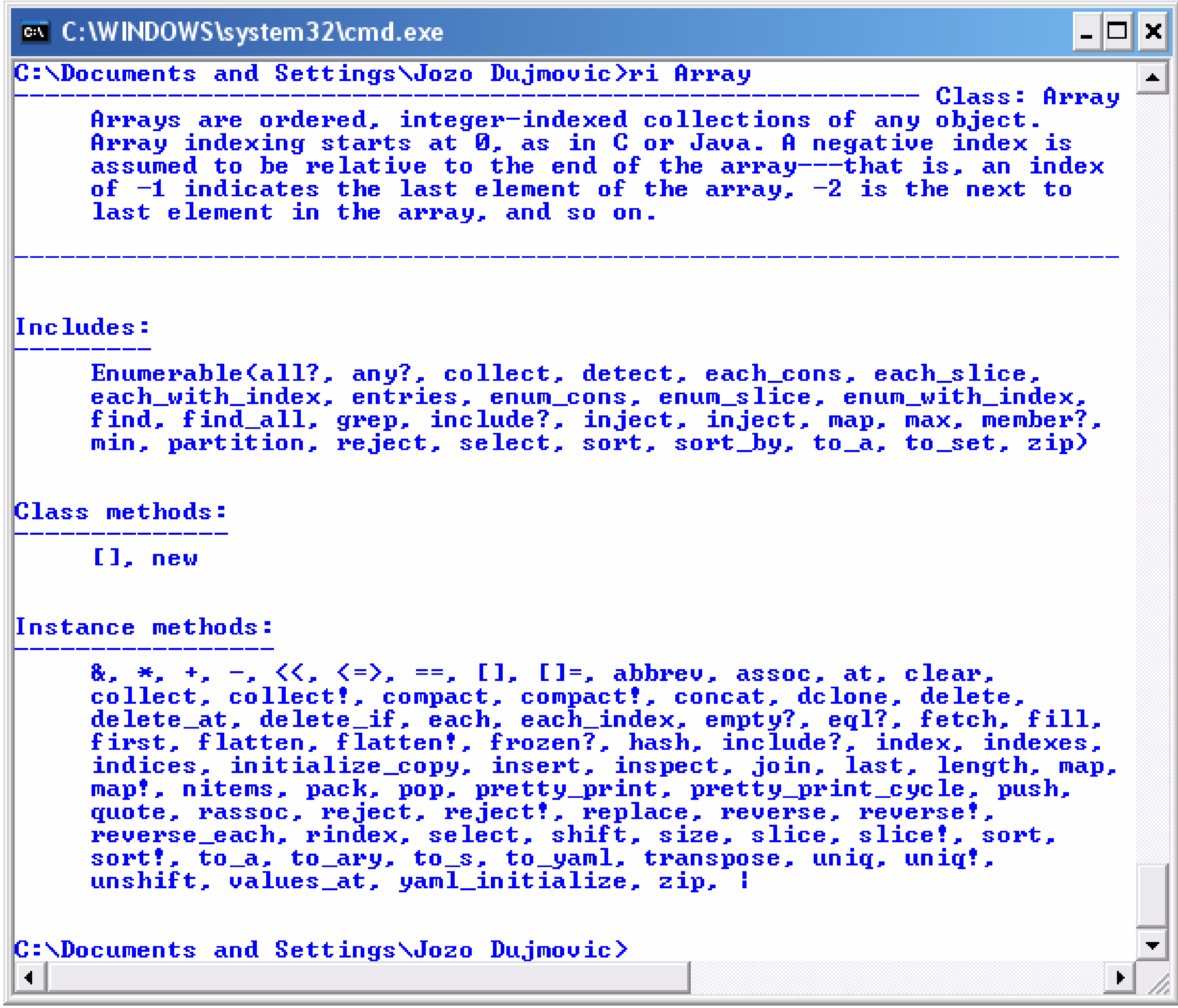
Using ri -c to find all classes



```
C:\Documents and Settings\Jozo Dujmovic>ri -c
---------------------------------------------- Known classes and modules

     ACL, ACL::ACLEntry, ACL::ACLList, Abbrev, Acceptables,
     AmbiguousArgument, AmbiguousOption, Arguable, ArgumentError, Array,
     Base64, Base64::Deprecated, BasicSocket, Benchmark, Benchmark::Job,
     Benchmark::Report, Benchmark::Tms, Bignum, Binding, CGI,
     CGI::Cookie, CGI::Html3, CGI::Html4, CGI::Html4Fr, CGI::Html4Tr,
     CGI::HtmlExtension, CGI::QueryExtension,
     CGI::QueryExtension::Value, CGI::Session, CGI::Session::FileStore,
     CGI::Session::MemoryStore, CGI::Session::NoSession, CGI::TagMaker,
     CSV, CSV::BasicWriter, CSV::Cell, CSV::IOBuf, CSV::IOReader,
     CSV::IllegalFormatError, CSV::Reader, CSV::Row, CSV::StreamBuf,
     CSV::StringReader, CSV::Writer, Class, Comparable, CompletingHash,
     Complex, CompositePublisher, ConditionVariable, Config,
     Continuation, DEBUGGER__, DEBUGGER__::Context, DEBUGGER__::Mutex,
     DOT, DOT::DOTDigraph, DOT::DOTEdge, DOT::DOTElement, DOT::DOTNode,
     DOT::DOTPort, DOT::DOTSimpleElement, DOT::DOTSubgraph, DRb,
     DRb::DRbArray, DRb::DRbBadScheme, DRb::DRbBadURI, DRb::DRbConn,
     DRb::DRbConnError, DRb::DRbError, DRb::DRbIdConv, DRb::DRbMessage,
     DRb::DRbObject, DRb::DRbObservable, DRb::DRbProtocol,
     DRb::DRbRemoteError, DRb::DRbSSLSocket,
     DRb::DRbSSLSocket::SSLConfig, DRb::DRbServer,
     DRb::DRbServer::InvokeMethod, DRb::DRbServer::InvokeMethod18Mixin,
     DRb::DRbServerNotFound, DRb::DRbTCPSocket, DRb::DRbUNIXSocket,
     DRb::DRbURIOption, DRb::DRbUndumped, DRb::DRbUnknown,
     DRb::DRbUnknownError, DRb::ExtServ, DRb::ExtServManager, DRb::GW,
     DRb::GWIdConv, DRb::TimerIdConv, DRb::TimerIdConv::TimerHolder2,
     DRb::TimerIdConv::TimerHolder2::InvalidIndexError, Data, Date,
     Date::Format, Date::Format::Bag, Date::Infinity, DateTime,
     DefaultDisplay, Delegator, Dir, EOFError, ERB, ERB::Compiler,
     ERB::Compiler::Buffer, ERB::Compiler::ExplicitScanner,
     ERB::Compiler::PercentLine, ERB::Compiler::PercentScanner,
     ERB::Compiler::Scanner, ERB::Compiler::SimpleScanner,
     ERB::Compiler::SimpleScanner2, ERB::Compiler::TrimScanner,
     ERB::DefMethod, ERB::Util, Enumerable, Enumerable::Enumerator,
     Errno, Errno::ECONNABORTED, Errno::ECONNRESET, Errno::EPROTO,
     Exception, Exception2MessageMapper, Exception2MessageMapper::E2MM,
     ExceptionForMatrix, FakeSite, FalseClass, File, File::Constants,
     File::Stat, FileTest, FileUtils, FileUtils::DryRun,
     FileUtils::Entry_, FileUtils::NoWrite, FileUtils::StreamUtils_,
     FileUtils::Verbose, Fill, Finalizer, Find, Fixnum, Float,
     FloatDomainError, Foo, Forwardable, Fox, Fox::FTNonModal,
     Fox::FX4Splitter, Fox::FX7Segment, Fox::FXAccelTable, Fox::FXApp,
     Fox::FXArc, Fox::FXArrowButton, Fox::FXBMPIcon, Fox::FXBMPImage,
     Fox::FXBitmap, Fox::FXBitmapFrame, Fox::FXBitmapView,
     Fox::FXButton, Fox::FXCURCursor, Fox::FXCalendar, Fox::FXCanvas,
     Fox::FXCheckButton, Fox::FXChoiceBox, Fox::FXColor,
     Fox::FXColorBar, Fox::FXColorDialog, Fox::FXColorItem,
     Fox::FXColorList, Fox::FXColorRing, Fox::FXColorSelector,
     Fox::FXColorWell, Fox::FXColorWheel, Fox::FXComboBox,
     Fox::FXComboTableItem, Fox::FXCommand, Fox::FXComposite,
     Fox::FXCursor, Fox::FXDC, Fox::FXDCPrint, Fox::FXDCWindow,
     Fox::FXDataTarget, Fox::FXDebugTarget, Fox::FXDelegator,
     Fox::FXDial, Fox::FXDialogBox, Fox::FXDict, Fox::FXDirBox,
     Fox::FXDirDialog, Fox::FXDirItem, Fox::FXDirList,
     Fox::FXDirSelector, Fox::FXDockBar, Fox::FXDockHandler,
     Fox::FXDockSite, Fox::FXDockTitle, Fox::FXDocument,
     Fox::FXDragCorner, Fox::FXDrawable, Fox::FXDriveBox, Fox::FXEvent,
     Fox::FXExtentd, Fox::FXExtentf, Fox::FXFileAssoc,
     Fox::FXFileDialog, Fox::FXFileDict, Fox::FXFileItem,
     Fox::FXFileList, Fox::FXFileSelector, Fox::FXFileStream,
     Fox::FXFoldingItem, Fox::FXFoldingList, Fox::FXFont,
-- More  --
```

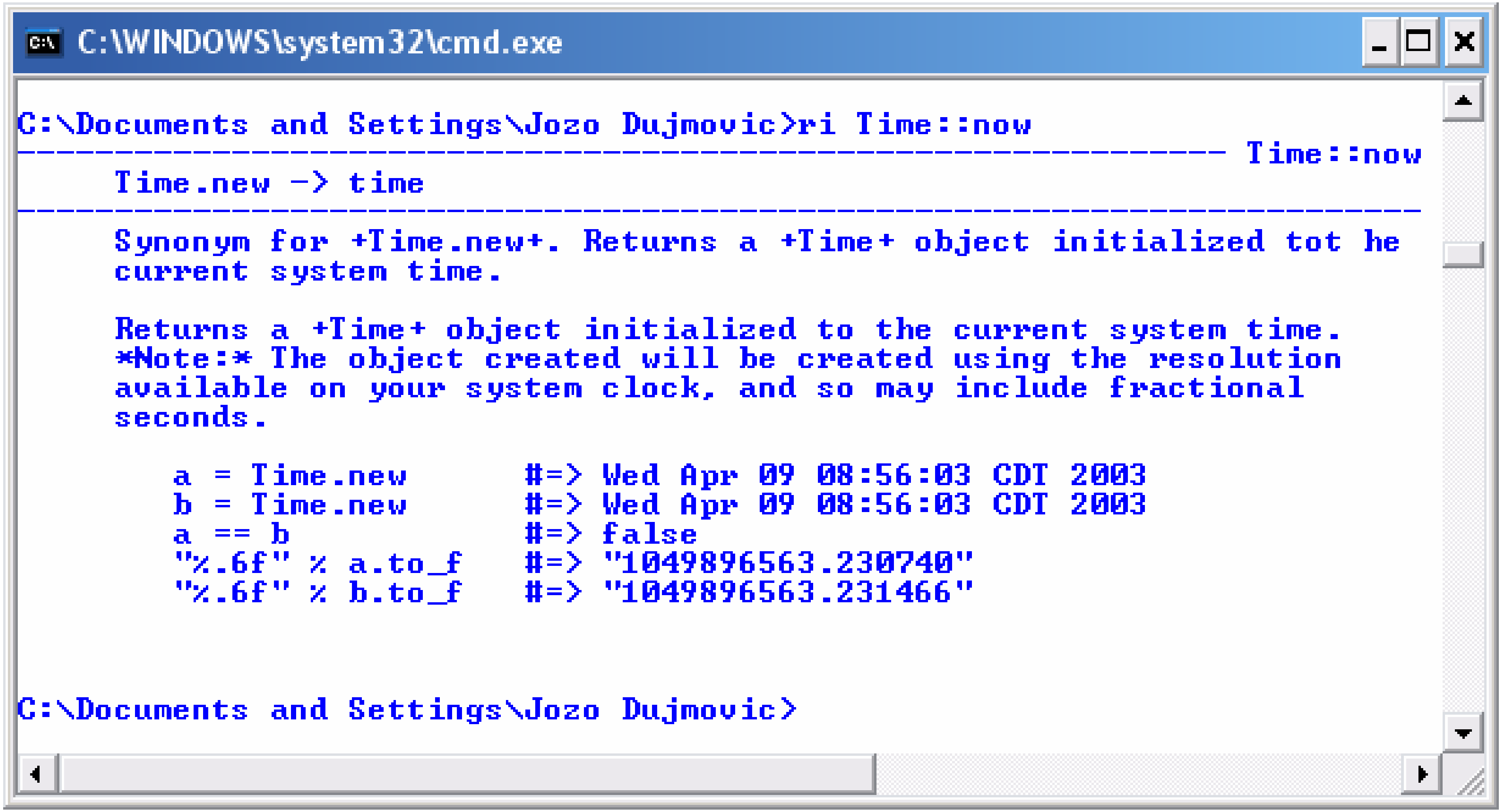
© Jozo Dujmović

Using ri ClassName

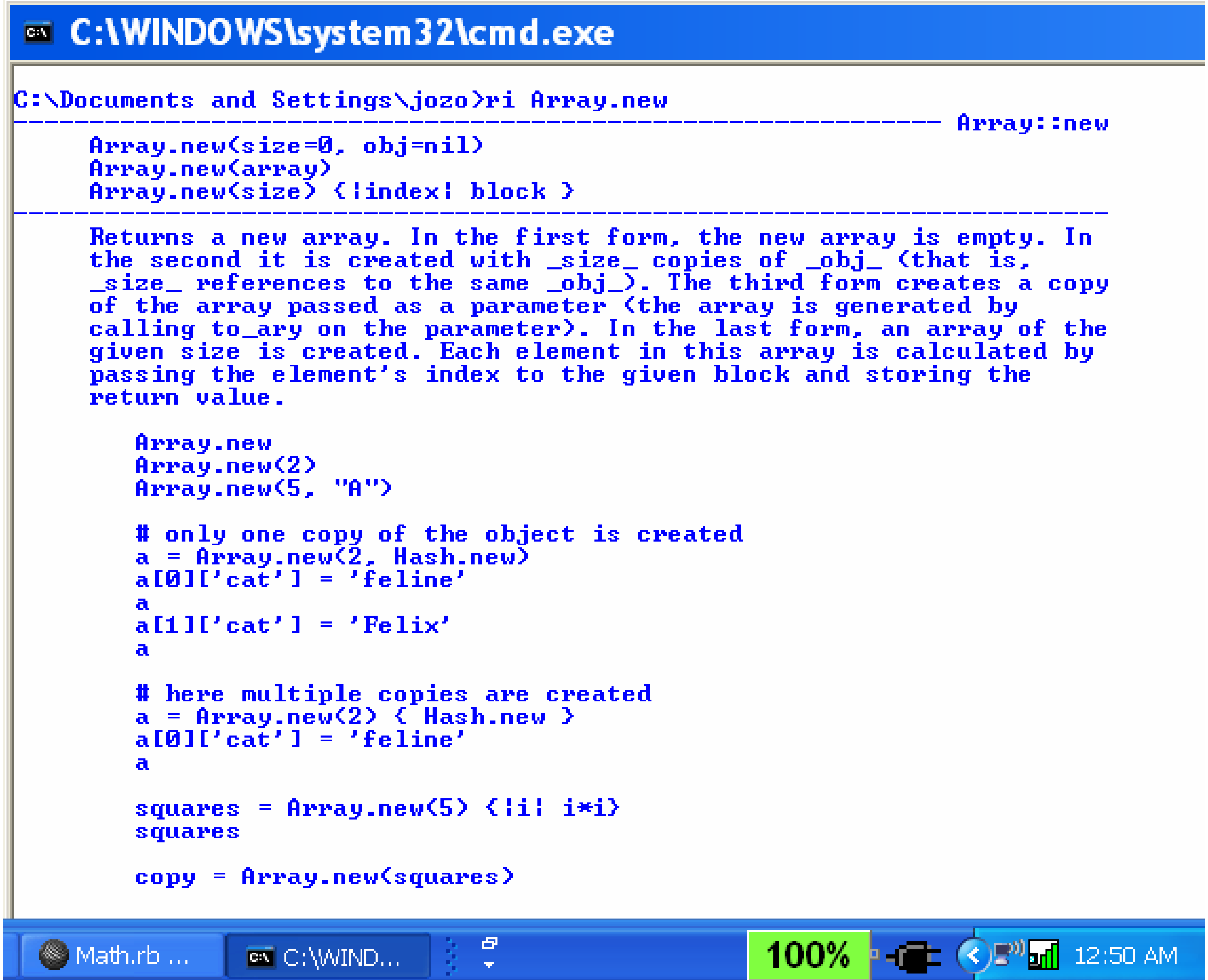


```
C:\Documents and Settings\Jozo Dujmovic>ri Array
----------------------------------------------------------- Class: Array
     Arrays are ordered, integer-indexed collections of any object.
     Array indexing starts at 0, as in C or Java. A negative index is
     assumed to be relative to the end of the array---that is, an index
     of -1 indicates the last element of the array, -2 is the next to
     last element in the array, and so on.

------------------------------------------------------------------------


Includes:
---------
     Enumerable(all?, any?, collect, detect, each_cons, each_slice,
     each_with_index, entries, enum_cons, enum_slice, enum_with_index,
     find, find_all, grep, include?, inject, inject, map, max, member?,
     min, partition, reject, select, sort, sort_by, to_a, to_set, zip)


Class methods:
--------------
     [], new


Instance methods:
-----------------
     &, *, +, -, <<, <=>, ==, [], []=, abbrev, assoc, at, clear,
     collect, collect!, compact, compact!, concat, dclone, delete,
     delete_at, delete_if, each, each_index, empty?, eql?, fetch, fill,
     first, flatten, flatten!, frozen?, hash, include?, index, indexes,
     indices, initialize_copy, insert, inspect, join, last, length, map,
     map!, nitems, pack, pop, pretty_print, pretty_print_cycle, push,
     quote, rassoc, reject, reject!, replace, reverse, reverse!,
     reverse_each, rindex, select, shift, size, slice, slice!, sort,
     sort!, to_a, to_ary, to_s, to_yaml, transpose, uniq, uniq!,
     unshift, values_at, yaml_initialize, zip, |


C:\Documents and Settings\Jozo Dujmovic>
```

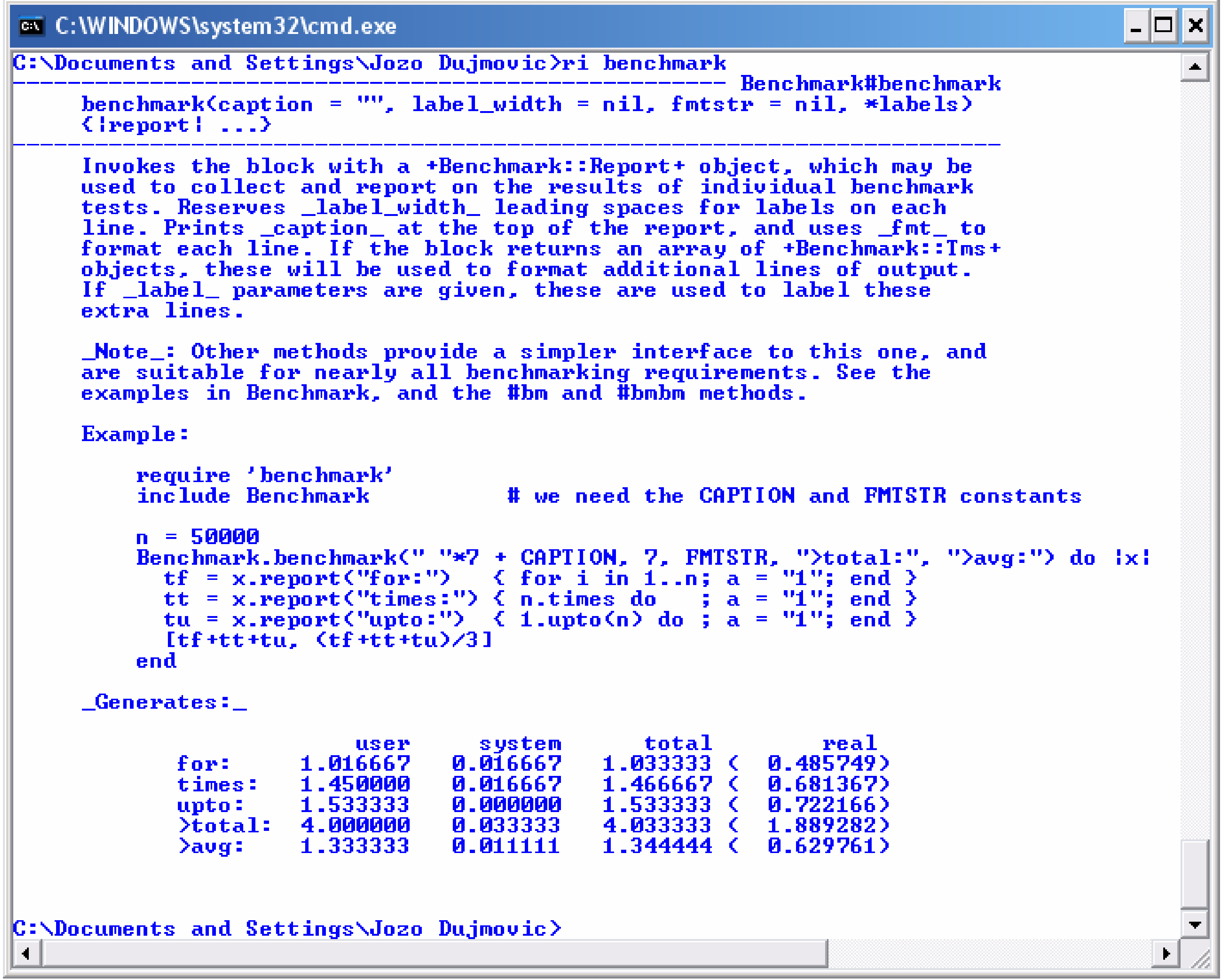
Using ri to find specific method



```
C:\Documents and Settings\Jozo Dujmovic>ri Time::now
--------------------------------------------------------------- Time::now
     Time.new -> time
------------------------------------------------------------------------
     Synonym for +Time.new+. Returns a +Time+ object initialized tot he
     current system time.

     Returns a +Time+ object initialized to the current system time.
     *Note:* The object created will be created using the resolution
     available on your system clock, and so may include fractional
     seconds.

        a = Time.new      #=> Wed Apr 09 08:56:03 CDT 2003
        b = Time.new      #=> Wed Apr 09 08:56:03 CDT 2003
        a == b            #=> false
        "%.6f" % a.to_f   #=> "1049896563.230740"
        "%.6f" % b.to_f   #=> "1049896563.231466"


C:\Documents and Settings\Jozo Dujmovic>
```



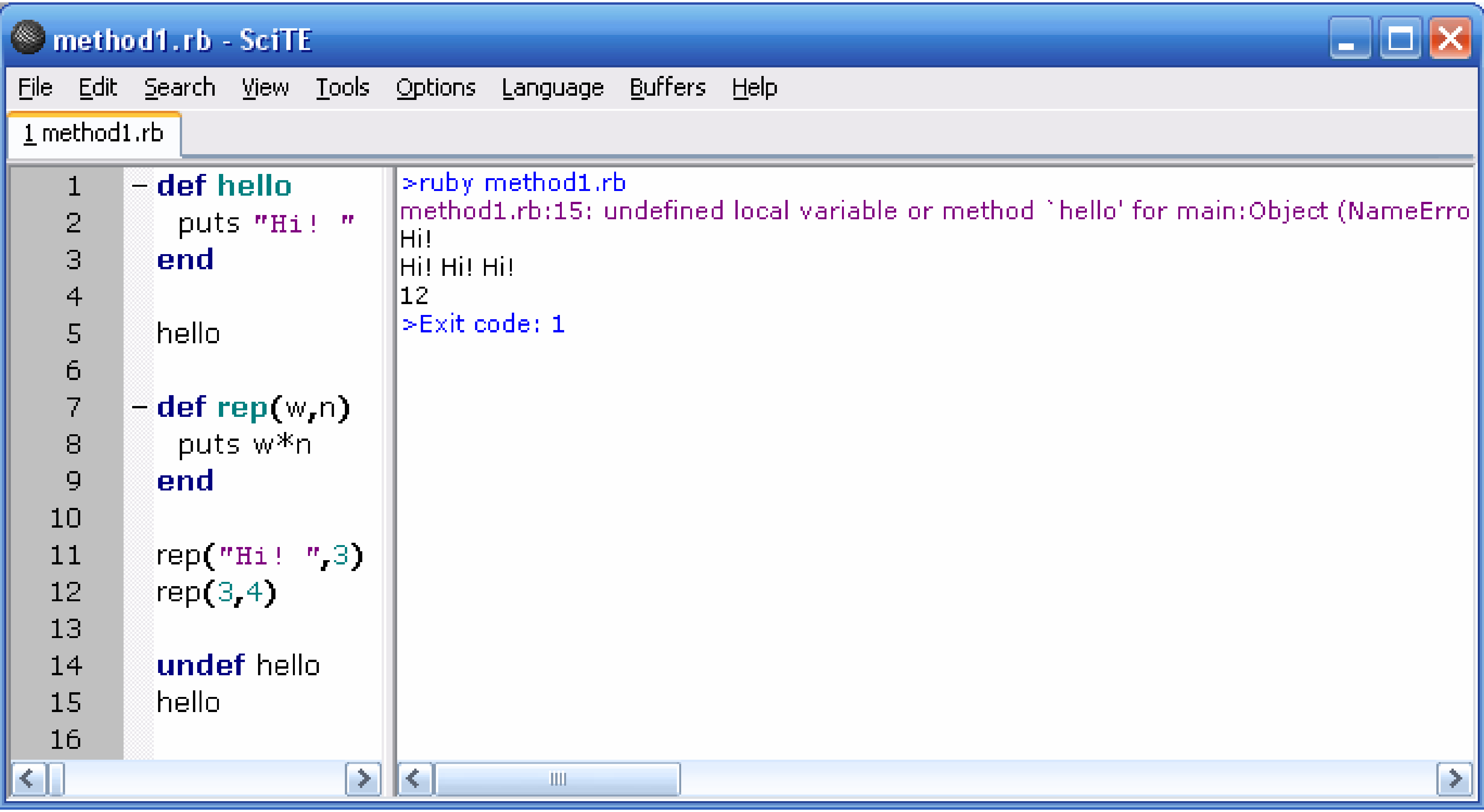
```
C:\Documents and Settings\jozo>ri Array.new
------------------------------------------------------------ Array::new
     Array.new(size=0, obj=nil)
     Array.new(array)
     Array.new(size) {|index| block }
------------------------------------------------------------------------
     Returns a new array. In the first form, the new array is empty. In
     the second it is created with _size_ copies of _obj_ (that is,
     _size_ references to the same _obj_). The third form creates a copy
     of the array passed as a parameter (the array is generated by
     calling to_ary on the parameter). In the last form, an array of the
     given size is created. Each element in this array is calculated by
     passing the element's index to the given block and storing the
     return value.

        Array.new
        Array.new(2)
        Array.new(5, "A")

        # only one copy of the object is created
        a = Array.new(2, Hash.new)
        a[0]['cat'] = 'feline'
        a
        a[1]['cat'] = 'Felix'
        a

        # here multiple copies are created
        a = Array.new(2) { Hash.new }
        a[0]['cat'] = 'feline'
        a

        squares = Array.new(5) {|i| i*i}
        squares

        copy = Array.new(squares)
```



```
C:\Documents and Settings\Jozo Dujmovic>ri benchmark
---------------------------------------------------- Benchmark#benchmark
     benchmark(caption = "", label_width = nil, fmtstr = nil, *labels)
     {|report| ...}
------------------------------------------------------------------------
     Invokes the block with a +Benchmark::Report+ object, which may be
     used to collect and report on the results of individual benchmark
     tests. Reserves _label_width_ leading spaces for labels on each
     line. Prints _caption_ at the top of the report, and uses _fmt_ to
     format each line. If the block returns an array of +Benchmark::Tms+
     objects, these will be used to format additional lines of output.
     If _label_ parameters are given, these are used to label these
     extra lines.

     _Note_: Other methods provide a simpler interface to this one, and
     are suitable for nearly all benchmarking requirements. See the
     examples in Benchmark, and the #bm and #bmbm methods.

     Example:

         require 'benchmark'
         include Benchmark          # we need the CAPTION and FMTSTR constants

         n = 50000
         Benchmark.benchmark(" "*7 + CAPTION, 7, FMTSTR, ">total:", ">avg:") do |x|
           tf = x.report("for:")   { for i in 1..n; a = "1"; end }
           tt = x.report("times:") { n.times do   ; a = "1"; end }
           tu = x.report("upto:")  { 1.upto(n) do ; a = "1"; end }
           [tf+tt+tu, (tf+tt+tu)/3]
         end

     _Generates:_

                         user   system        total        real
            for:     1.016667   0.016667   1.033333 (  0.485749)
            times:   1.450000   0.016667   1.466667 (  0.681367)
            upto:    1.533333   0.000000   1.533333 (  0.722166)
            >total:  4.000000   0.033333   4.033333 (  1.889282)
            >avg:    1.333333   0.011111   1.344444 (  0.629761)


C:\Documents and Settings\Jozo Dujmovic>
```

Methods

- Method is a named collection of statements that can be called repeatedly
- Method can be defined (using keyword def) and undefined (using keyword undef)
- Methods return the last expression that is evaluated
- Methods can use the return statement to return a value
- Methods are sometimes used only for side effects



```ruby
def hello
  puts "Hi! "
end

hello

def rep(w,n)
  puts w*n
end

rep("Hi! ",3)
rep(3,4)

undef hello
hello
```

```
>ruby method1.rb
method1.rb:15: undefined local variable or method `hello' for main:Object (NameErro
Hi!
Hi! Hi! Hi!
12
>Exit code: 1
```

Methods - syntax

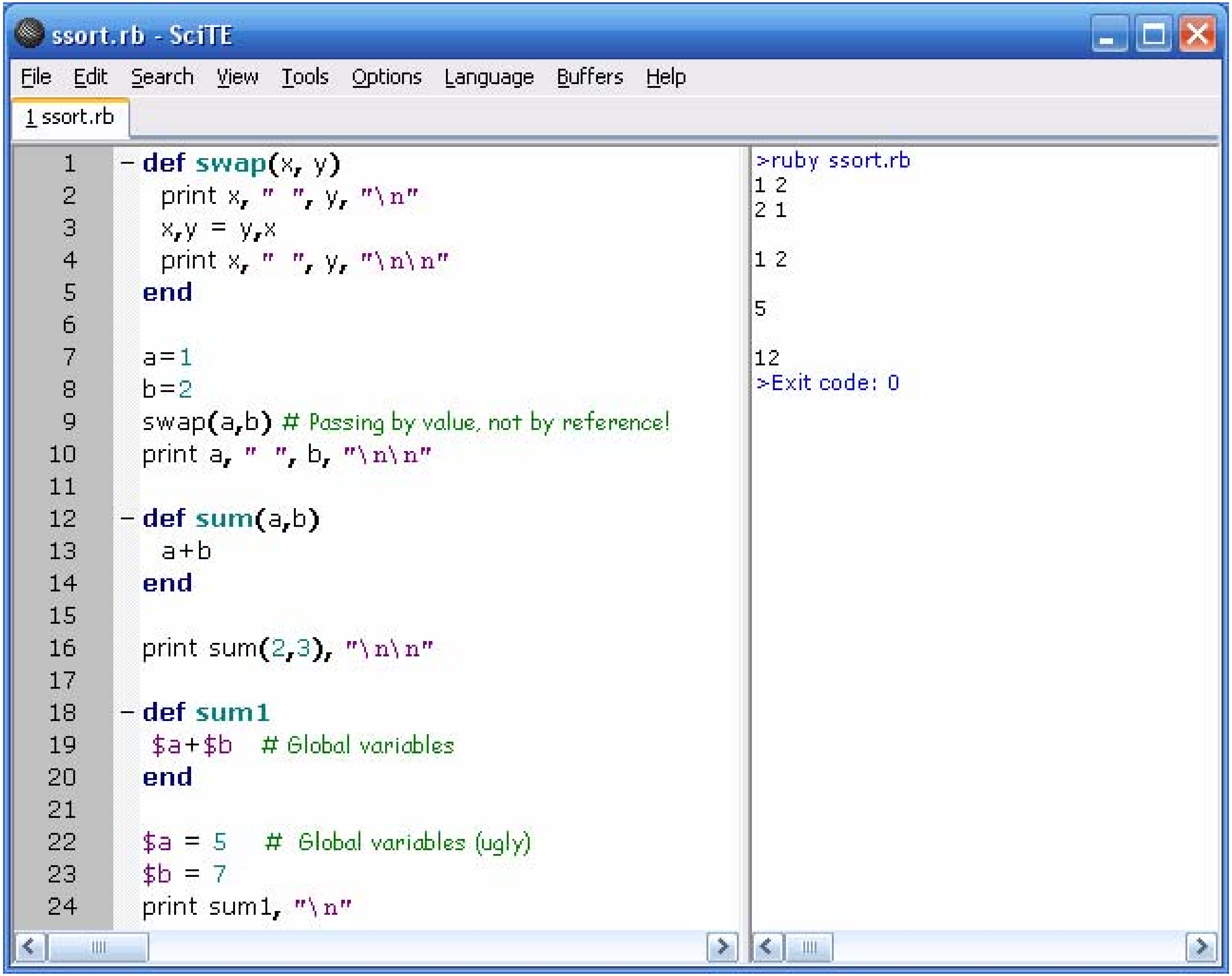
- Method is inserted inside def – end
- Method name normally starts with a lowercase letter. The trailing letter can be ‘?’ (used for recognizers), ‘!’ (dangerous), or ‘=‘ (used for class instance setters)
- Syntax:

```
def <method name> [ [( ] <arg1>,…,<argn> [ )] ]
      <statements>
end
```

- Simple arguments are local variables of the method (must be lowercase); their scope is within the method

Convention about parentheses

- Parentheses are optional. If there are local arguments it is suitable (because of notation in math and other languages) to use parentheses
- If there are no arguments the empty parentheses ( ) can be omitted

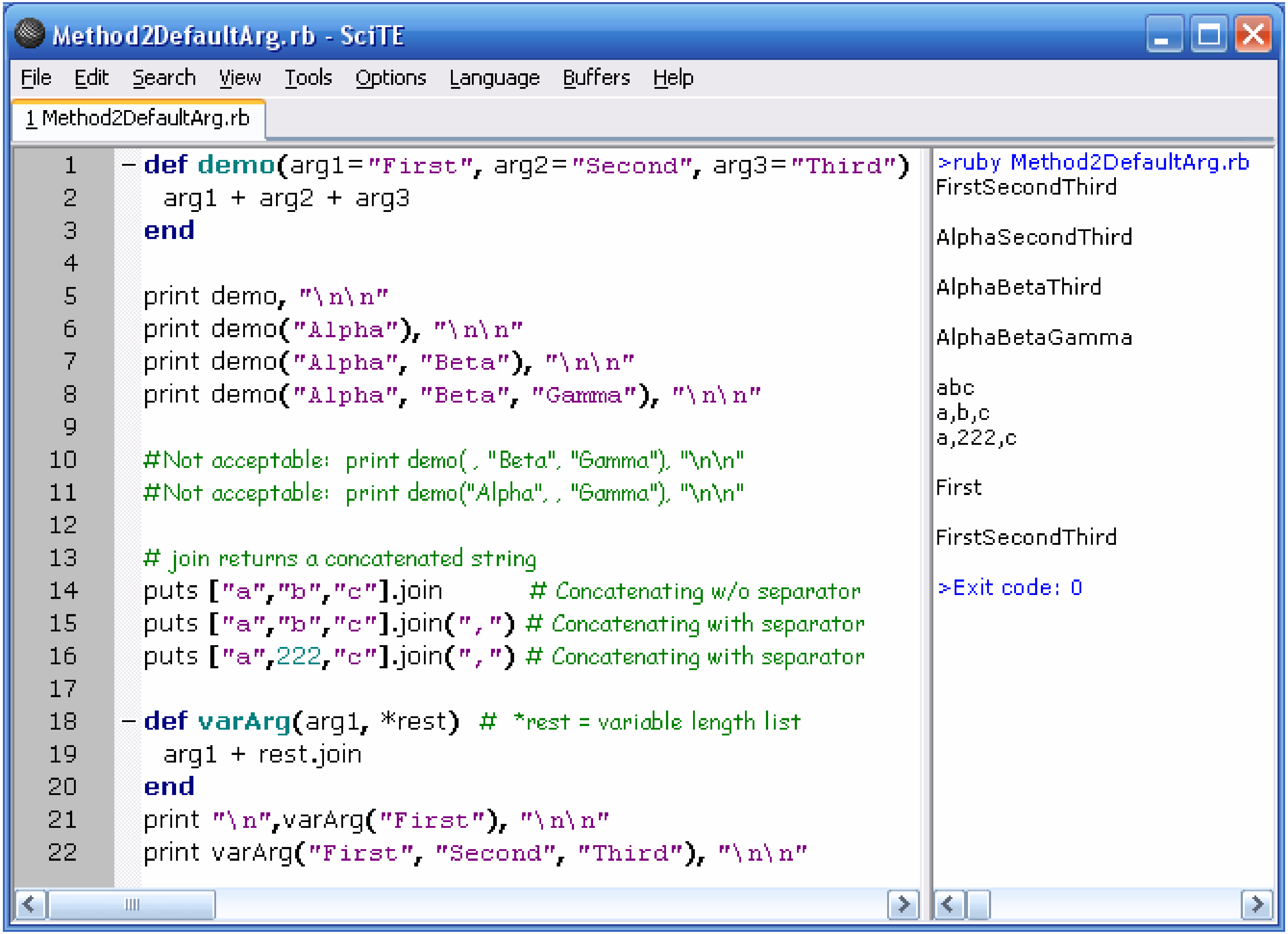


```
def swap(x, y)
  print x, "  ", y, "\n"
  x,y = y,x
  print x, "  ", y, "\n\n"
end

a=1
b=2
swap(a,b) # Passing by value, not by reference!
print a, "  ", b, "\n\n"

def sum(a,b)
  a+b
end

print sum(2,3), "\n\n"

def sum1
 $a+$b   # Global variables
end

$a = 5    #  Global variables (ugly)
$b = 7
print sum1, "\n"
```

```
>ruby ssort.rb
1 2
2 1

1 2

5

12
>Exit code: 0
```

Default values of method arguments

- Ruby permits the use of default values of method arguments
- Default values are used if the user does not provide the values of arguments
- The list of arguments can have variable length



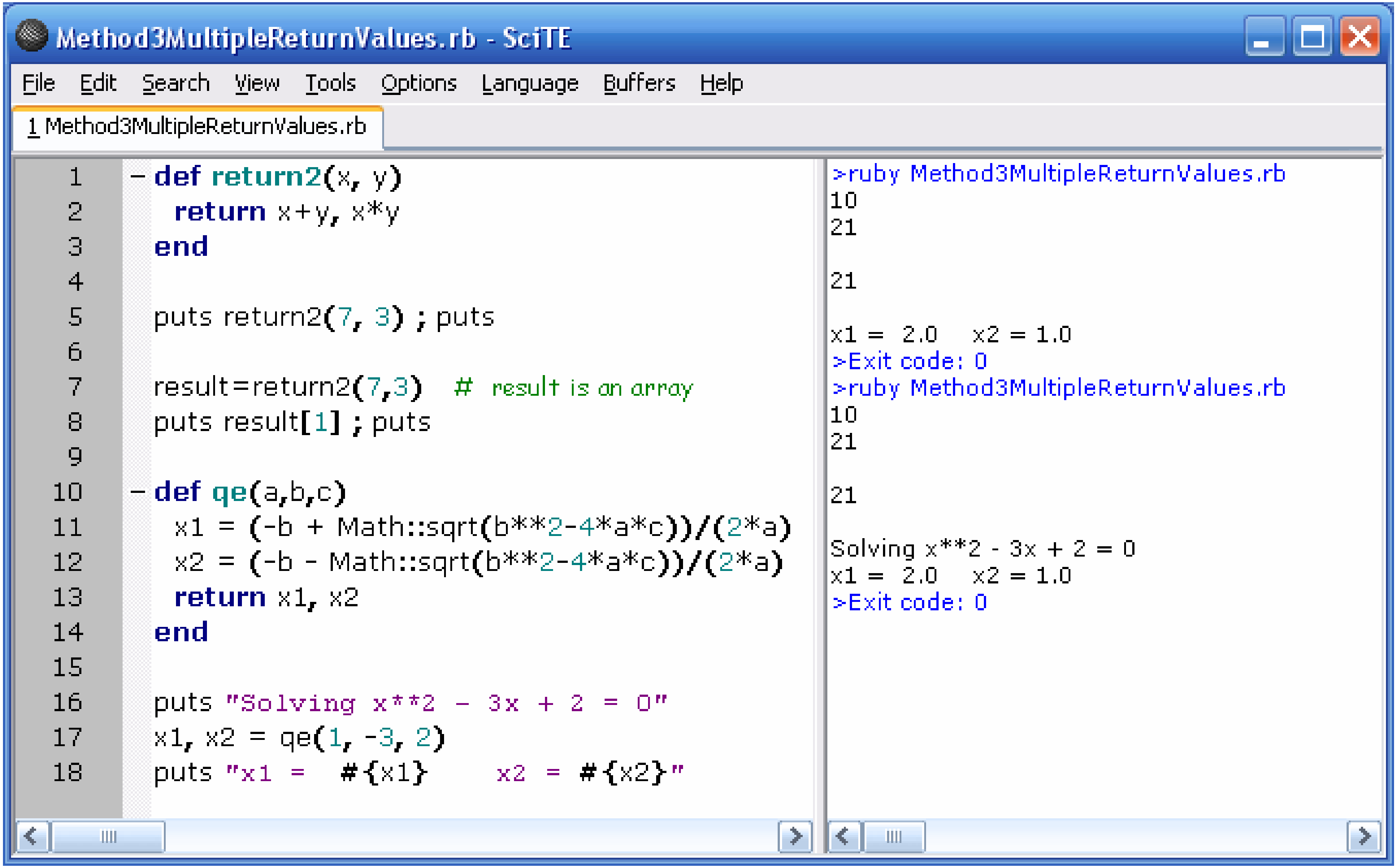
```ruby
def demo(arg1="First", arg2="Second", arg3="Third")
  arg1 + arg2 + arg3
end

print demo, "\n\n"
print demo("Alpha"), "\n\n"
print demo("Alpha", "Beta"), "\n\n"
print demo("Alpha", "Beta", "Gamma"), "\n\n"

#Not acceptable:  print demo( , "Beta", "Gamma"), "\n\n"
#Not acceptable:  print demo("Alpha", , "Gamma"), "\n\n"

# join returns a concatenated string
puts ["a","b","c"].join         # Concatenating w/o separator
puts ["a","b","c"].join(",") # Concatenating with separator
puts ["a",222,"c"].join(",") # Concatenating with separator

def varArg(arg1, *rest)  #  *rest = variable length list
  arg1 + rest.join
end
print "\n",varArg("First"), "\n\n"
print varArg("First", "Second", "Third"), "\n\n"
```

```
>ruby Method2DefaultArg.rb
FirstSecondThird

AlphaSecondThird

AlphaBetaThird

AlphaBetaGamma

abc
a,b,c
a,222,c

First

FirstSecondThird

>Exit code: 0
```

Return values

- Every method returns a value
- The returned value may or may not be used
- The returned value of the method is the value of the last executed statement
- There might be one or more returned values
- Multiple returned values can be returned by

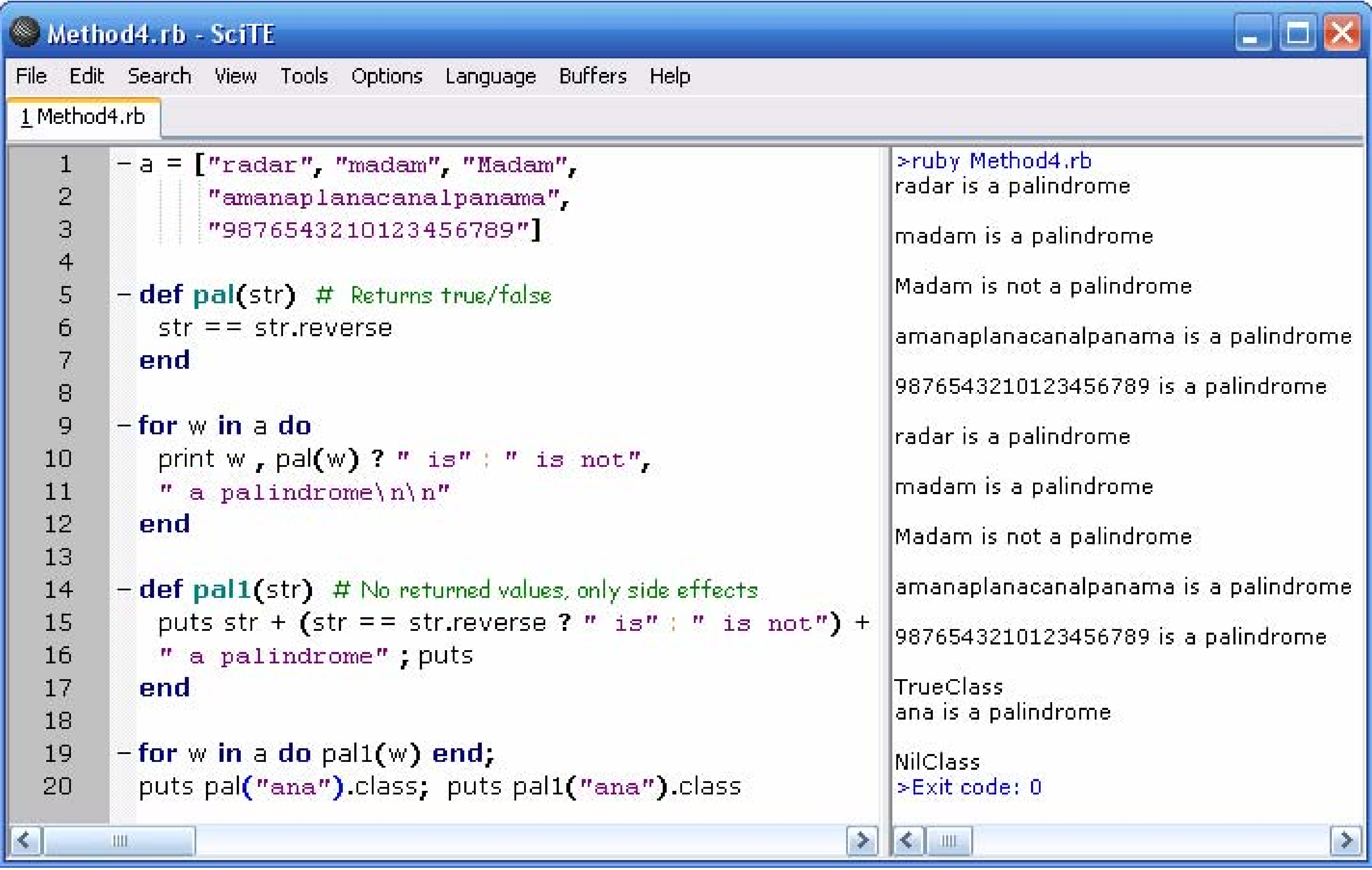
  return first, second, third
- Multiple returned values are returned in an array: [first, second, third]


Method3MultipleReturnValues.rb - SciTE
File Edit Search View Tools Options Language Buffers Help
1 Method3MultipleReturnValues.rb

```ruby
def return2(x, y)
  return x+y, x*y
end

puts return2(7, 3) ; puts

result=return2(7,3)   #  result is an array
puts result[1] ; puts

def qe(a,b,c)
  x1 = (-b + Math::sqrt(b**2-4*a*c))/(2*a)
  x2 = (-b - Math::sqrt(b**2-4*a*c))/(2*a)
  return x1, x2
end

puts "Solving x**2 - 3x + 2 = 0"
x1, x2 = qe(1, -3, 2)
puts "x1 =  #{x1}     x2 = #{x2}"
```

```
>ruby Method3MultipleReturnValues.rb
10
21

21

x1 =  2.0    x2 = 1.0
>Exit code: 0
>ruby Method3MultipleReturnValues.rb
10
21

21

Solving x**2 - 3x + 2 = 0
x1 =  2.0    x2 = 1.0
>Exit code: 0
```

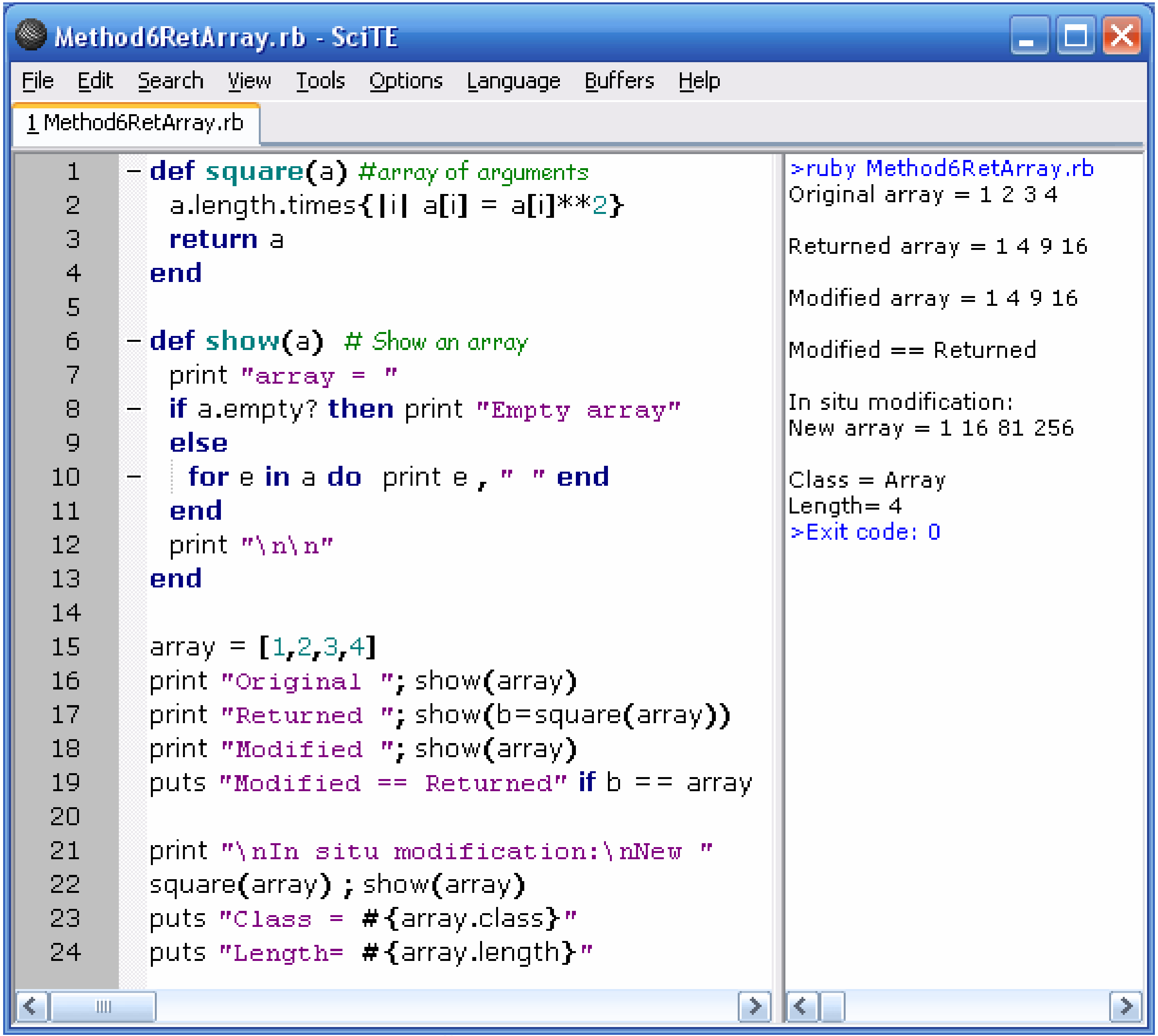
Using side effects



```ruby
a = ["radar", "madam", "Madam",
     "amanaplanacanalpanama",
     "98765432101234567889"]

def pal(str)  # Returns true/false
  str == str.reverse
end

for w in a do
  print w , pal(w) ? " is" : " is not",
  " a palindrome\n\n"
end

def pal1(str)  # No returned values, only side effects
  puts str + (str == str.reverse ? " is" : " is not") +
  " a palindrome" ; puts
end

for w in a do pal1(w) end;
puts pal("ana").class;  puts pal1("ana").class
```

```
>ruby Method4.rb
radar is a palindrome

madam is a palindrome

Madam is not a palindrome

amanaplanacanalpanama is a palindrome

98765432101234567889 is a palindrome

radar is a palindrome

madam is a palindrome

Madam is not a palindrome

amanaplanacanalpanama is a palindrome

98765432101234567889 is a palindrome

TrueClass
ana is a palindrome

NilClass
>Exit code: 0
```

Ruby methods can modify (in situ) and return arrays



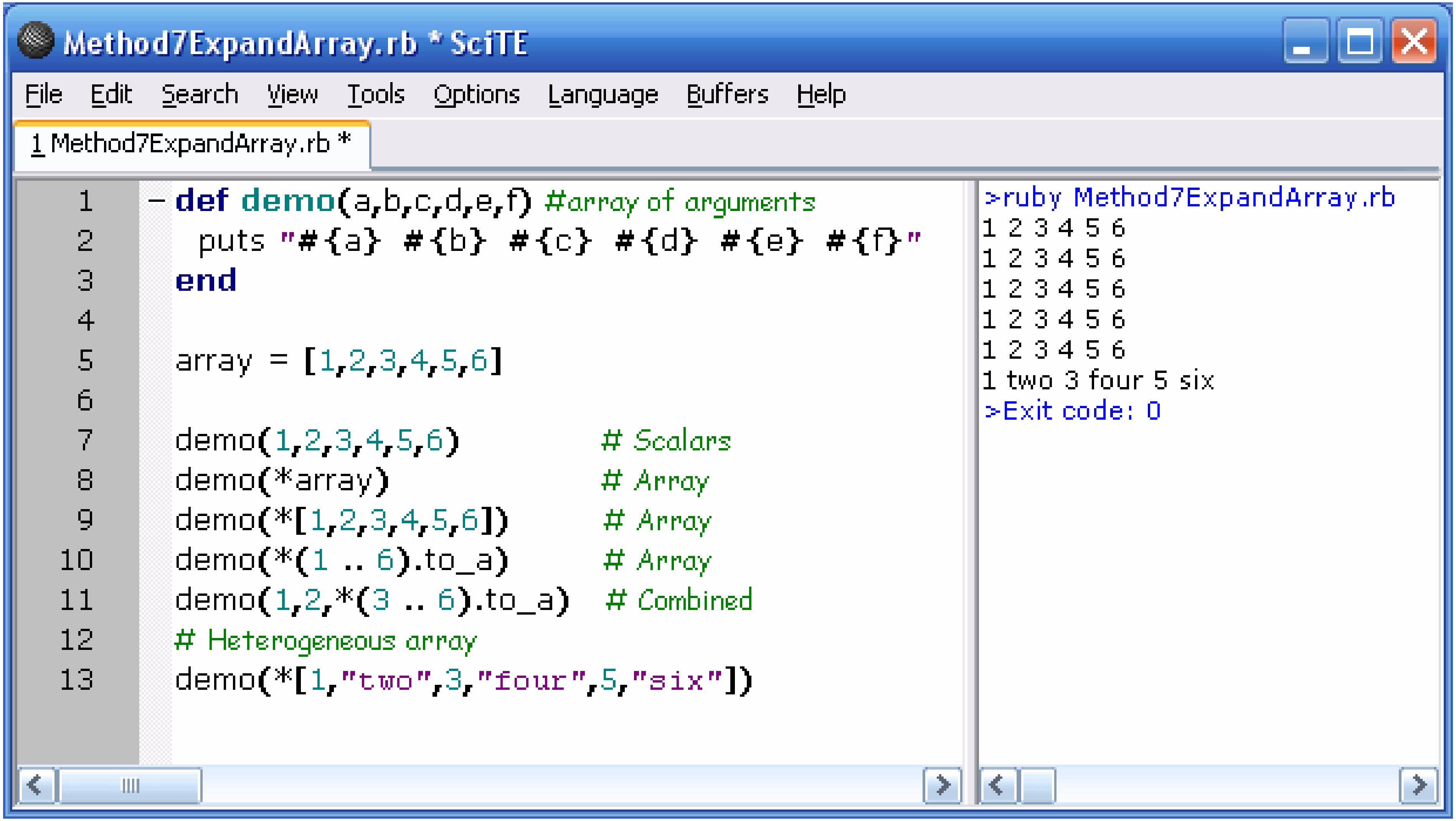
```ruby
def square(a) #array of arguments
  a.length.times{|i| a[i] = a[i]**2}
  return a
end

def show(a)  # Show an array
  print "array = "
  if a.empty? then print "Empty array"
  else
   for e in a do  print e , " " end
  end
  print "\n\n"
end

array = [1,2,3,4]
print "Original "; show(array)
print "Returned "; show(b=square(array))
print "Modified "; show(array)
puts "Modified == Returned" if b == array

print "\nIn situ modification:\nNew "
square(array) ; show(array)
puts "Class = #{array.class}"
puts "Length= #{array.length}"
```

```
>ruby Method6RetArray.rb
Original array = 1 2 3 4

Returned array = 1 4 9 16

Modified array = 1 4 9 16

Modified == Returned

In situ modification:
New array = 1 16 81 256

Class = Array
Length= 4
>Exit code: 0
```

Expanding arrays in method calls

- List of scalar arguments can be replaced by arrays or their parts
- Such arguments must be prefixed by an asterisk (“splat operator”)
- The number of arguments must be correct (array cannot be larger than the original number of arguments)

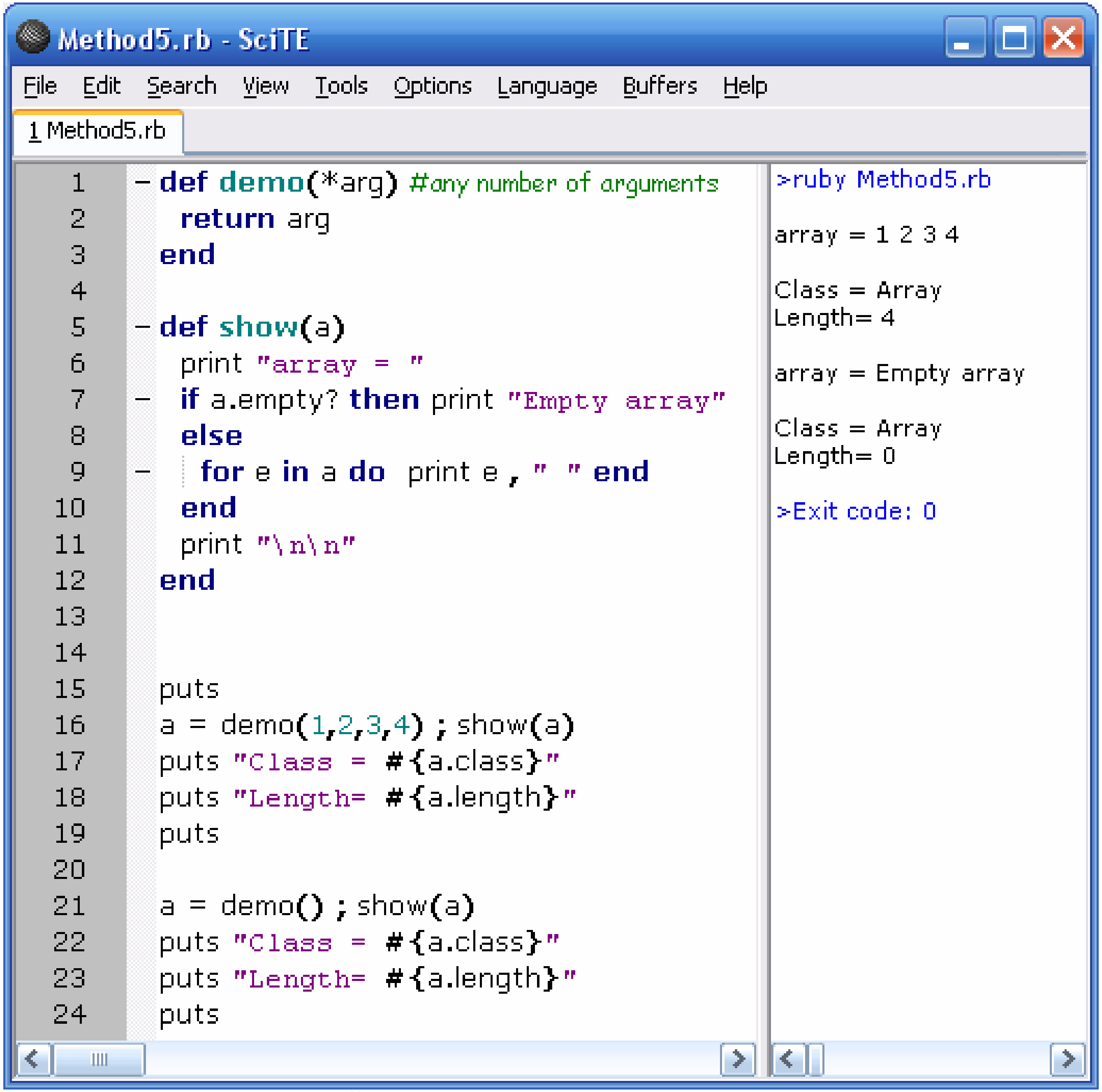
Expanding arrays


Method7ExpandArray.rb * SciTE
File Edit Search View Tools Options Language Buffers Help
1 Method7ExpandArray.rb *

```ruby
def demo(a,b,c,d,e,f) #array of arguments
  puts "#{a} #{b} #{c} #{d} #{e} #{f}"
end

array = [1,2,3,4,5,6]

demo(1,2,3,4,5,6)           # Scalars
demo(*array)                # Array
demo(*[1,2,3,4,5,6])        # Array
demo(*(1 .. 6).to_a)        # Array
demo(1,2,*(3 .. 6).to_a)    # Combined
# Heterogeneous array
demo(*[1,"two",3,"four",5,"six"])
```

```
>ruby Method7ExpandArray.rb
1 2 3 4 5 6
1 2 3 4 5 6
1 2 3 4 5 6
1 2 3 4 5 6
1 2 3 4 5 6
1 two 3 four 5 six
>Exit code: 0
```

Method with an arbitrary number of arguments: multiple arguments are bundled into an array. A method can return an empty array

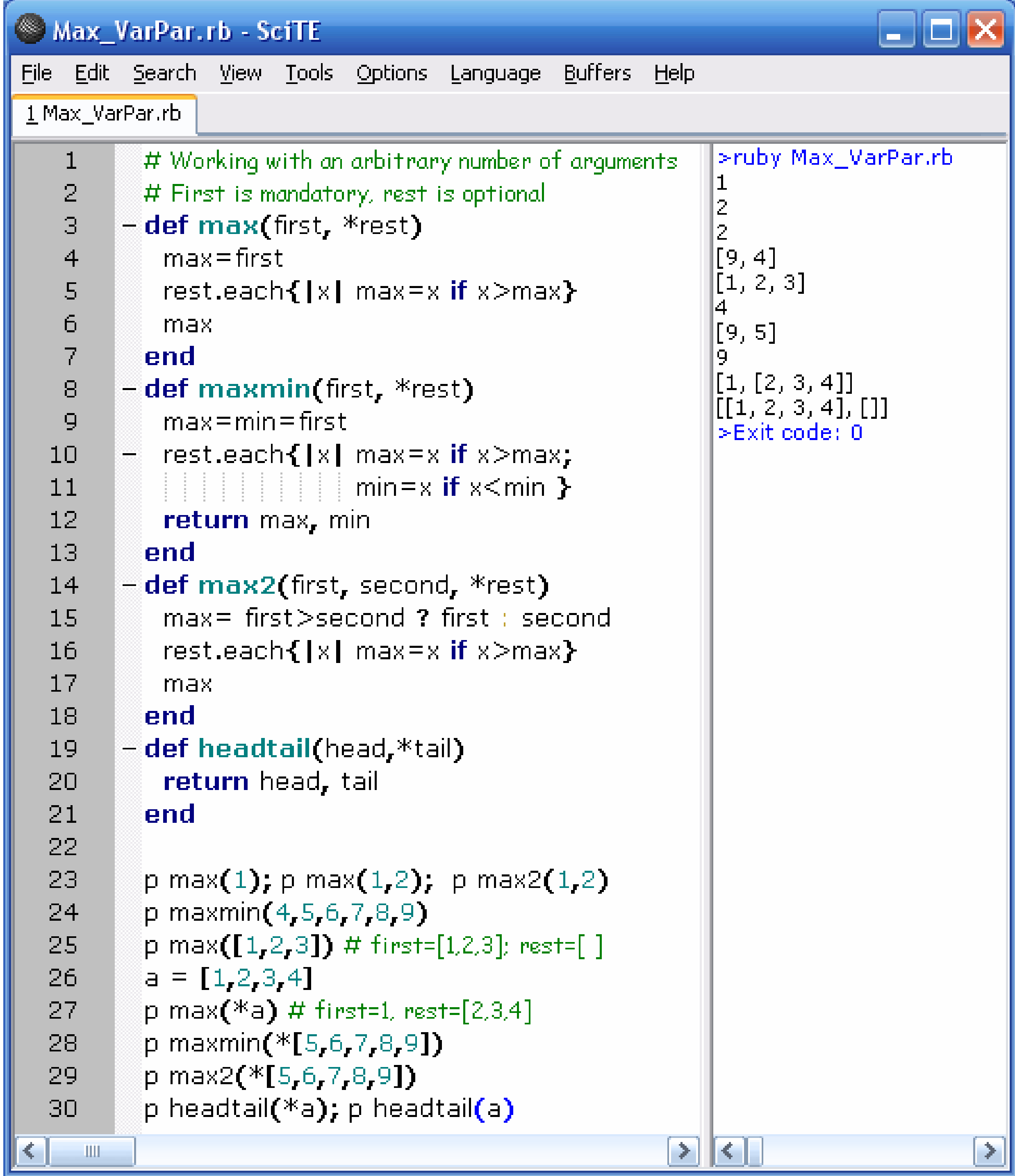


```
def demo(*arg) #any number of arguments
  return arg
end

def show(a)
  print "array = "
  if a.empty? then print "Empty array"
  else
   for e in a do  print e , " " end
  end
  print "\n\n"
end


puts
a = demo(1,2,3,4) ; show(a)
puts "Class = #{a.class}"
puts "Length= #{a.length}"
puts

a = demo() ; show(a)
puts "Class = #{a.class}"
puts "Length= #{a.length}"
puts
```

```
>ruby Method5.rb

array = 1 2 3 4

Class = Array
Length= 4

array = Empty array

Class = Array
Length= 0

>Exit code: 0
```

*“Splat operator” (*)*

It can be use to split array and distribute it to specific arguments of the method



```ruby
# Working with an arbitrary number of arguments
# First is mandatory, rest is optional
def max(first, *rest)
  max=first
  rest.each{|x| max=x if x>max}
  max
end
def maxmin(first, *rest)
  max=min=first
  rest.each{|x| max=x if x>max;
                min=x if x<min }
  return max, min
end
def max2(first, second, *rest)
  max= first>second ? first : second
  rest.each{|x| max=x if x>max}
  max
end
def headtail(head,*tail)
  return head, tail
end

p max(1); p max(1,2);  p max2(1,2)
p maxmin(4,5,6,7,8,9)
p max([1,2,3]) # first=[1,2,3]; rest=[ ]
a = [1,2,3,4]
p max(*a) # first=1, rest=[2,3,4]
p maxmin(*[5,6,7,8,9])
p max2(*[5,6,7,8,9])
p headtail(*a); p headtail(a)
```

```
>ruby Max_VarPar.rb
1
2
2
[9, 4]
[1, 2, 3]
4
[9, 5]
9
[1, [2, 3, 4]]
[[1, 2, 3, 4], []]
>Exit code: 0
```

© Jozo Dujmović

Class Array methods

- Definition: array is an ordered, integer-indexed collection of any object
- Array indexing starts at 0
- Ruby uses circular indexing a[-1] = last element of array, a[-2] = element before a[-1]
- Array elements can have different data types
- Example of an array: [“a”, ”b, ”c”, 1, 2, 3]
- Array = array class name
- Array.new = class method that creates a new array object
- a = Array.new is equivalent to a = [ ]
- a[int] is object or nil
- a[start, length] = subarray a[start … start+length-s]
- a[p .. q] = subarray of given range a[p .. q]

MAKE

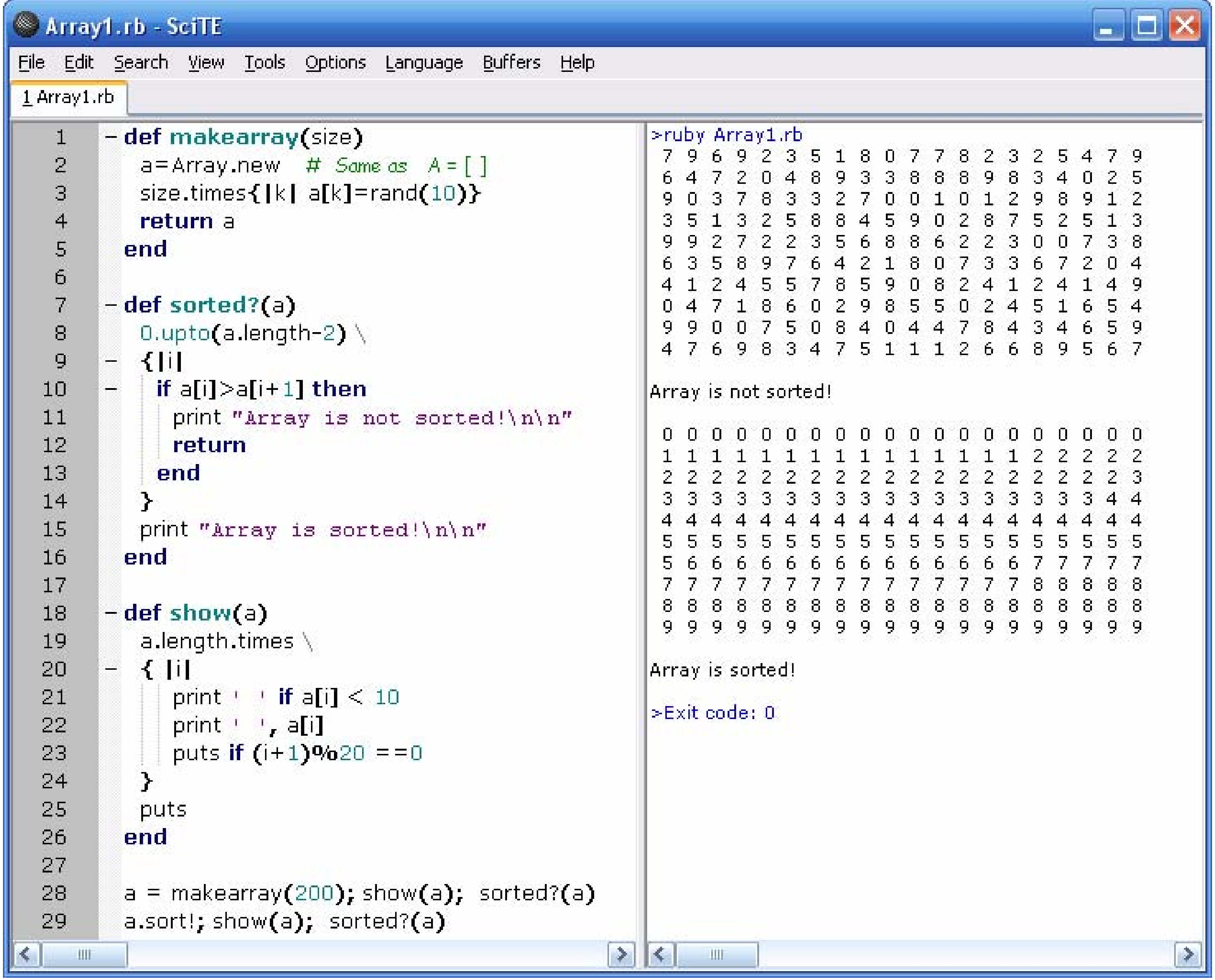
ARRAY

SORT

TEST

SHOW

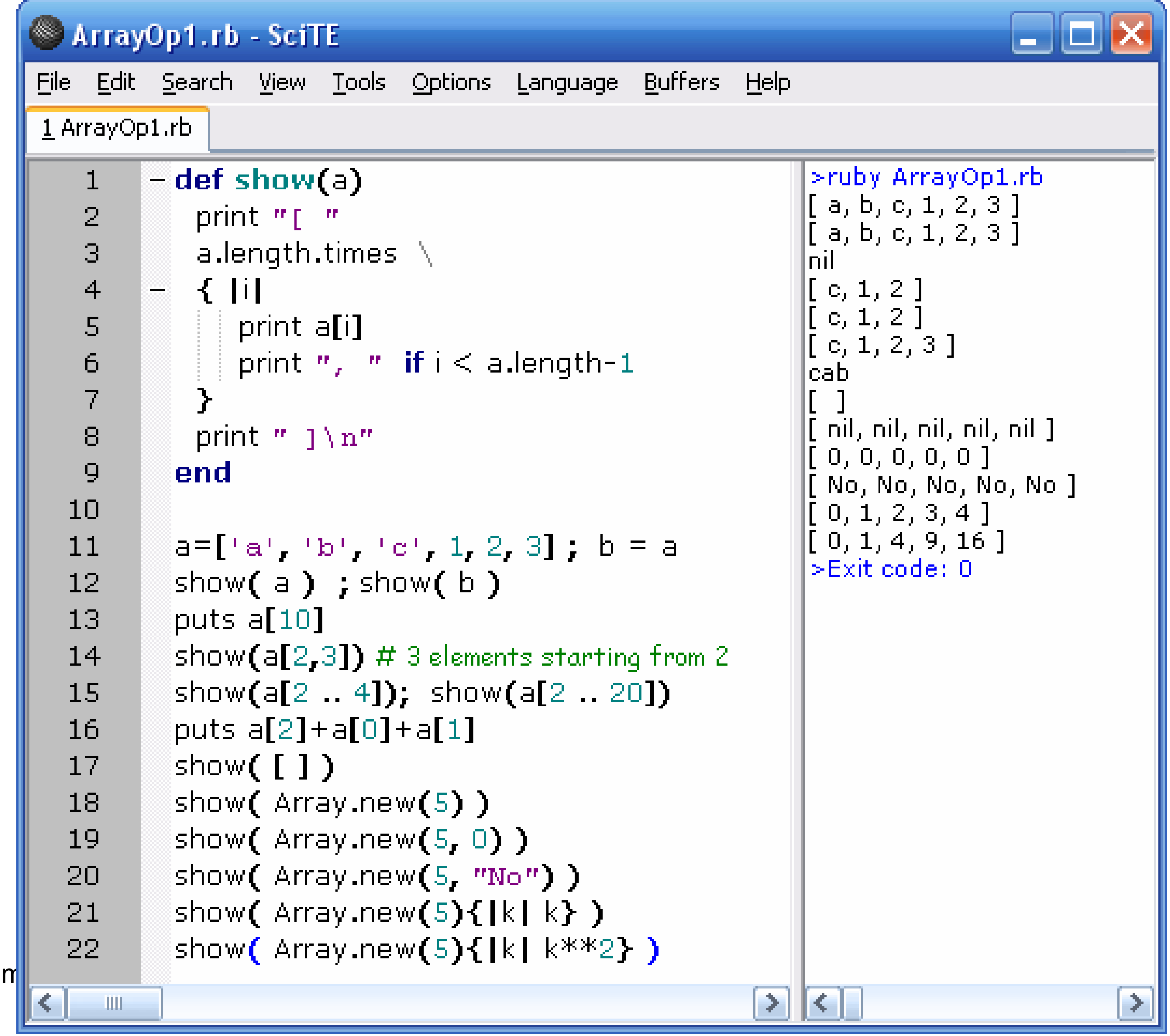
ARRAY



```ruby
def makearray(size)
  a=Array.new   #  Same as   A = [ ]
  size.times{|k| a[k]=rand(10)}
  return a
end

def sorted?(a)
  0.upto(a.length-2) \
  {|i|
    if a[i]>a[i+1] then
      print "Array is not sorted!\n\n"
      return
    end
  }
  print "Array is sorted!\n\n"
end

def show(a)
  a.length.times \
  { |i|
      print '  ' if a[i] < 10
      print '  ', a[i]
      puts if (i+1)%20 ==0
  }
  puts
end

a = makearray(200); show(a);  sorted?(a)
a.sort!; show(a);  sorted?(a)
```

```
>ruby Array1.rb
  7  9  6  9  2  3  5  1  8  0  7  7  8  2  3  2  5  4  7  9
  6  4  7  2  0  4  8  9  3  3  8  8  8  9  8  3  4  0  2  5
  9  0  3  7  8  3  3  2  7  0  0  1  0  1  2  9  8  9  1  2
  3  5  1  3  2  5  8  8  4  5  9  0  2  8  7  5  2  5  1  3
  9  9  2  7  2  2  3  5  6  8  8  6  2  2  3  0  0  7  3  8
  6  3  5  8  9  7  6  4  2  1  8  0  7  3  3  6  7  2  0  4
  4  1  2  4  5  5  7  8  5  9  0  8  2  4  1  2  4  1  4  9
  0  4  7  1  8  6  0  2  9  8  5  5  0  2  4  5  1  6  5  4
  9  9  0  0  7  5  0  8  4  0  4  4  7  8  4  3  4  6  5  9
  4  7  6  9  8  3  4  7  5  1  1  1  2  6  6  8  9  5  6  7

Array is not sorted!

  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  2  2  2  2  2
  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  3
  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  4  4
  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
  5  6  6  6  6  6  6  6  6  6  6  6  6  6  6  7  7  7  7  7
  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  8  8  8  8  8
  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8
  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9

Array is sorted!

>Exit code: 0
```

SHOW ARRAY METHOD



```ruby
def show(a)
  print "[  "
  a.length.times  \
  { |i|
    print a[i]
    print ",  "  if i < a.length-1
  }
  print "  ]\n"
end

a=['a', 'b', 'c', 1, 2, 3] ; b = a
show( a )  ; show( b )
puts a[10]
show(a[2,3]) # 3 elements starting from 2
show(a[2 .. 4]); show(a[2 .. 20])
puts a[2]+a[0]+a[1]
show( [ ] )
show( Array.new(5) )
show( Array.new(5, 0) )
show( Array.new(5, "No") )
show( Array.new(5){|k| k} )
show( Array.new(5){|k| k**2} )
```

```
>ruby ArrayOp1.rb
[ a, b, c, 1, 2, 3 ]
[ a, b, c, 1, 2, 3 ]
nil
[ c, 1, 2 ]
[ c, 1, 2 ]
[ c, 1, 2, 3 ]
cab
[  ]
[ nil, nil, nil, nil, nil ]
[ 0, 0, 0, 0, 0 ]
[ No, No, No, No, No ]
[ 0, 1, 2, 3, 4 ]
[ 0, 1, 4, 9, 16 ]
>Exit code: 0
```

Note:

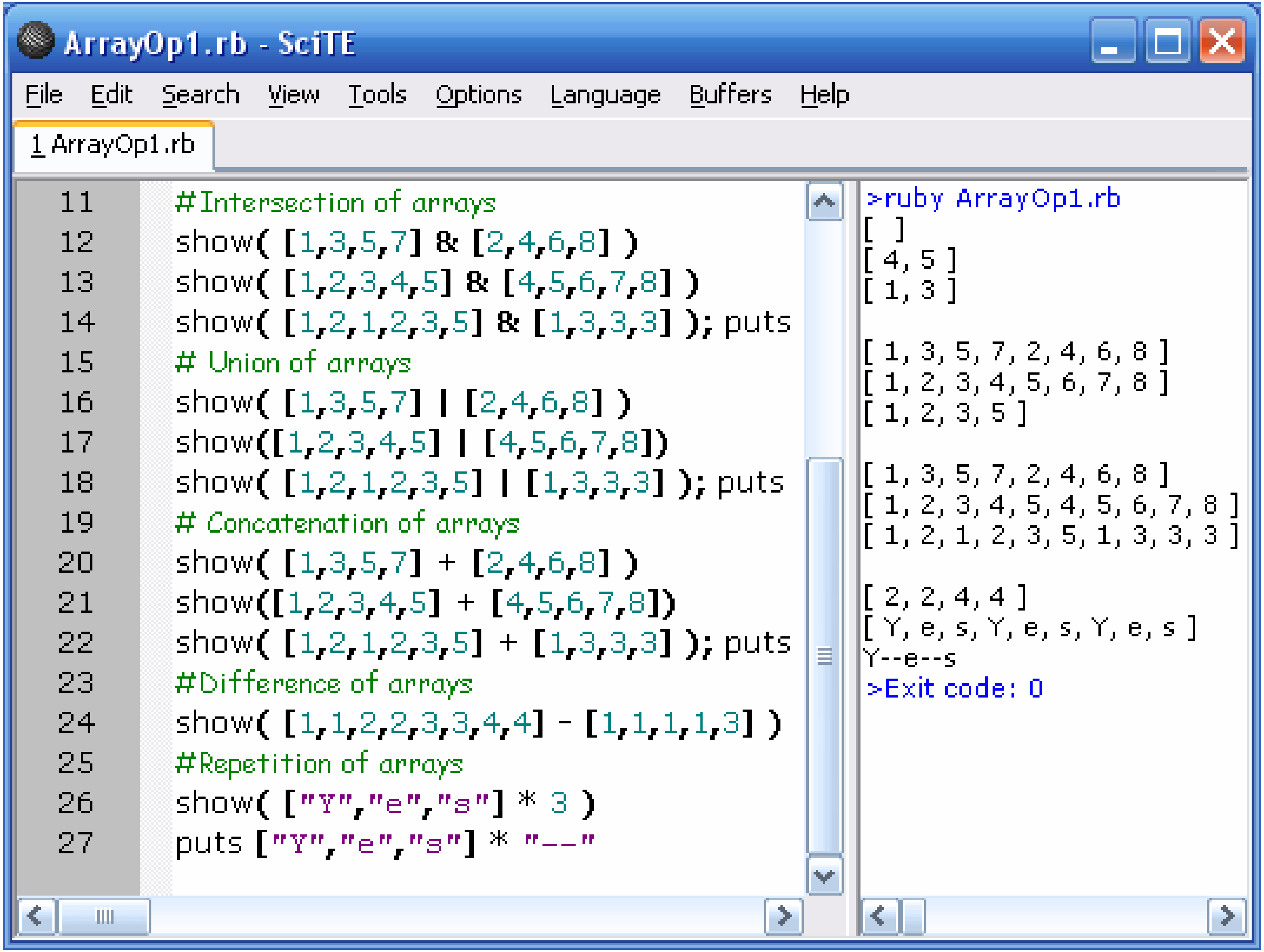
show(a) works similarly as the kernel method p

© Jozo Dujm

Basic array instance methods

- Intersection
- Union
- Repetition
- Concatenation
- Difference
- Append
- Compare a <=> b returns -1, 0, 1 for a<b, a=b, a>b
- Equality of arrays ==

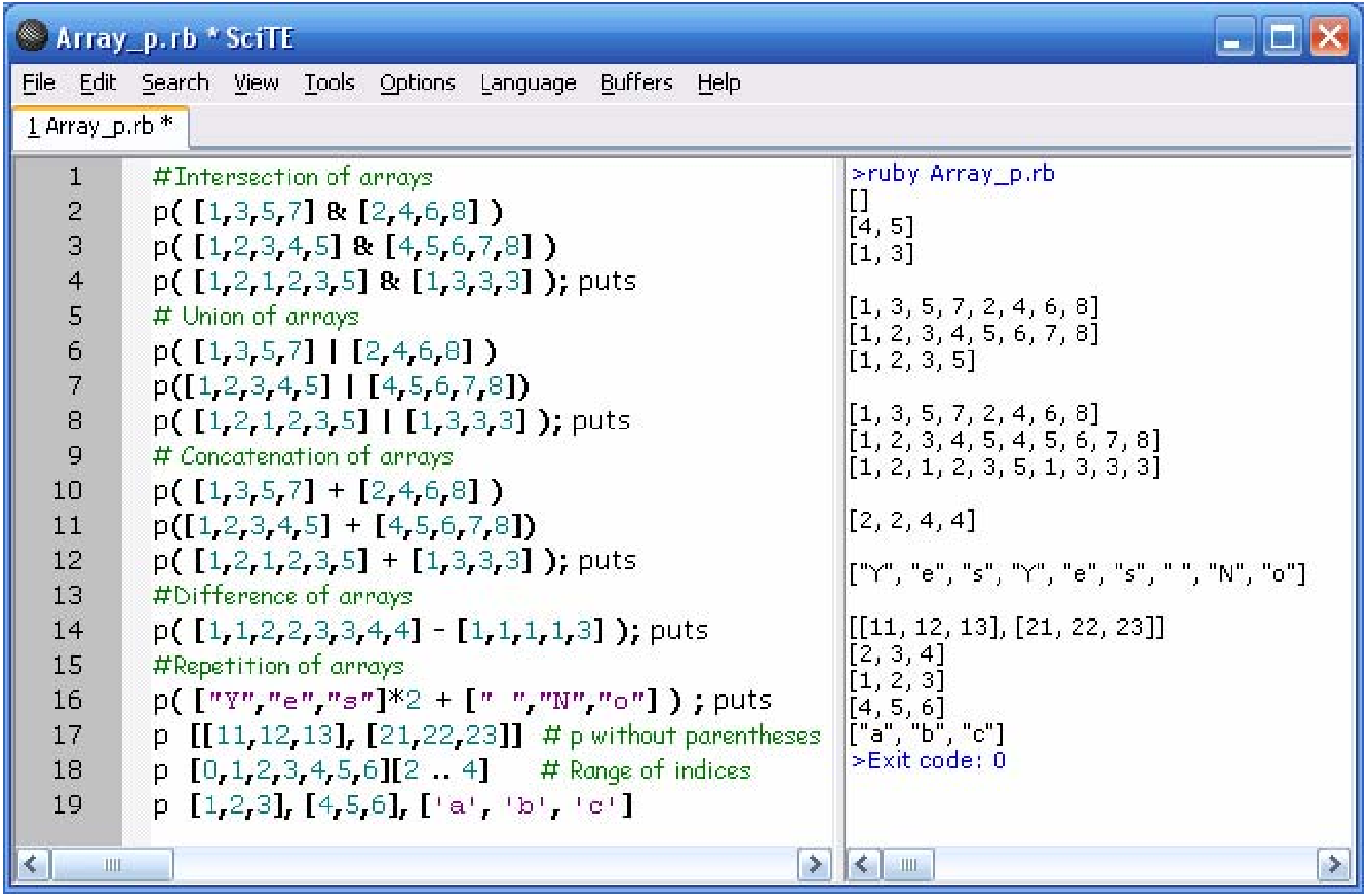
- Include
- Delete
- Clear
- Reverse
- First
- Each
- Sort
- transpose
- to_a
- to_s



```ruby
#Intersection of arrays
show( [1,3,5,7] & [2,4,6,8] )
show( [1,2,3,4,5] & [4,5,6,7,8] )
show( [1,2,1,2,3,5] & [1,3,3,3] ); puts
# Union of arrays
show( [1,3,5,7] | [2,4,6,8] )
show([1,2,3,4,5] | [4,5,6,7,8])
show( [1,2,1,2,3,5] | [1,3,3,3] ); puts
# Concatenation of arrays
show( [1,3,5,7] + [2,4,6,8] )
show([1,2,3,4,5] + [4,5,6,7,8])
show( [1,2,1,2,3,5] + [1,3,3,3] ); puts
#Difference of arrays
show( [1,1,2,2,3,3,4,4] - [1,1,1,1,3] )
#Repetition of arrays
show( ["Y","e","s"] * 3 )
puts ["Y","e","s"] * "--"
```

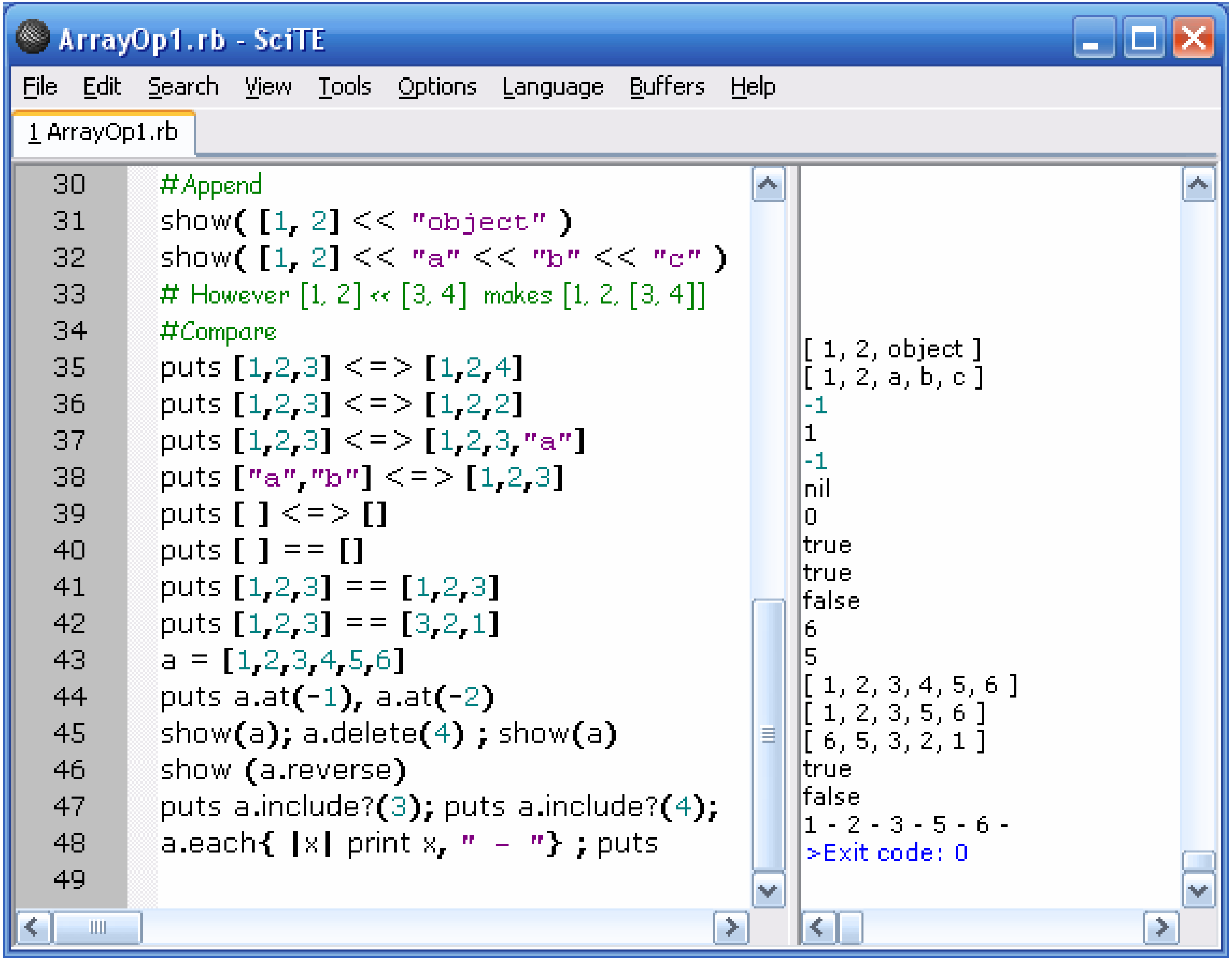
```
>ruby ArrayOp1.rb
[ ]
[ 4, 5 ]
[ 1, 3 ]

[ 1, 3, 5, 7, 2, 4, 6, 8 ]
[ 1, 2, 3, 4, 5, 6, 7, 8 ]
[ 1, 2, 3, 5 ]

[ 1, 3, 5, 7, 2, 4, 6, 8 ]
[ 1, 2, 3, 4, 5, 4, 5, 6, 7, 8 ]
[ 1, 2, 1, 2, 3, 5, 1, 3, 3, 3 ]

[ 2, 2, 4, 4 ]
[ Y, e, s, Y, e, s, Y, e, s ]
Y--e--s
>Exit code: 0
```

Kernel method p for printing



```
#Intersection of arrays
p( [1,3,5,7] & [2,4,6,8] )
p( [1,2,3,4,5] & [4,5,6,7,8] )
p( [1,2,1,2,3,5] & [1,3,3,3] ); puts
# Union of arrays
p( [1,3,5,7] | [2,4,6,8] )
p([1,2,3,4,5] | [4,5,6,7,8])
p( [1,2,1,2,3,5] | [1,3,3,3] ); puts
# Concatenation of arrays
p( [1,3,5,7] + [2,4,6,8] )
p([1,2,3,4,5] + [4,5,6,7,8])
p( [1,2,1,2,3,5] + [1,3,3,3] ); puts
#Difference of arrays
p( [1,1,2,2,3,3,4,4] - [1,1,1,1,3] ); puts
#Repetition of arrays
p( ["Y","e","s"]*2 + [" ","N","o"] ) ; puts
p  [[11,12,13], [21,22,23]]  # p without parentheses
p  [0,1,2,3,4,5,6][2 .. 4]      # Range of indices
p  [1,2,3], [4,5,6], ['a', 'b', 'c']
```

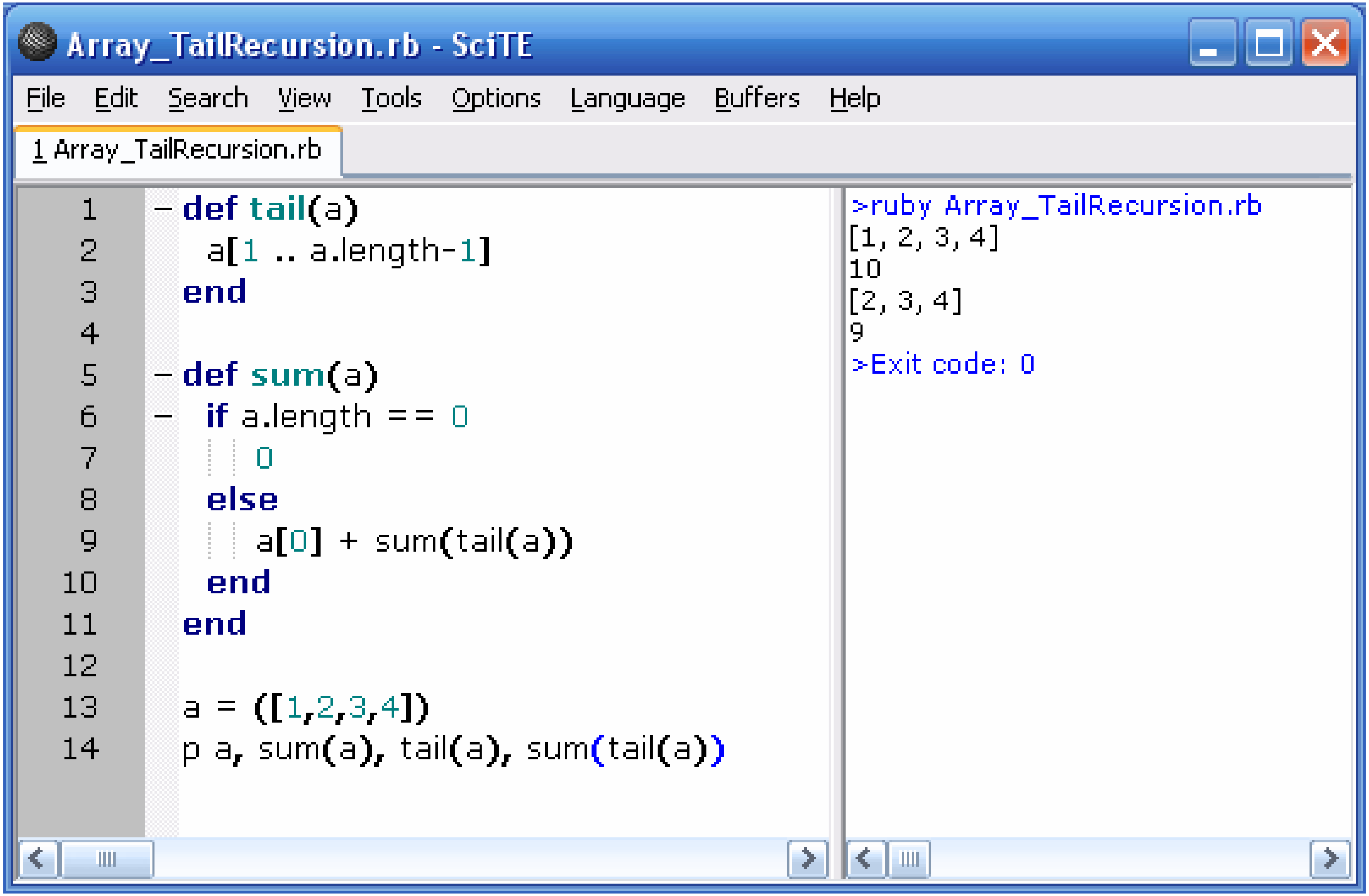
```
>ruby Array_p.rb
[]
[4, 5]
[1, 3]

[1, 3, 5, 7, 2, 4, 6, 8]
[1, 2, 3, 4, 5, 6, 7, 8]
[1, 2, 3, 5]

[1, 3, 5, 7, 2, 4, 6, 8]
[1, 2, 3, 4, 5, 4, 5, 6, 7, 8]
[1, 2, 1, 2, 3, 5, 1, 3, 3, 3]

[2, 2, 4, 4]

["Y", "e", "s", "Y", "e", "s", " ", "N", "o"]

[[11, 12, 13], [21, 22, 23]]
[2, 3, 4]
[1, 2, 3]
[4, 5, 6]
["a", "b", "c"]
>Exit code: 0
```



```ruby
#Append
show( [1, 2] << "object" )
show( [1, 2] << "a" << "b" << "c" )
# However [1, 2] << [3, 4]  makes [1, 2, [3, 4]]
#Compare
puts [1,2,3] <=> [1,2,4]
puts [1,2,3] <=> [1,2,2]
puts [1,2,3] <=> [1,2,3,"a"]
puts ["a","b"] <=> [1,2,3]
puts [ ] <=> []
puts [ ] == []
puts [1,2,3] == [1,2,3]
puts [1,2,3] == [3,2,1]
a = [1,2,3,4,5,6]
puts a.at(-1), a.at(-2)
show(a); a.delete(4) ; show(a)
show (a.reverse)
puts a.include?(3); puts a.include?(4);
a.each{ |x| print x, " - "} ; puts
```

```
[ 1, 2, object ]
[ 1, 2, a, b, c ]
-1
1
-1
nil
0
true
true
false
6
5
[ 1, 2, 3, 4, 5, 6 ]
[ 1, 2, 3, 5, 6 ]
[ 6, 5, 3, 2, 1 ]
true
false
1 - 2 - 3 - 5 - 6 -
>Exit code: 0
```

Tail recursion in Ruby



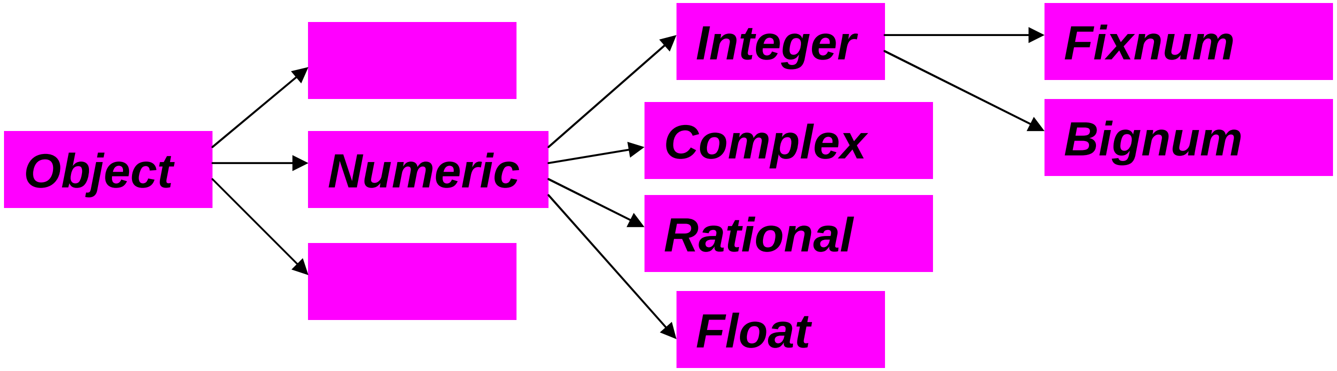
Array_TailRecursion.rb - SciTE
File Edit Search View Tools Options Language Buffers Help
1 Array_TailRecursion.rb

```
def tail(a)
  a[1 .. a.length-1]
end

def sum(a)
  if a.length == 0
    0
  else
    a[0] + sum(tail(a))
  end
end

a = ([1,2,3,4])
p a, sum(a), tail(a), sum(tail(a))
```

```
>ruby Array_TailRecursion.rb
[1, 2, 3, 4]
10
[2, 3, 4]
9
>Exit code: 0
```

Classes

- Class is a container for *variables* and *methods* (“*properties*”)
- Class can inherit properties from a single parent class (no multiple inheritance)
- The base (root) class in Ruby is **Object**
- Classes are always open (it is possible to add to any class including the build-in classes)

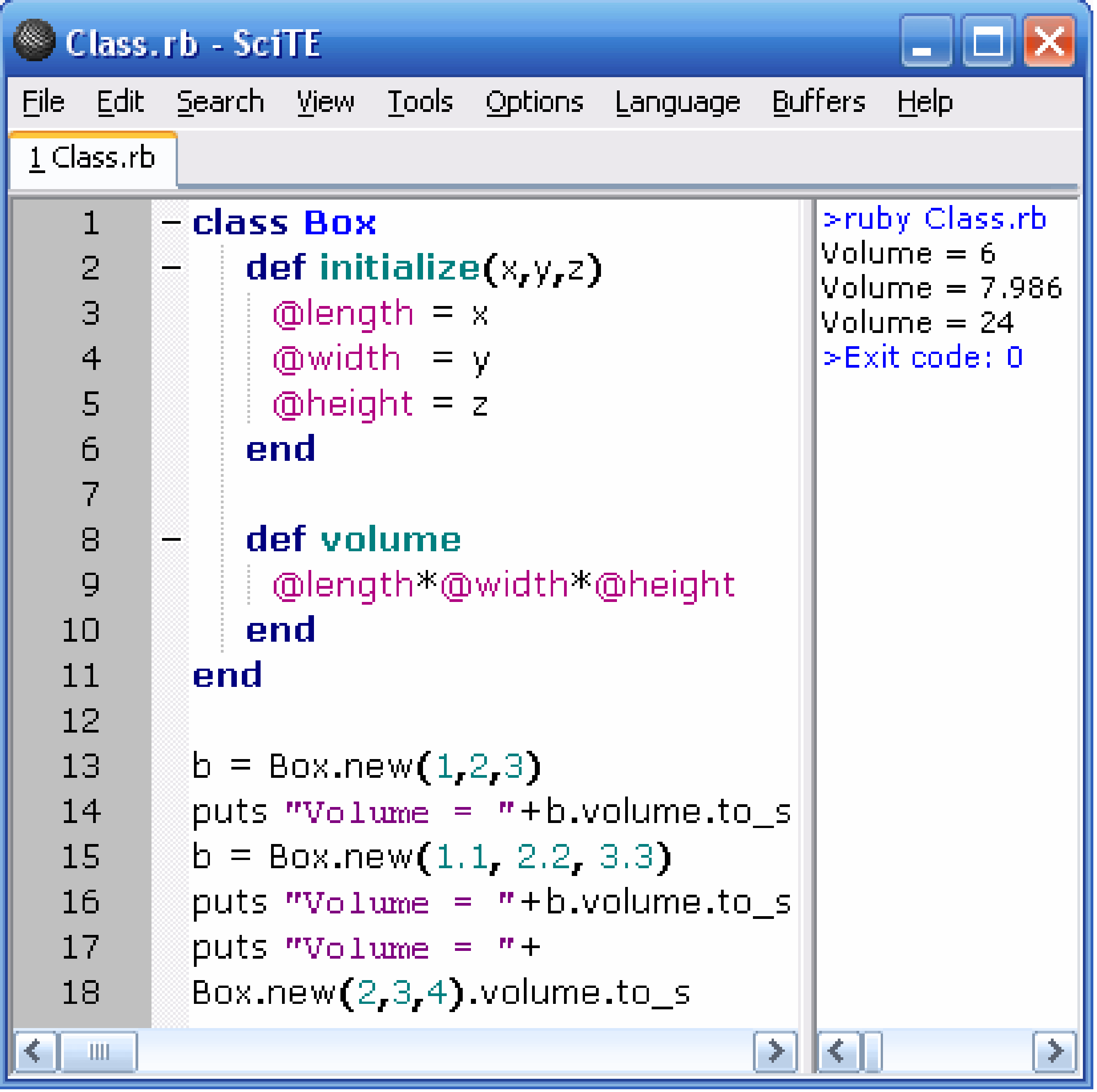


Classes - Syntax

- Basic syntax:

```
class <Classname>
……………………….
end
```

- Classname must be CONSTANT (i.e. it must beging with a capital letter)
- @var = instance variable (private variable reachable using an accessor method)
- @@var = class variable (shared among all instances of a class)



```ruby
class Box
  def initialize(x,y,z)
    @length = x
    @width  = y
    @height = z
  end

  def volume
    @length*@width*@height
  end
end

b = Box.new(1,2,3)
puts "Volume = "+b.volume.to_s
b = Box.new(1.1, 2.2, 3.3)
puts "Volume = "+b.volume.to_s
puts "Volume = "+
Box.new(2,3,4).volume.to_s
```

```
>ruby Class.rb
Volume = 6
Volume = 7.986
Volume = 24
>Exit code: 0
```

The instance variables @length, @width, @height are private attributes (descriptors) of each instance of the Box object.

The initialize method is the constructor-initializer of an instance. Its role is to reserve memory space for instance variables @length, @width, @height, and to instantiate their values using the x,y,z arguments. Initialize is activated by Box.new(…).

Box.new(1,2,3) is a constant anonymous object (a nameless set of initialized @length, @width, @height instance variables).

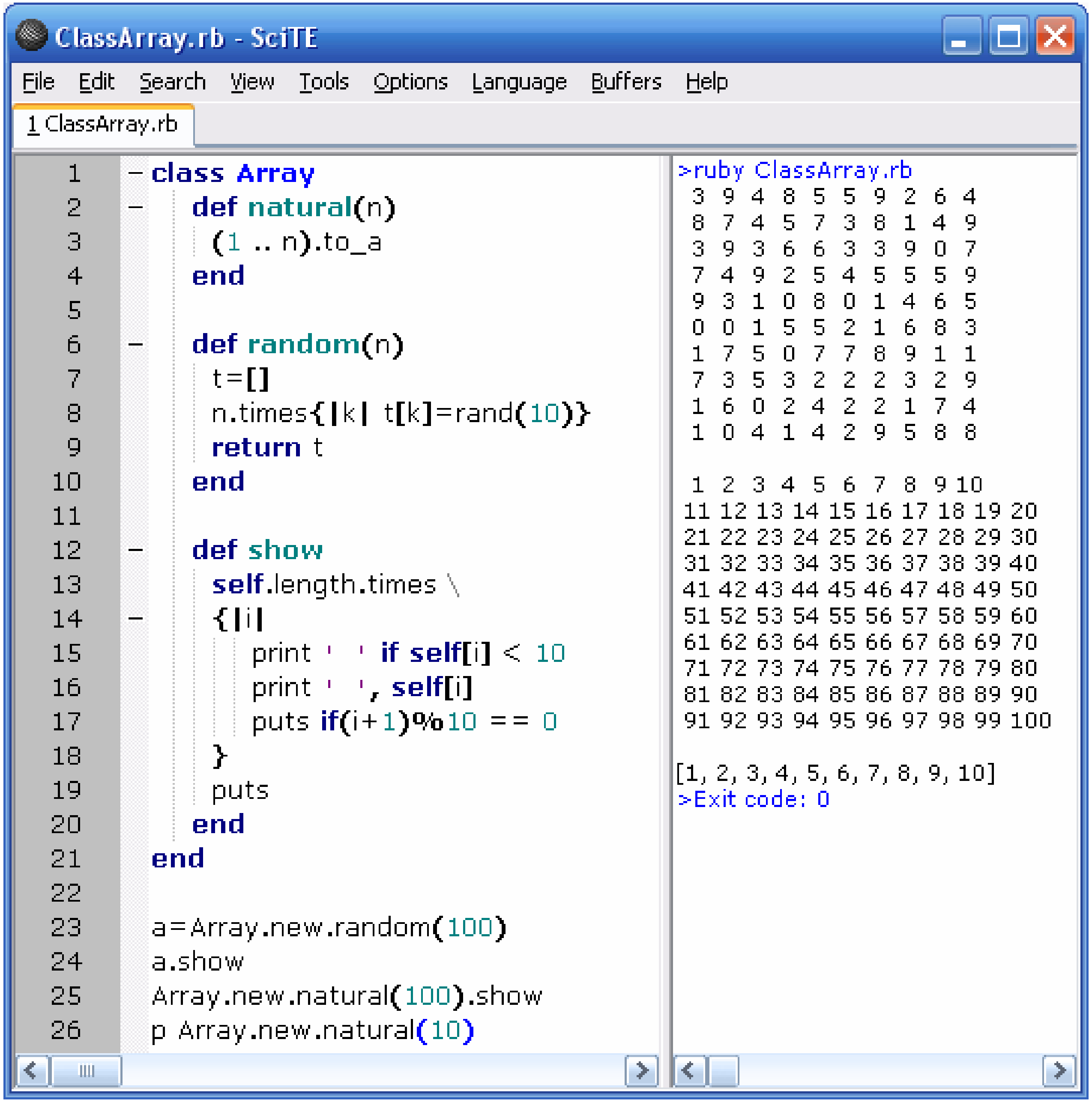
The instance variables are accessible to all class member functions (they act as global variables for the member functions)

Expanding existing classes

- All existing classes come with a number of methods. E.g. Array includes methods:

  &, *, +, -, <<, <=>, ==, [ ], [ ]=, abbrev, assoc, at, clear, collect, collect!, compact, compact!, concat, dclone, delete, delete_at, delete_if, each, each_index, empty?, eql?, fetch, fill, first, flatten, flatten!, frozen?, hash, include?, index, indexes, indices, initialize_copy, insert, inspect, join, last, length, map, map!, nitems, pack, pop, pretty_print, pretty_print_cycle, push, quote, rassoc, reject, reject!, replace, reverse, reverse!, reverse_each, rindex, select, shift, size, slice, slice!, sort, sort!, to_a, to_ary, to_s, to_yaml, transpose, uniq, uniq!, unshift, values_at, yaml_initialize, zip, |

- Users can expand the collection of methods of existing classes by adding new methods
- Example: add special initialization and display methods for arrays



```ruby
class Array
  def natural(n)
    (1 .. n).to_a
  end

  def random(n)
    t=[]
    n.times{|k| t[k]=rand(10)}
    return t
  end

  def show
    self.length.times \
    {|i|
      print ' ' if self[i] < 10
      print ' ', self[i]
      puts if(i+1)%10 == 0
    }
    puts
  end
end

a=Array.new.random(100)
a.show
Array.new.natural(100).show
p Array.new.natural(10)
```

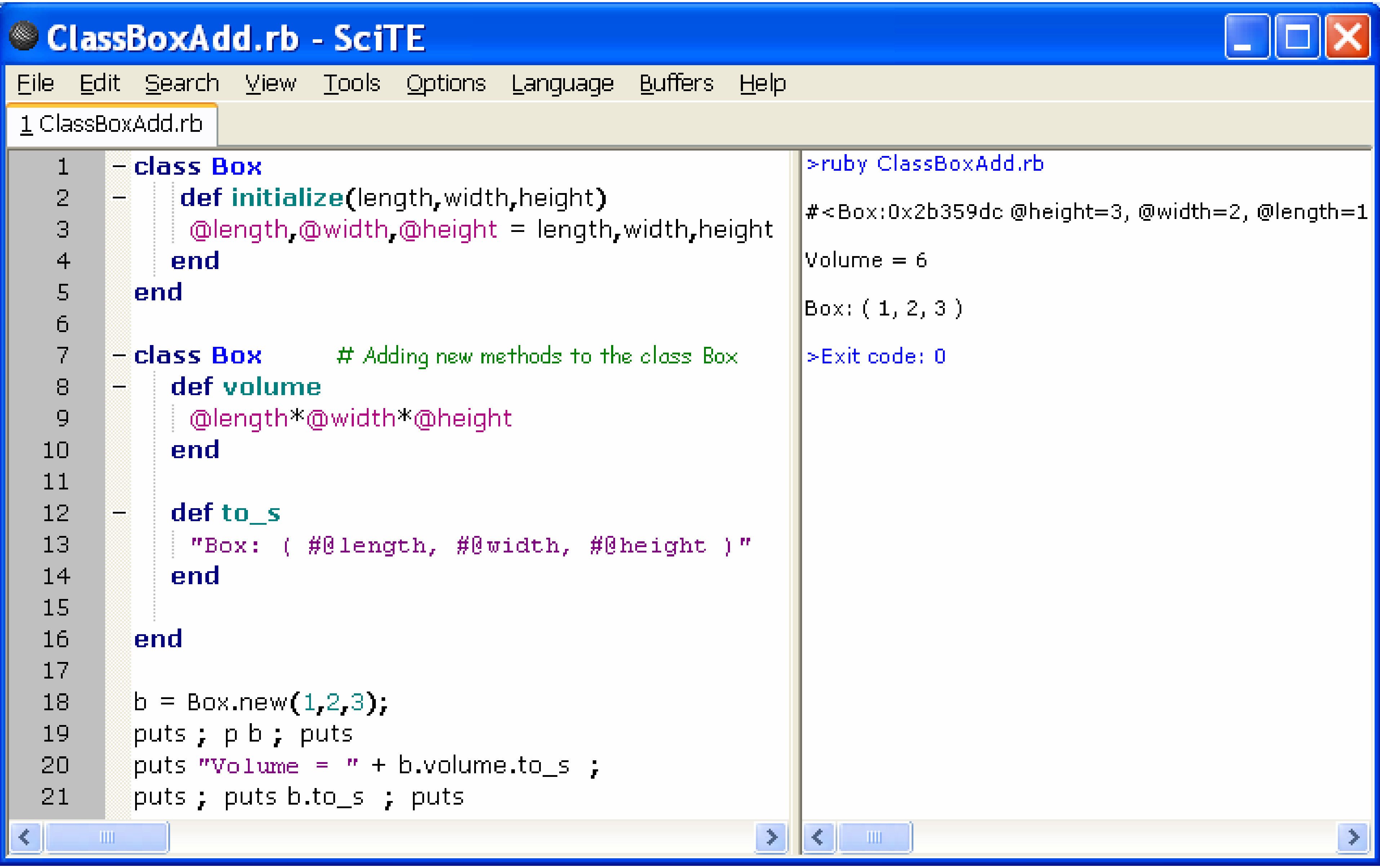
```
>ruby ClassArray.rb
  3  9  4  8  5  5  9  2  6  4
  8  7  4  5  7  3  8  1  4  9
  3  9  3  6  6  3  3  9  0  7
  7  4  9  2  5  4  5  5  5  9
  9  3  1  0  8  0  1  4  6  5
  0  0  1  5  5  2  1  6  8  3
  1  7  5  0  7  7  8  9  1  1
  7  3  5  3  2  2  2  3  2  9
  1  6  0  2  4  2  2  1  7  4
  1  0  4  1  4  2  9  5  8  8

  1  2  3  4  5  6  7  8  9 10
 11 12 13 14 15 16 17 18 19 20
 21 22 23 24 25 26 27 28 29 30
 31 32 33 34 35 36 37 38 39 40
 41 42 43 44 45 46 47 48 49 50
 51 52 53 54 55 56 57 58 59 60
 61 62 63 64 65 66 67 68 69 70
 71 72 73 74 75 76 77 78 79 80
 81 82 83 84 85 86 87 88 89 90
 91 92 93 94 95 96 97 98 99 100

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>Exit code: 0
```

When expanding an existing system class (Array) the initialization of the object is out of reach of user.

However, the current object (array) can be manipulated using the pseudovariable self .

By defining new methods inside existing classes, we can expand and improve existing method libraries and add new functionality.

Incremental build of the class Box

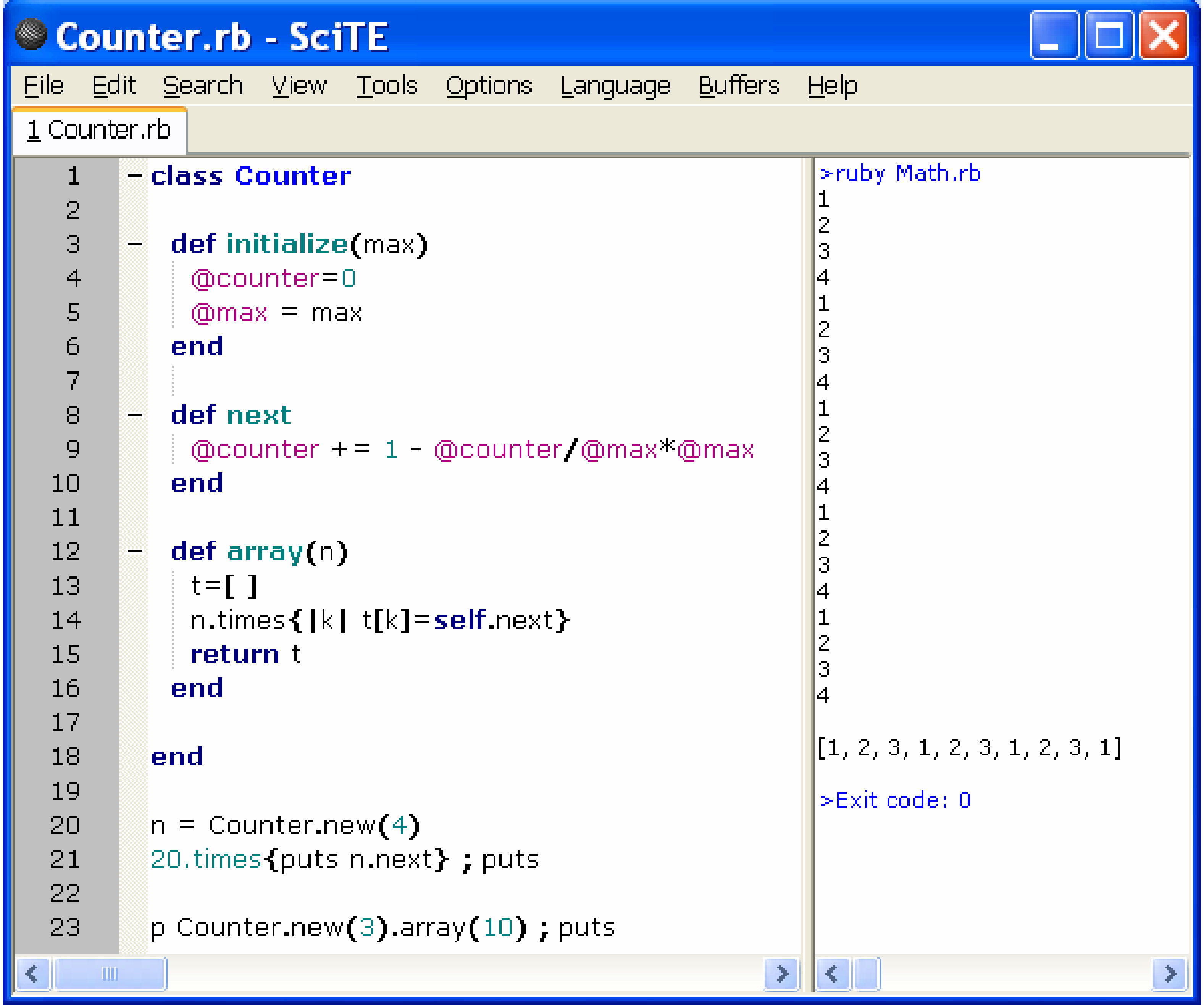


```ruby
class Box
   def initialize(length,width,height)
     @length,@width,@height = length,width,height
   end
end

class Box          # Adding new methods to the class Box
  def volume
     @length*@width*@height
  end

  def to_s
     "Box: ( #@length, #@width, #@height )"
  end

end

b = Box.new(1,2,3);
puts ; p b ; puts
puts "Volume = " + b.volume.to_s ;
puts ; puts b.to_s ; puts
```

```
>ruby ClassBoxAdd.rb

#<Box:0x2b359dc @height=3, @width=2, @length=1

Volume = 6

Box: ( 1, 2, 3 )

>Exit code: 0
```

The concept of pseudovariable

- Properties of pseudovariables:
  - They look as all other variables
  - They act as constants (cannot be assigned a value)
- Ruby pseudovariables:
  - **self** = current object, invoked by a method
  - **true** = logical true; an instance of TrueClass
  - **false** = logical false; an instance of FalseClass
  - **nil** = empty/uninitialized/invalid; an instance of NillClass
  - **__FILE__** = name of current source file
  - **__LINE__** = number of current line in the current source file



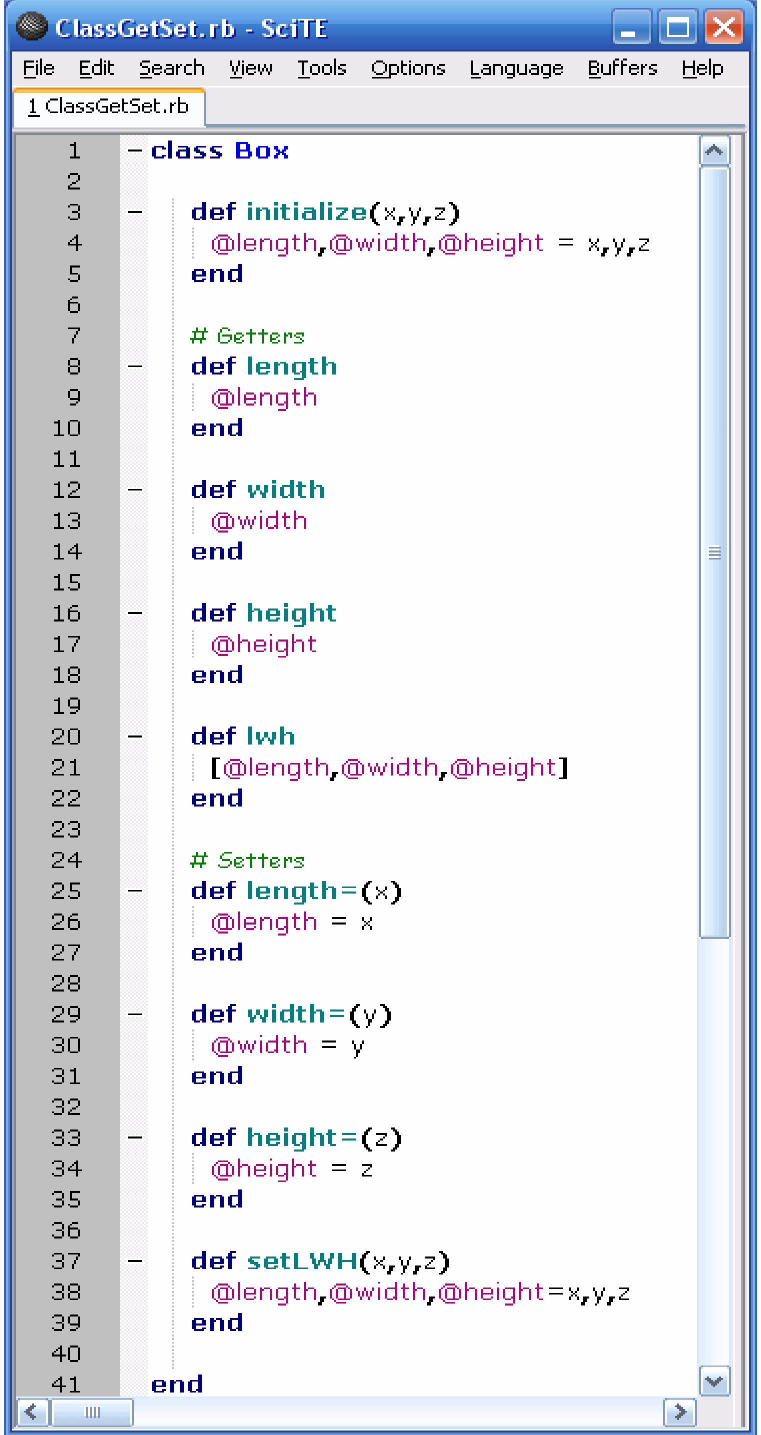
```ruby
class Counter

  def initialize(max)
    @counter=0
    @max = max
  end

  def next
    @counter += 1 - @counter/@max*@max
  end

  def array(n)
    t=[ ]
    n.times{|k| t[k]=self.next}
    return t
  end

end

n = Counter.new(4)
20.times{puts n.next} ; puts

p Counter.new(3).array(10) ; puts
```

```
>ruby Math.rb
1
2
3
4
1
2
3
4
1
2
3
4
1
2
3
4
1
2
3
4

[1, 2, 3, 1, 2, 3, 1, 2, 3, 1]

>Exit code: 0
```

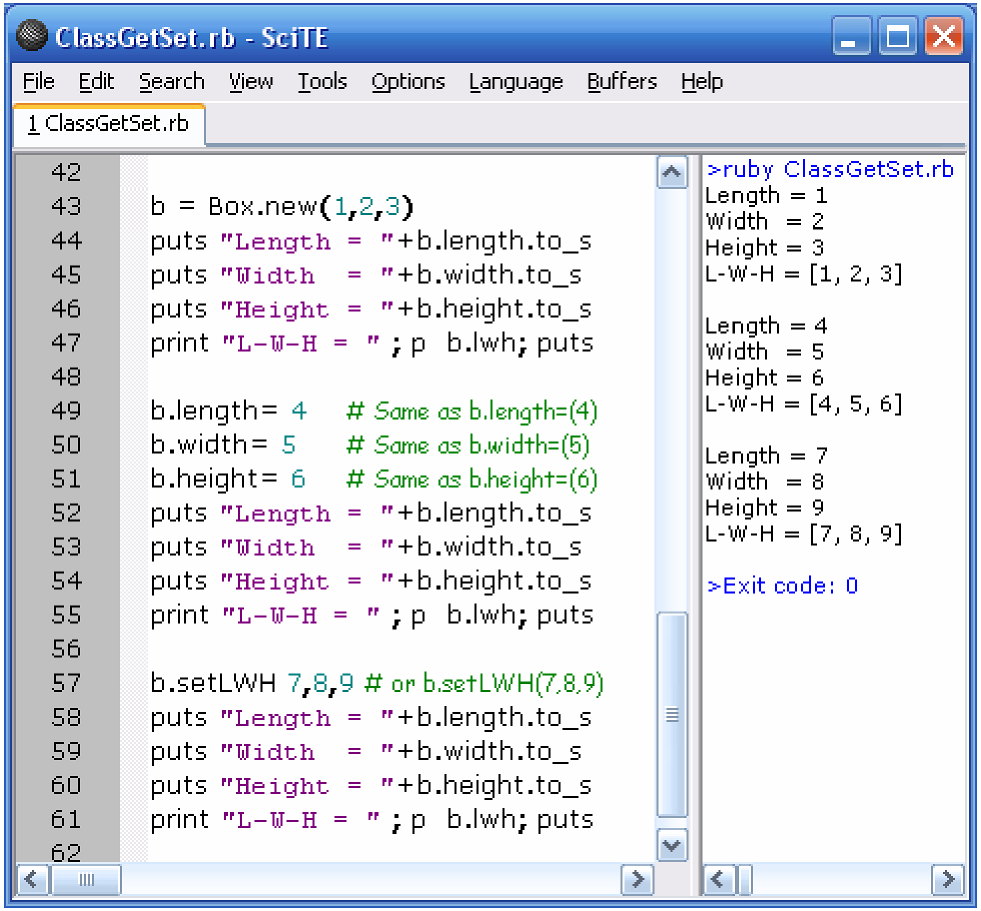
Getters and setters

- Getter = a method for reading instance variables (object attributes) from programs that are not class members
- Setter = a method for writing instance variables (object attributes) from programs that are not class members
- Ruby convention: setter name usually terminates with “=“

Getters and setters for the class Box



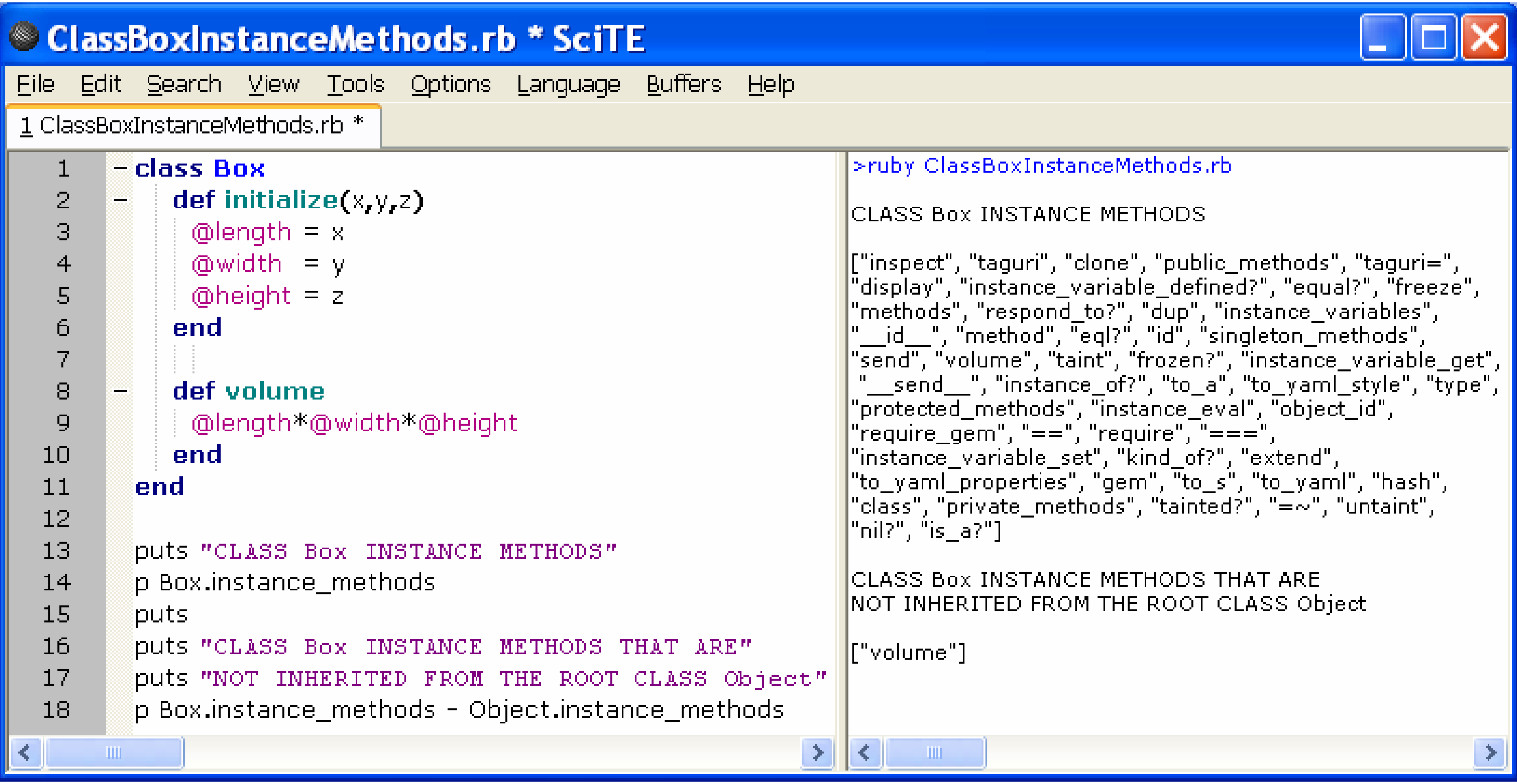
```
class Box

  def initialize(x,y,z)
    @length,@width,@height = x,y,z
  end

  # Getters
  def length
    @length
  end

  def width
    @width
  end

  def height
    @height
  end

  def lwh
    [@length,@width,@height]
  end

  # Setters
  def length=(x)
    @length = x
  end

  def width=(y)
    @width = y
  end

  def height=(z)
    @height = z
  end

  def setLWH(x,y,z)
    @length,@width,@height=x,y,z
  end

end
```



```

b = Box.new(1,2,3)
puts "Length = "+b.length.to_s
puts "Width  = "+b.width.to_s
puts "Height = "+b.height.to_s
print "L-W-H = " ; p  b.lwh; puts

b.length= 4    # Same as b.length=(4)
b.width= 5     # Same as b.width=(5)
b.height= 6    # Same as b.height=(6)
puts "Length = "+b.length.to_s
puts "Width  = "+b.width.to_s
puts "Height = "+b.height.to_s
print "L-W-H = " ; p  b.lwh; puts

b.setLWH 7,8,9 # or b.setLWH(7,8,9)
puts "Length = "+b.length.to_s
puts "Width  = "+b.width.to_s
puts "Height = "+b.height.to_s
print "L-W-H = " ; p  b.lwh; puts

```

```
>ruby ClassGetSet.rb
Length = 1
Width  = 2
Height = 3
L-W-H = [1, 2, 3]

Length = 4
Width  = 5
Height = 6
L-W-H = [4, 5, 6]

Length = 7
Width  = 8
Height = 9
L-W-H = [7, 8, 9]

>Exit code: 0
```

Classes inherit a basic set of instance methods from the root class Object. In the case of the class Box the only method that is not inherited is “volume”


ClassBoxInstanceMethods.rb * SciTE
File
Edit
Search
View
Tools
Options
Language
Buffers
Help
1 ClassBoxInstanceMethods.rb *

```ruby
class Box
  def initialize(x,y,z)
    @length = x
    @width  = y
    @height = z
  end

  def volume
    @length*@width*@height
  end
end

puts "CLASS Box INSTANCE METHODS"
p Box.instance_methods
puts
puts "CLASS Box INSTANCE METHODS THAT ARE"
puts "NOT INHERITED FROM THE ROOT CLASS Object"
p Box.instance_methods - Object.instance_methods
```

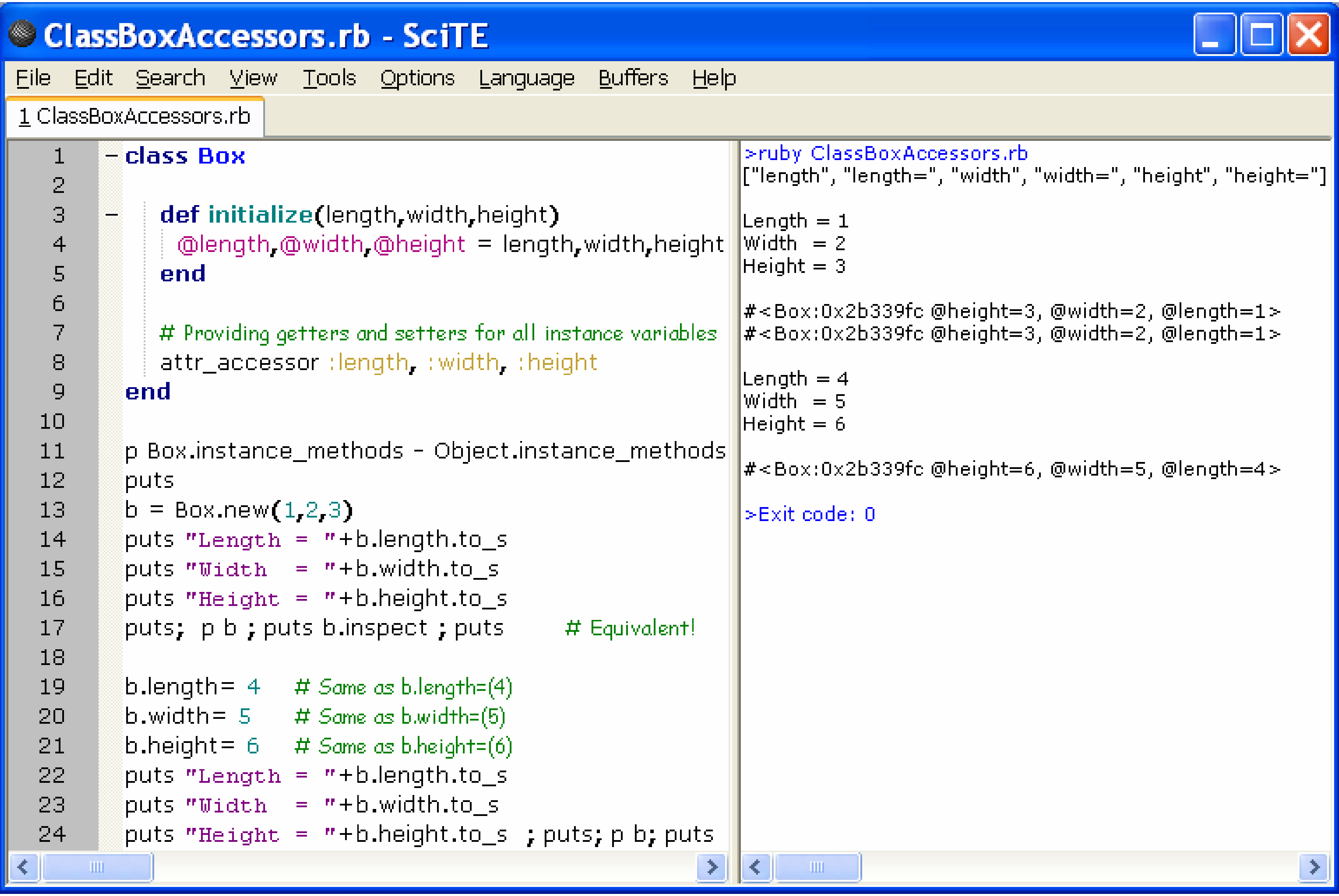
```
>ruby ClassBoxInstanceMethods.rb

CLASS Box INSTANCE METHODS

["inspect", "taguri", "clone", "public_methods", "taguri=",
"display", "instance_variable_defined?", "equal?", "freeze",
"methods", "respond_to?", "dup", "instance_variables",
"__id__", "method", "eql?", "id", "singleton_methods",
"send", "volume", "taint", "frozen?", "instance_variable_get",
 "__send__", "instance_of?", "to_a", "to_yaml_style", "type",
"protected_methods", "instance_eval", "object_id",
"require_gem", "==", "require", "===",
"instance_variable_set", "kind_of?", "extend",
"to_yaml_properties", "gem", "to_s", "to_yaml", "hash",
"class", "private_methods", "tainted?", "=~", "untaint",
"nil?", "is_a?"]

CLASS Box INSTANCE METHODS THAT ARE
NOT INHERITED FROM THE ROOT CLASS Object

["volume"]
```

Accessors for instance variables

- Let ivar be an instance variable of an object obj. Ruby provides the following four accessors (getters and setters):

attr :ivar [, true]	=>	obj.ivar [obj.ivar=]
attr_reader :ivar	=>	obj.ivar
attr_writer :ivar	=>	obj.ivar=
attr_accessor :ivar	=>	obj.ivar , obj.ivar=

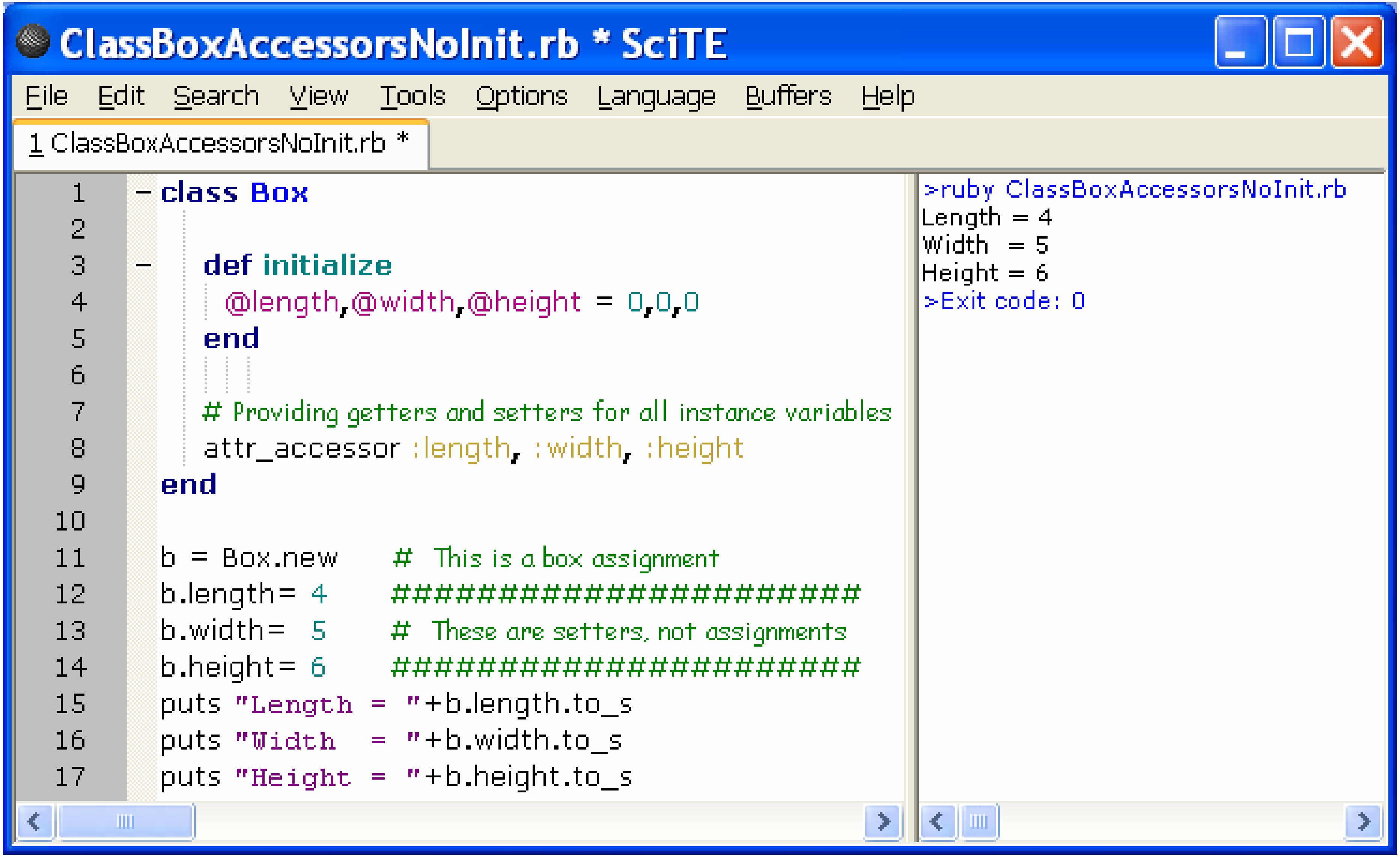
- attr_accessor is a general solution



```ruby
class Box

  def initialize(length,width,height)
    @length,@width,@height = length,width,height
  end

  # Providing getters and setters for all instance variables
  attr_accessor :length, :width, :height
end

p Box.instance_methods - Object.instance_methods
puts
b = Box.new(1,2,3)
puts "Length = "+b.length.to_s
puts "Width  = "+b.width.to_s
puts "Height = "+b.height.to_s
puts;  p b ; puts b.inspect ; puts      # Equivalent!

b.length= 4    # Same as b.length=(4)
b.width= 5     # Same as b.width=(5)
b.height= 6    # Same as b.height=(6)
puts "Length = "+b.length.to_s
puts "Width  = "+b.width.to_s
puts "Height = "+b.height.to_s  ; puts; p b; puts
```

```
>ruby ClassBoxAccessors.rb
["length", "length=", "width", "width=", "height", "height="]

Length = 1
Width  = 2
Height = 3

#<Box:0x2b339fc @height=3, @width=2, @length=1>
#<Box:0x2b339fc @height=3, @width=2, @length=1>

Length = 4
Width  = 5
Height = 6

#<Box:0x2b339fc @height=6, @width=5, @length=4>

>Exit code: 0
```

In a special case the initializer can be used without arguments

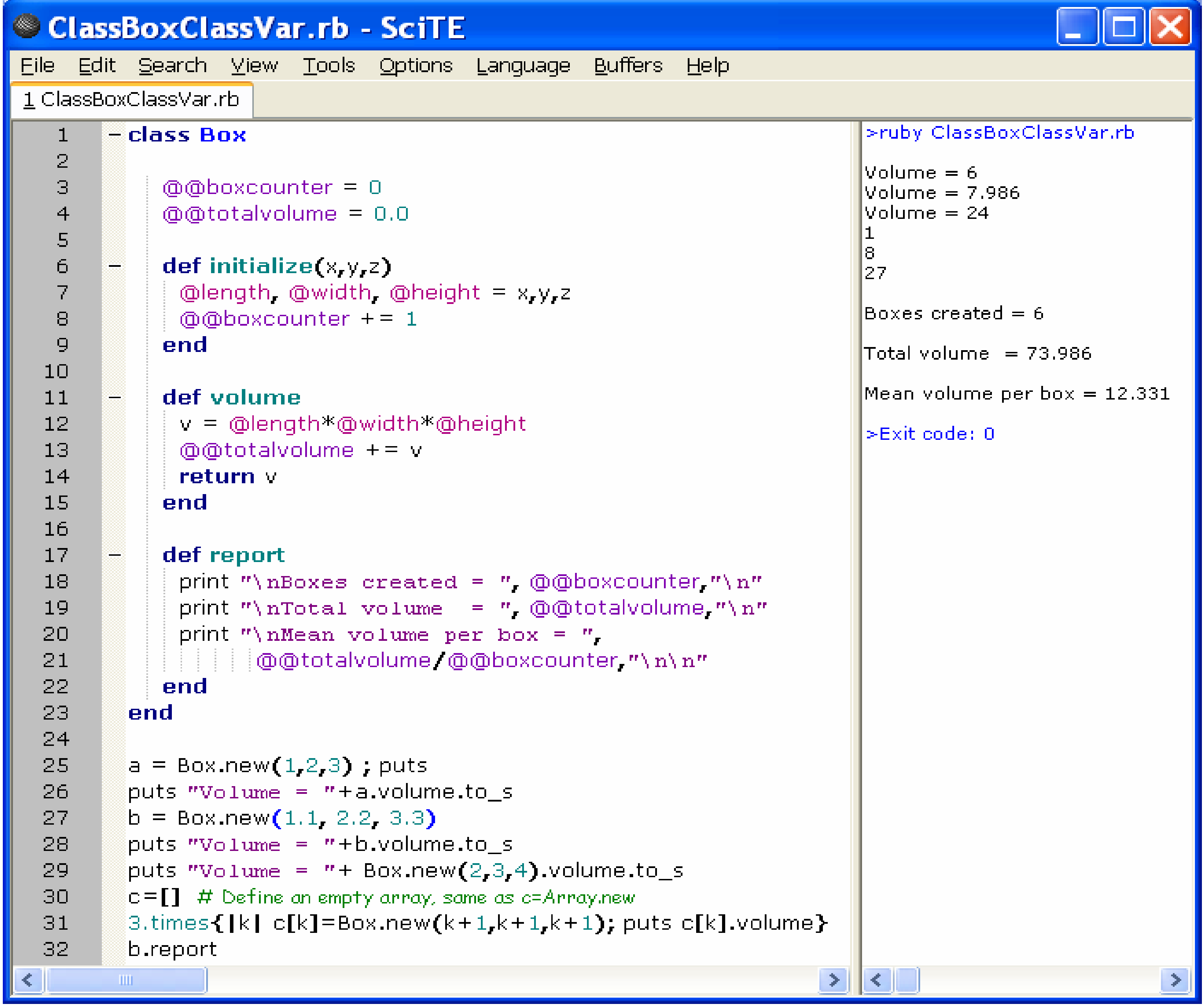

ClassBoxAccessorsNoInit.rb * SciTE
File Edit Search View Tools Options Language Buffers Help
1 ClassBoxAccessorsNoInit.rb *

```
class Box

  def initialize
    @length,@width,@height = 0,0,0
  end

  # Providing getters and setters for all instance variables
  attr_accessor :length, :width, :height
end

b = Box.new    #  This is a box assignment
b.length= 4    #######################
b.width=  5    #  These are setters, not assignments
b.height= 6    #######################
puts "Length = "+b.length.to_s
puts "Width  = "+b.width.to_s
puts "Height = "+b.height.to_s
```

```
>ruby ClassBoxAccessorsNoInit.rb
Length = 4
Width  = 5
Height = 6
>Exit code: 0
```

Class variables

- Class variables are prefixed by @@
- All instances share the same @@classvar
- Class variables are suitable as counters and accumulator
- If used as accumulators or counters the class variables must be initialized before their use

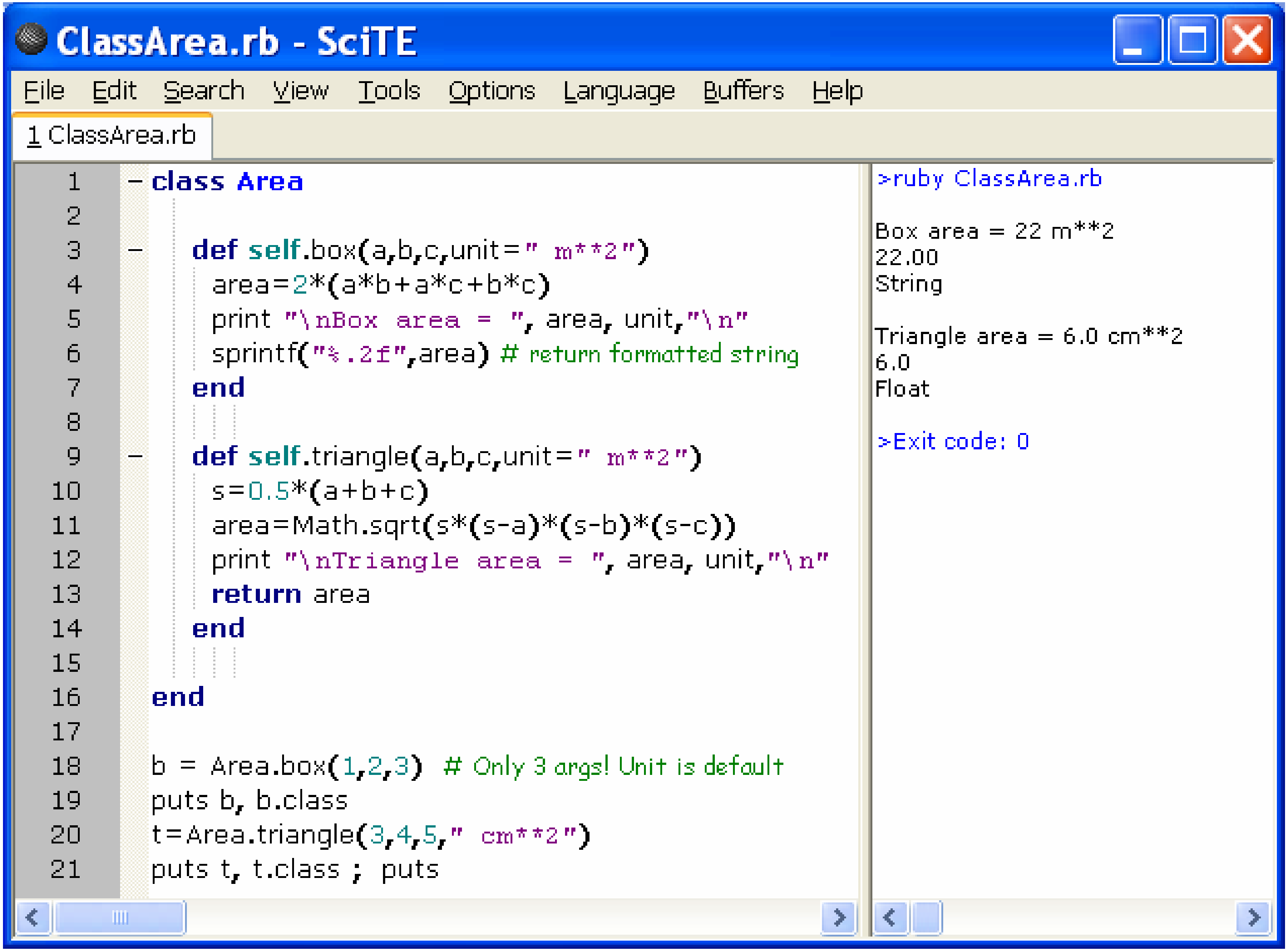

ClassBoxClassVar.rb - SciTE
File Edit Search View Tools Options Language Buffers Help
1 ClassBoxClassVar.rb

```ruby
class Box

  @@boxcounter = 0
  @@totalvolume = 0.0

  def initialize(x,y,z)
    @length, @width, @height = x,y,z
    @@boxcounter += 1
  end

  def volume
    v = @length*@width*@height
    @@totalvolume += v
    return v
  end

  def report
    print "\nBoxes created = ", @@boxcounter,"\n"
    print "\nTotal volume  = ", @@totalvolume,"\n"
    print "\nMean volume per box = ",
          @@totalvolume/@@boxcounter,"\n\n"
  end
end

a = Box.new(1,2,3) ; puts
puts "Volume = "+a.volume.to_s
b = Box.new(1.1, 2.2, 3.3)
puts "Volume = "+b.volume.to_s
puts "Volume = "+ Box.new(2,3,4).volume.to_s
c=[] # Define an empty array, same as c=Array.new
3.times{|k| c[k]=Box.new(k+1,k+1,k+1); puts c[k].volume}
b.report
```

```
>ruby ClassBoxClassVar.rb

Volume = 6
Volume = 7.986
Volume = 24
1
8
27

Boxes created = 6

Total volume  = 73.986

Mean volume per box = 12.331

>Exit code: 0
```

Class (static) methods

- Class method belongs to a class
- Class method is not associated with an instance of a class
- Class method is defined prefixed with **self** (which needs never to be changed) or the **name of the class to which it belongs** (which might be changed)
- Class method is invoked prefixed with the name of the class to which it belongs, like Math.sqrt(x)



```ruby
class Area

  def self.box(a,b,c,unit=" m**2")
   area=2*(a*b+a*c+b*c)
   print "\nBox area = ", area, unit,"\n"
   sprintf("%.2f",area) # return formatted string
  end

  def self.triangle(a,b,c,unit=" m**2")
   s=0.5*(a+b+c)
   area=Math.sqrt(s*(s-a)*(s-b)*(s-c))
   print "\nTriangle area = ", area, unit,"\n"
   return area
  end

end

b = Area.box(1,2,3)  # Only 3 args! Unit is default
puts b, b.class
t=Area.triangle(3,4,5," cm**2")
puts t, t.class ;  puts
```

```
>ruby ClassArea.rb

Box area = 22 m**2
22.00
String

Triangle area = 6.0 cm**2
6.0
Float

>Exit code: 0
```

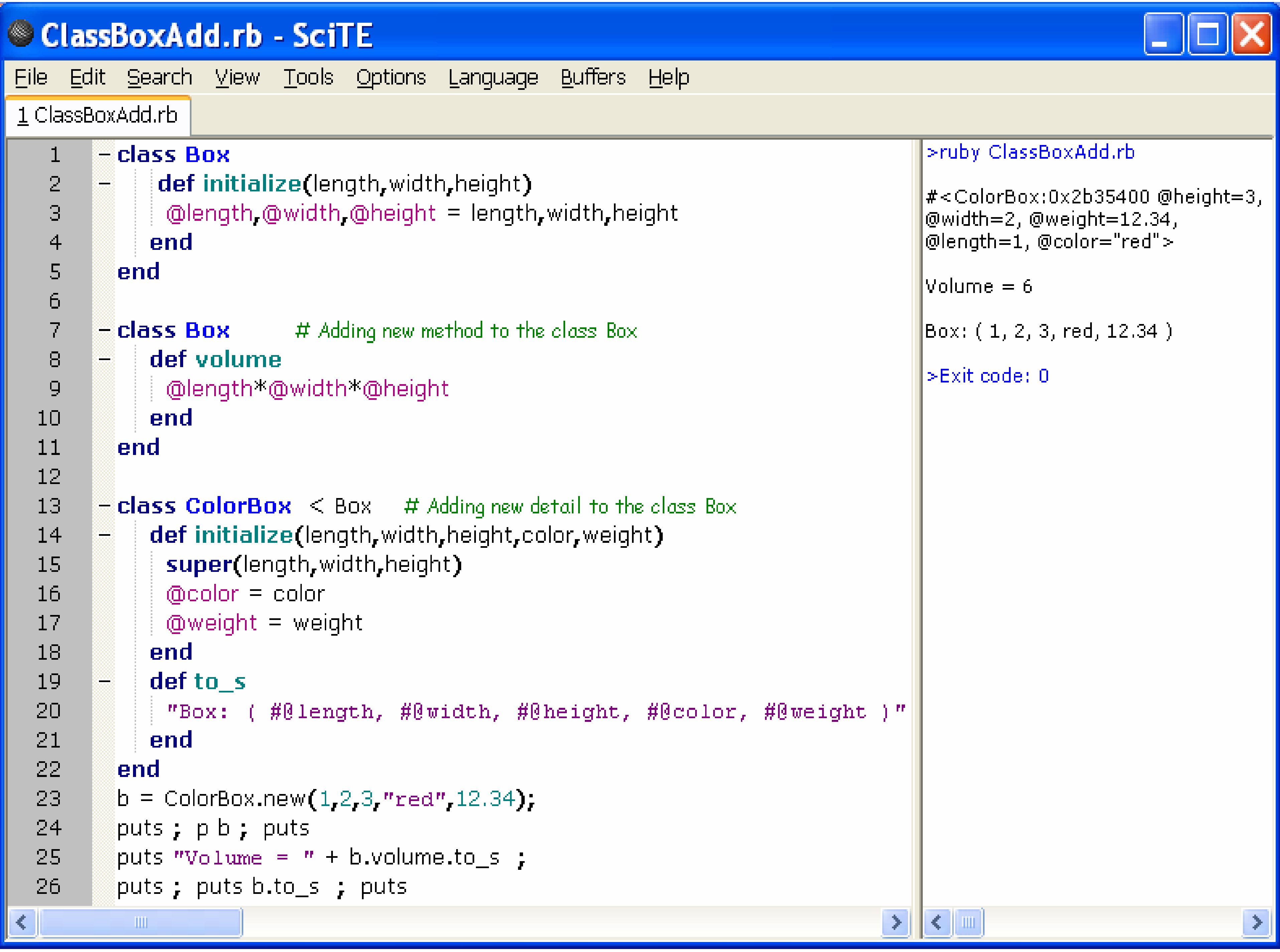
Inheritance

- Inheritance is a method to create a class that is a refinement or specialization of another class
- Syntax:

```
class <old class>
  .....
end
class <new class>  <  <old class>
  .....
end
```

Old class = superclass/parent class

New class = subclass/child class



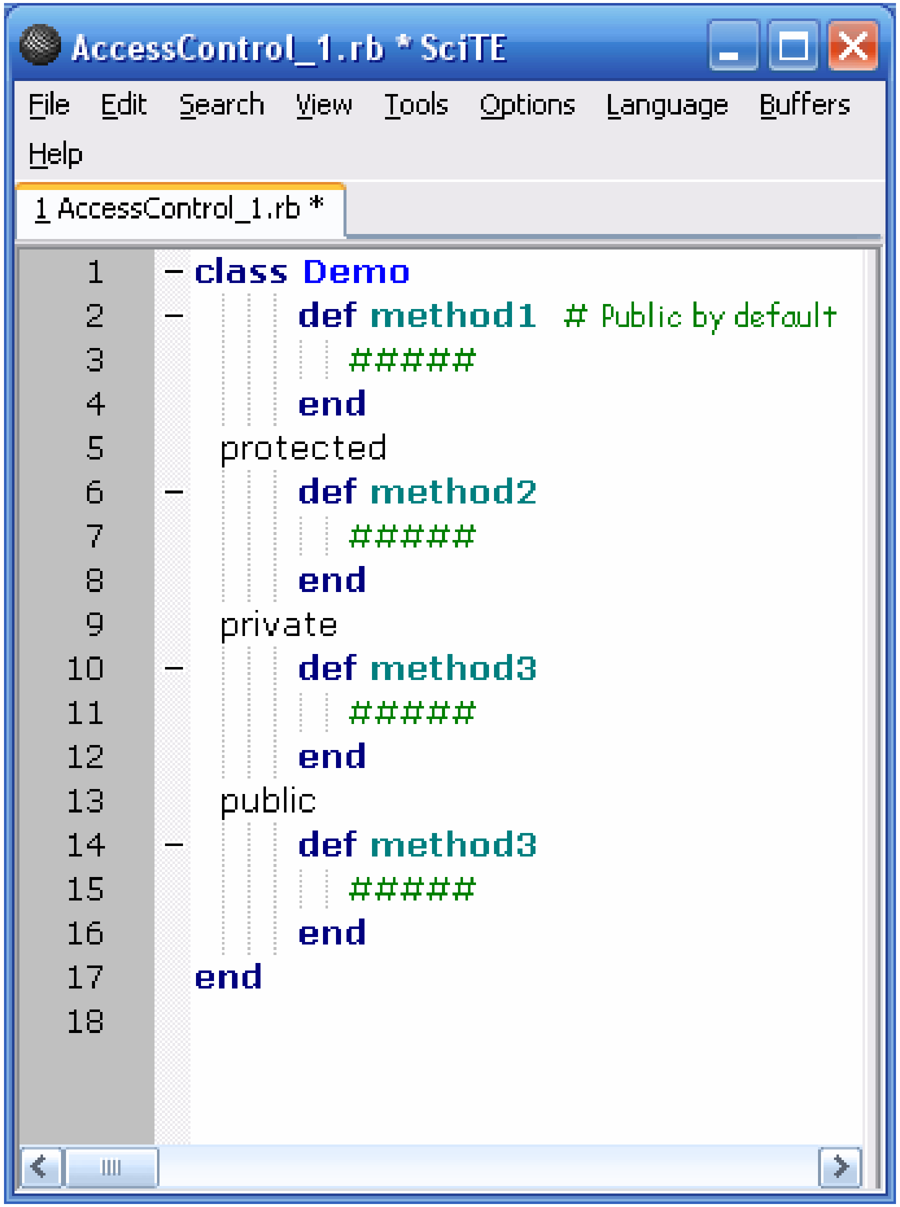
```ruby
class Box
  def initialize(length,width,height)
    @length,@width,@height = length,width,height
  end
end

class Box        # Adding new method to the class Box
  def volume
    @length*@width*@height
  end
end

class ColorBox  < Box    # Adding new detail to the class Box
  def initialize(length,width,height,color,weight)
    super(length,width,height)
    @color = color
    @weight = weight
  end
  def to_s
    "Box: ( #@length, #@width, #@height, #@color, #@weight )"
  end
end
b = ColorBox.new(1,2,3,"red",12.34);
puts ; p b ; puts
puts "Volume = " + b.volume.to_s  ;
puts ; puts b.to_s  ; puts
```

```
>ruby ClassBoxAdd.rb

#<ColorBox:0x2b35400 @height=3,
@width=2, @weight=12.34,
@length=1, @color="red">

Volume = 6

Box: ( 1, 2, 3, red, 12.34 )

>Exit code: 0
```

Access Control

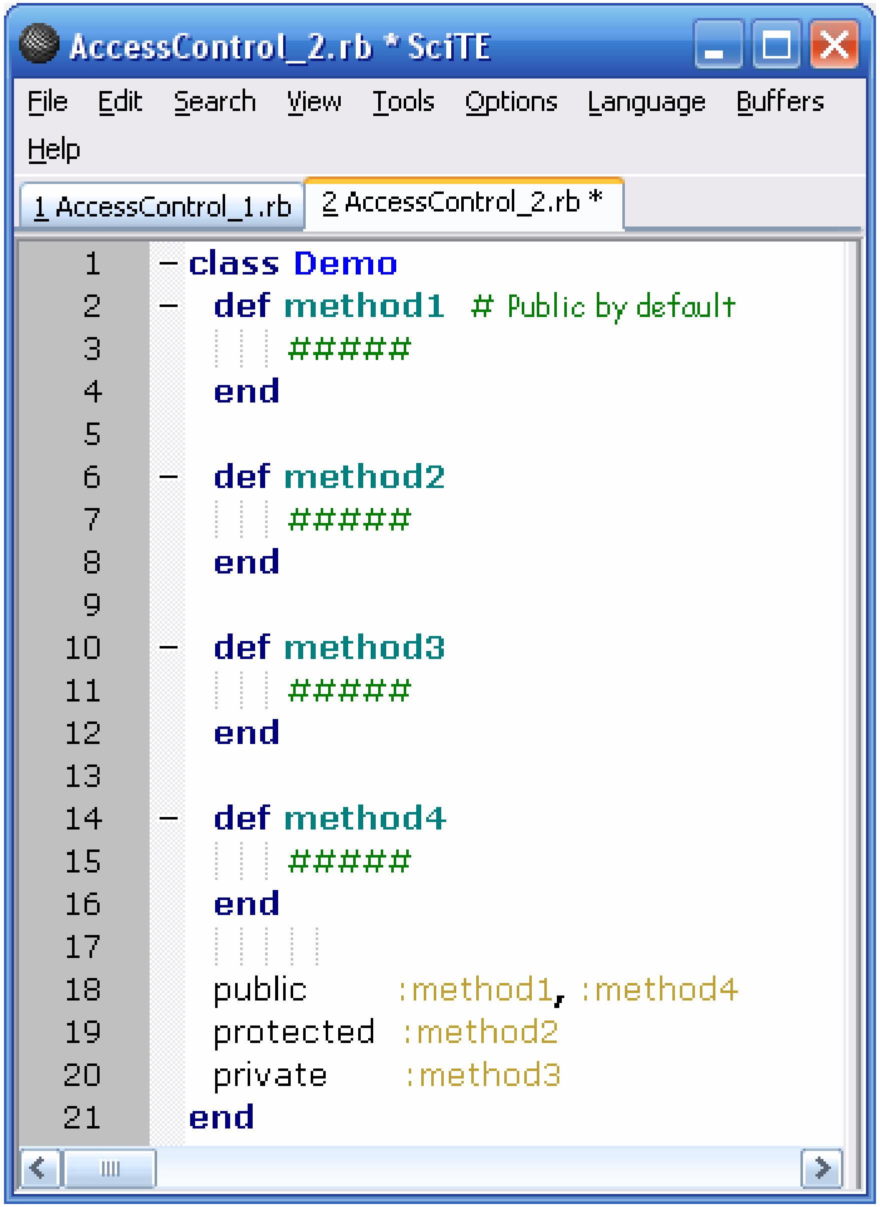
- **Public methods**: can be called by anyone (this is default, except for initialize, which is always private)
- **Protected methods**: access is restricted to family (accessed by the class and its subclasses/children)
- **Private methods**: cannot be called with receiver other than **self**

Equivalent notation



AccessControl_1.rb * SciTE
File Edit Search View Tools Options Language Buffers
Help
1 AccessControl_1.rb *

```ruby
class Demo
    def method1  # Public by default
      #####
    end
  protected
    def method2
      #####
    end
  private
    def method3
      #####
    end
  public
    def method3
      #####
    end
end
```



AccessControl_2.rb * SciTE
File Edit Search View Tools Options Language Buffers
Help
1 AccessControl_1.rb
2 AccessControl_2.rb *

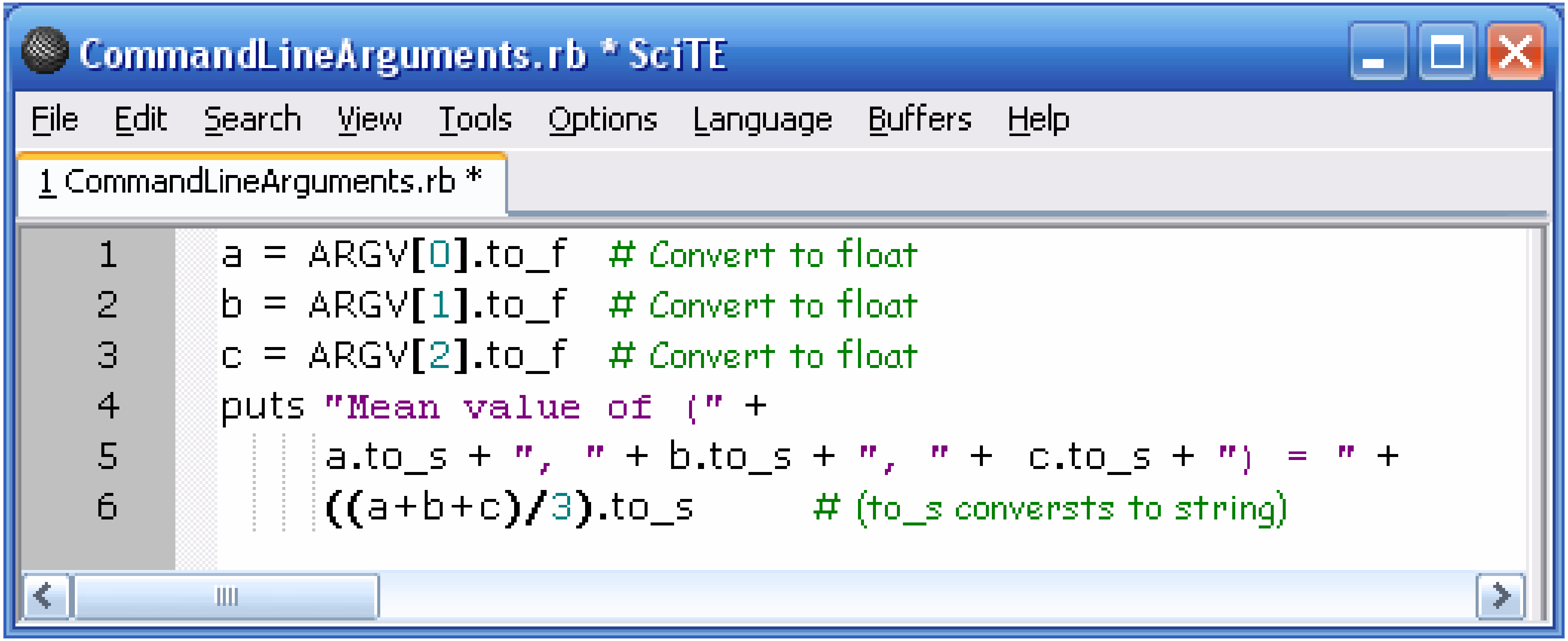
```ruby
class Demo
  def method1  # Public by default
    #####
  end

  def method2
    #####
  end

  def method3
    #####
  end

  def method4
    #####
  end

  public     :method1, :method4
  protected  :method2
  private    :method3
end
```

Elements of input/output

- Input/output comes from several sources:
  - Command line argument
  - Keyboard
    - Interactive mode
    - Interpretative mode
  - Files
- Keyboard input and command line input are processed in the command line environment

Using command line arguments

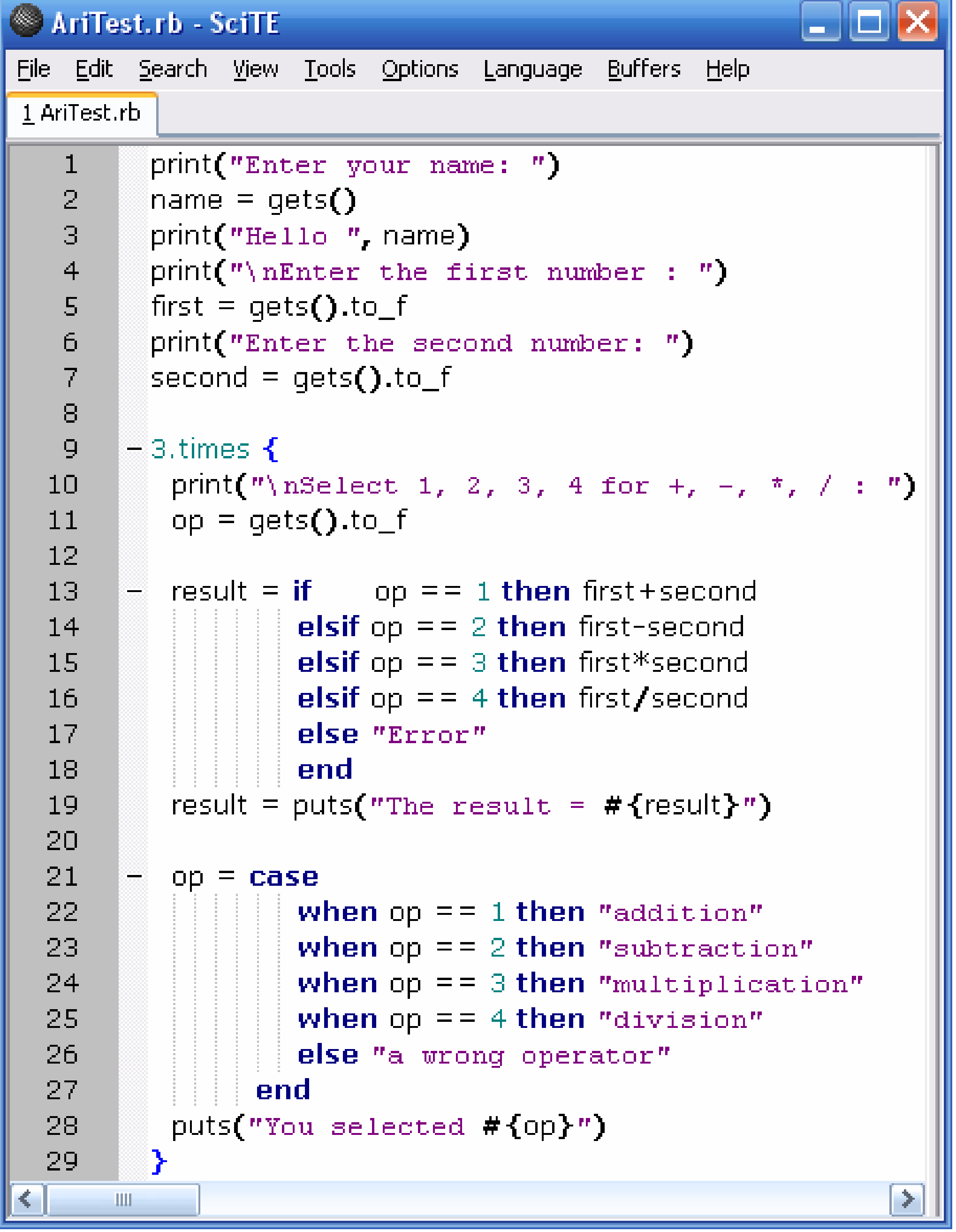
- Program file CommndLineArguments.rb:


CommandLineArguments.rb * SciTE
File Edit Search View Tools Options Language Buffers Help
1 CommandLineArguments.rb *

```
a = ARGV[0].to_f   # Convert to float
b = ARGV[1].to_f   # Convert to float
c = ARGV[2].to_f   # Convert to float
puts "Mean value of (" +
      a.to_s + ", " + b.to_s + ", " +  c.to_s + ") = " +
      ((a+b+c)/3).to_s       # (to_s conversts to string)
```

- Execution in the command prompt mode:

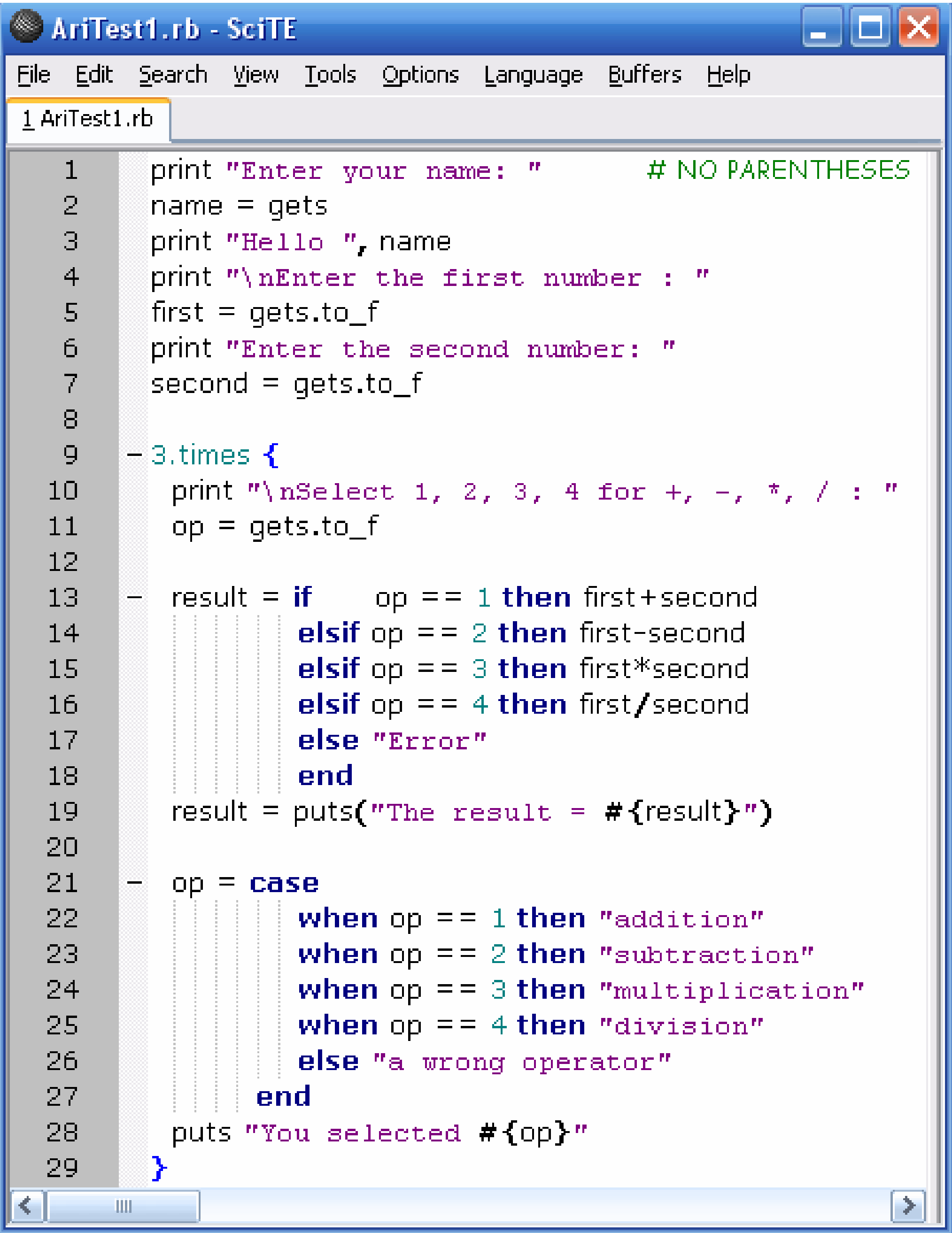
```
>ruby CommandLineArguments.rb  1  4  5
Mean value of (1.0, 4.0, 5.0) = 3.33333333333333
```



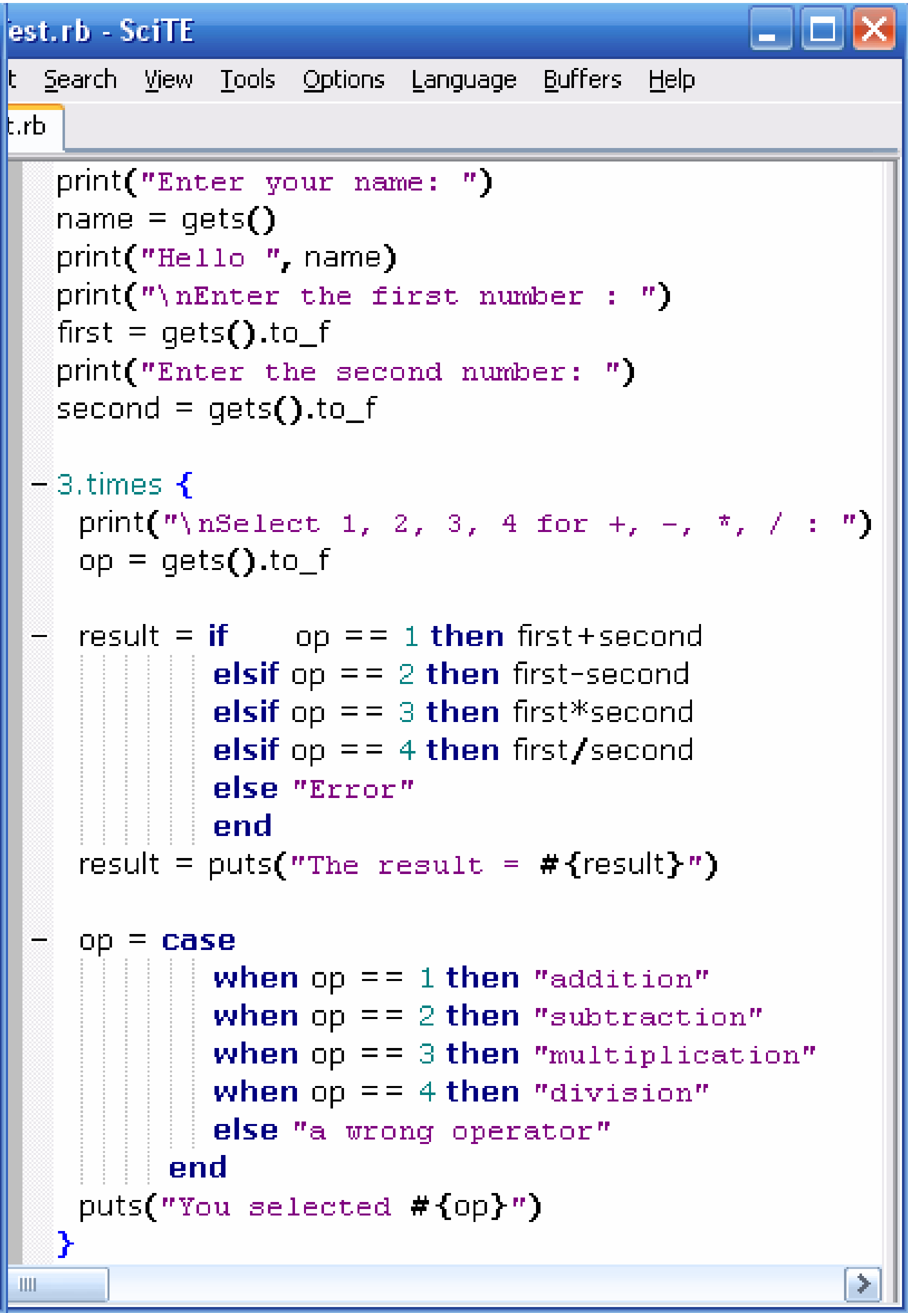
```ruby
print("Enter your name: ")
name = gets()
print("Hello ", name)
print("\nEnter the first number : ")
first = gets().to_f
print("Enter the second number: ")
second = gets().to_f

3.times {
  print("\nSelect 1, 2, 3, 4 for +, -, *, / : ")
  op = gets().to_f

  result = if    op == 1 then first+second
           elsif op == 2 then first-second
           elsif op == 3 then first*second
           elsif op == 4 then first/second
           else "Error"
           end
  result = puts("The result = #{result}")

  op = case
           when op == 1 then "addition"
           when op == 2 then "subtraction"
           when op == 3 then "multiplication"
           when op == 4 then "division"
           else "a wrong operator"
        end
  puts("You selected #{op}")
}
```

```
>ruby AriTest.rb
Enter your name: Ruby
Hello Ruby

Enter the first number : 1.1111
Enter the second number: 2.2222

Select 1, 2, 3, 4 for +, -, *, / : 4
The result = 0.5
You selected division

Select 1, 2, 3, 4 for +, -, *, / : 5
The result = Error
You selected a wrong operator

Select 1, 2, 3, 4 for +, -, *, / : 1
The result = 3.3333
You selected addition
```

Parentheses can be omitted. These two versions work in the same way



```
print "Enter your name: "          # NO PARENTHESES
name = gets
print "Hello ", name
print "\nEnter the first number : "
first = gets.to_f
print "Enter the second number: "
second = gets.to_f

3.times {
  print "\nSelect 1, 2, 3, 4 for +, -, *, / : "
  op = gets.to_f

  result = if     op == 1 then first+second
                  elsif op == 2 then first-second
                  elsif op == 3 then first*second
                  elsif op == 4 then first/second
                  else "Error"
                  end
  result = puts("The result = #{result}")

  op = case
                  when op == 1 then "addition"
                  when op == 2 then "subtraction"
                  when op == 3 then "multiplication"
                  when op == 4 then "division"
                  else "a wrong operator"
              end
  puts "You selected #{op}"
}
```



```
print("Enter your name: ")
name = gets()
print("Hello ", name)
print("\nEnter the first number : ")
first = gets().to_f
print("Enter the second number: ")
second = gets().to_f

3.times {
  print("\nSelect 1, 2, 3, 4 for +, -, *, / : ")
  op = gets().to_f

  result = if     op == 1 then first+second
                  elsif op == 2 then first-second
                  elsif op == 3 then first*second
                  elsif op == 4 then first/second
                  else "Error"
                  end
  result = puts("The result = #{result}")

  op = case
                  when op == 1 then "addition"
                  when op == 2 then "subtraction"
                  when op == 3 then "multiplication"
                  when op == 4 then "division"
                  else "a wrong operator"
              end
  puts("You selected #{op}")
}
```

Files

- Dir = class for manipulating directories

```
Dir.chdir(“/Users/bill/ruby”)
home = Dir.pwd  # Users/peter/ruby
Dir.mkdir(“/Users/bill/scheme”)
Dir.rmdir(“/Users/bill/pascal”)
```

- File = class to manipulate files
  - Create
  - Open
  - Rename
  - Delete

Create file

- Create and open file for writing

  file = File.new(“data.txt”, “w”)

- Modes:

  “r” = read only, start from beginning

  “r+” = read-write, start from beginning

  “w” = write only (create new or overwrite an existing file)

  “w+” = write-read (create new or overwrite existing file)

  “a” = append (write only) or create new file, start at EOF

  “a+” = append (read-write) or create new file, start at EOF

  “b” = binary file mode (DOS/Windows only)

Open/close file

```
file = File.open(“data.txt”)
file.each{ |line| print “#{file.lineno}“, line }
file.close

File.open(“data.txt”) if File::exists?(“data.txt”)
```

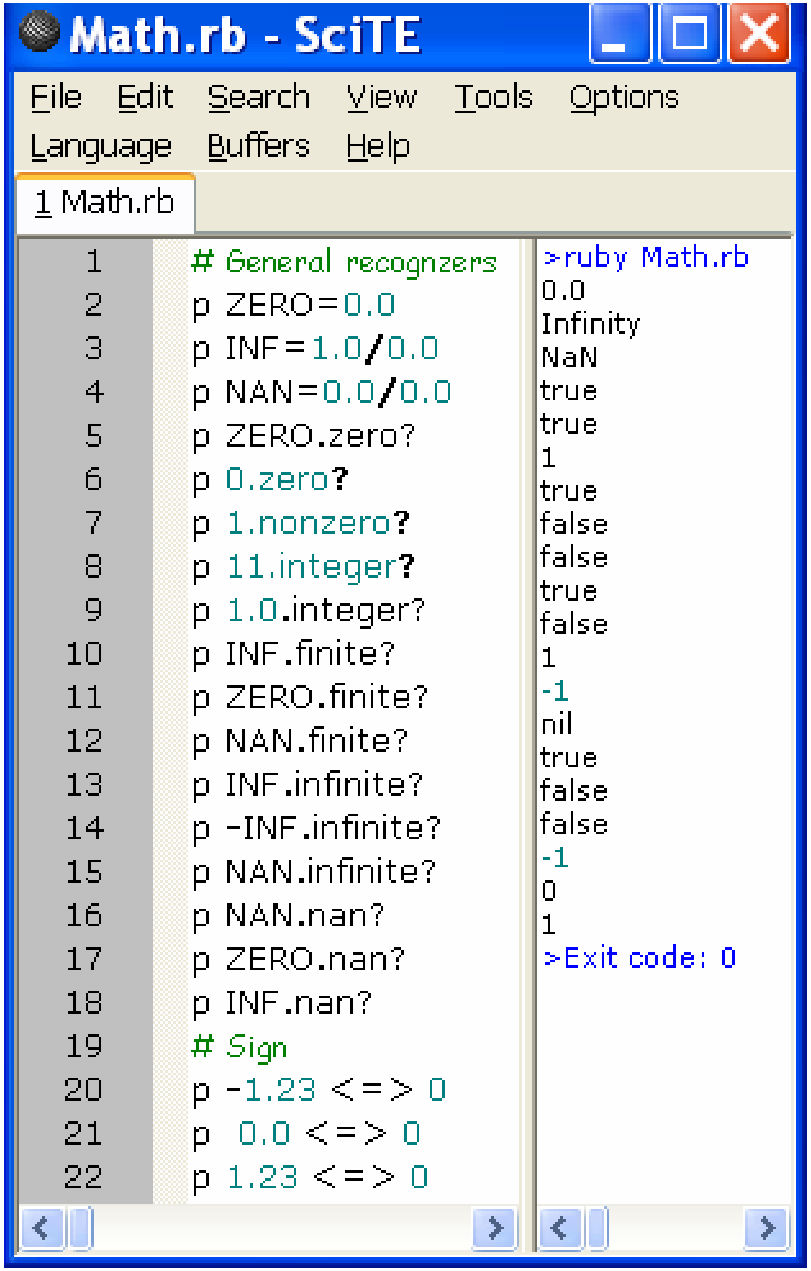
Rename, Delete, Test

```
file = File.new(“data.txt”, “w”)
File.rename(“data.txt”, “junk.txt”)
File.delete(“junk.txt”)

file.closed?  #  returns true/false
file.size     #  returns size in bytes
```

Basic numeric methods (Math)

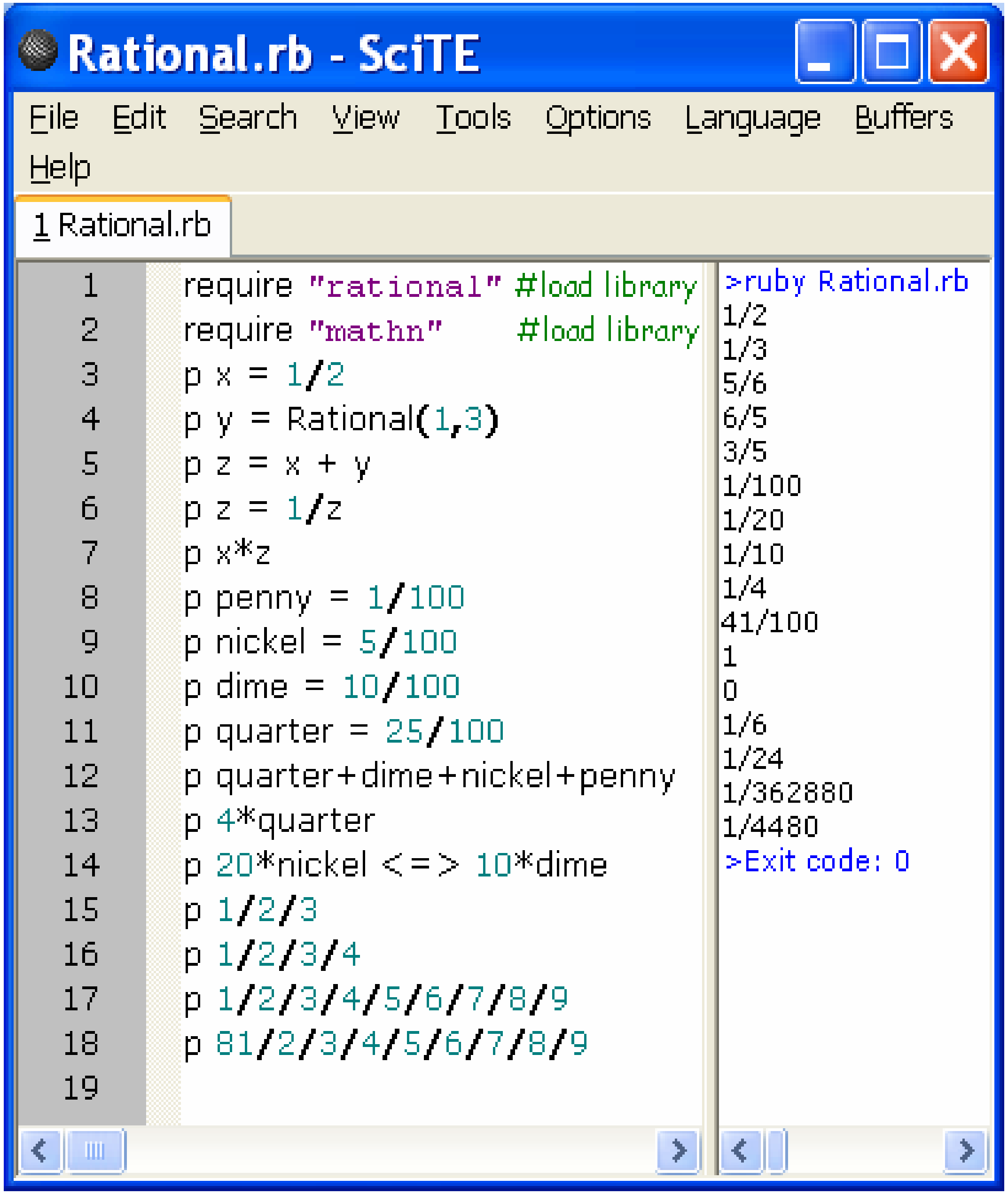
- Recognizers
- Rounding
- Absolute value
- Sign
- Constants
- Not a number
- Infinity
- Roots
- Logarithms/exponentials
- Cartesian/polar conversion
- Dates and time measure

- Trigonometry
- Hyperbolic functions
- Fraction/exponent decomposition
- Error function
- BigDecimal arithmetic
- Complex numbers and functions
- Rational numbers and rational arithmetic oper.
- Random numbers (seed and computation)
- Vectors and matrices



```
# General recognzers
p ZERO=0.0
p INF=1.0/0.0
p NAN=0.0/0.0
p ZERO.zero?
p 0.zero?
p 1.nonzero?
p 11.integer?
p 1.0.integer?
p INF.finite?
p ZERO.finite?
p NAN.finite?
p INF.infinite?
p -INF.infinite?
p NAN.infinite?
p NAN.nan?
p ZERO.nan?
p INF.nan?
# Sign
p -1.23 <=> 0
p  0.0 <=> 0
p 1.23 <=> 0
```

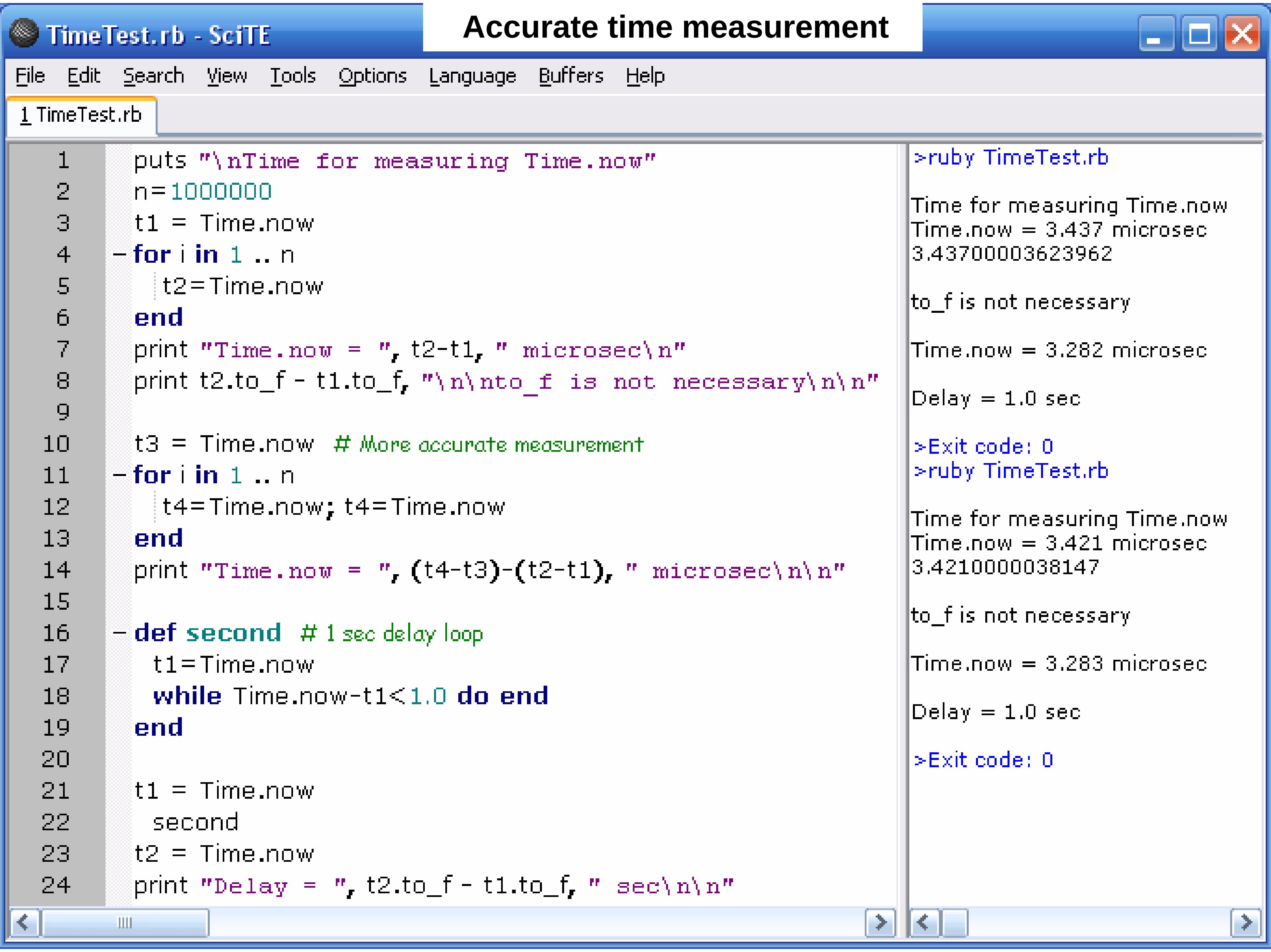
```
>ruby Math.rb
0.0
Infinity
NaN
true
true
1
true
false
false
true
false
1
-1
nil
true
false
false
-1
0
1
>Exit code: 0
```



```
require "rational" #load library
require "mathn"     #load library
p x = 1/2
p y = Rational(1,3)
p z = x + y
p z = 1/z
p x*z
p penny  = 1/100
p nickel  = 5/100
p dime  = 10/100
p quarter  = 25/100
p quarter+dime+nickel+penny
p 4*quarter
p 20*nickel <=> 10*dime
p 1/2/3
p 1/2/3/4
p 1/2/3/4/5/6/7/8/9
p 81/2/3/4/5/6/7/8/9
```

```
>ruby Rational.rb
1/2
1/3
5/6
6/5
3/5
1/100
1/20
1/10
1/4
41/100
1
0
1/6
1/24
1/362880
1/4480
>Exit code: 0
```

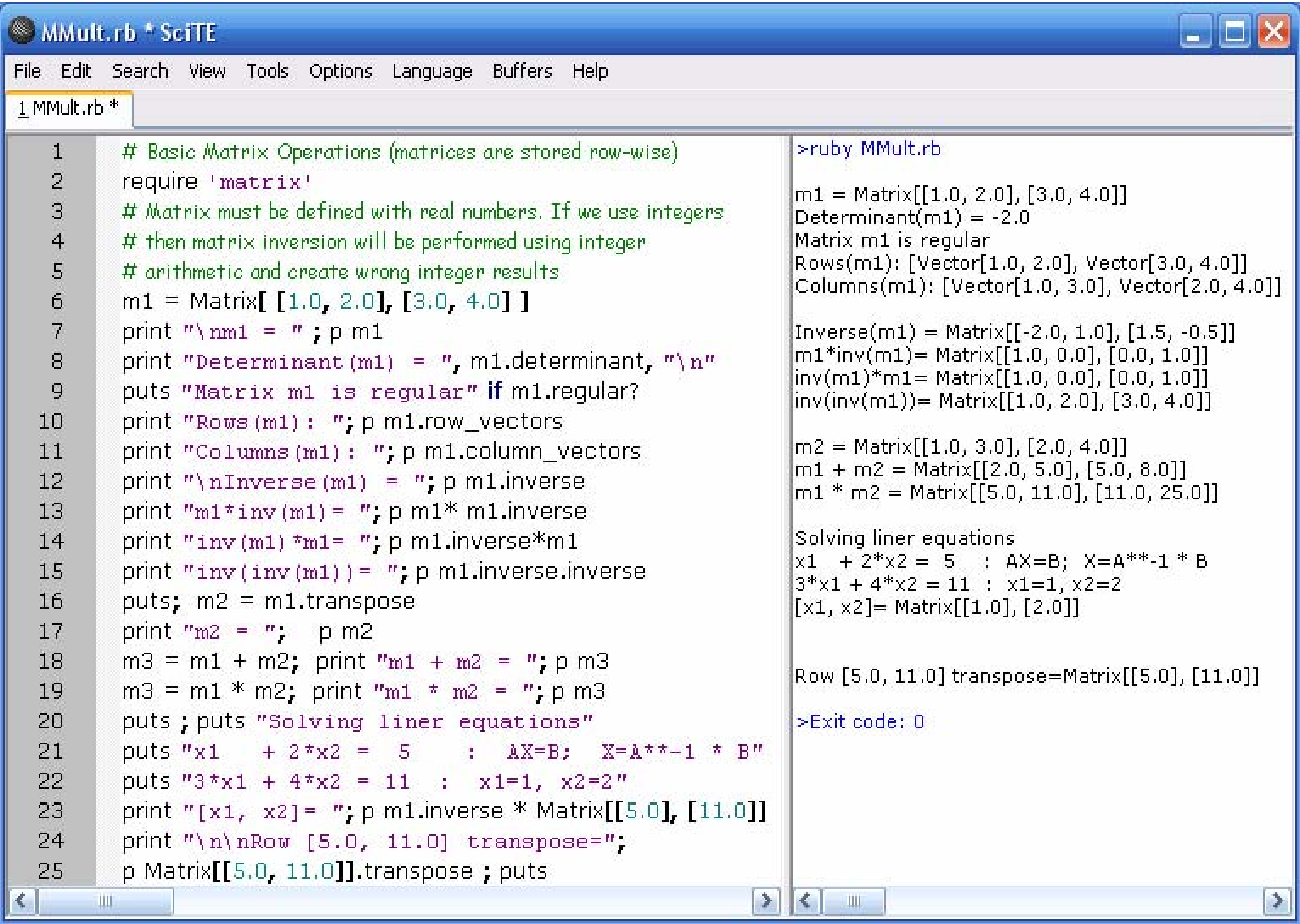
Accurate time measurement



```ruby
puts "\nTime for measuring Time.now"
n=1000000
t1 = Time.now
for i in 1 .. n
  t2=Time.now
end
print "Time.now = ", t2-t1, " microsec\n"
print t2.to_f - t1.to_f, "\n\nto_f is not necessary\n\n"

t3 = Time.now  # More accurate measurement
for i in 1 .. n
  t4=Time.now; t4=Time.now
end
print "Time.now = ", (t4-t3)-(t2-t1), " microsec\n\n"

def second  # 1 sec delay loop
  t1=Time.now
  while Time.now-t1<1.0 do end
end

t1 = Time.now
  second
t2 = Time.now
print "Delay = ", t2.to_f - t1.to_f, " sec\n\n"
```

```
>ruby TimeTest.rb

Time for measuring Time.now
Time.now = 3.437 microsec
3.43700003623962

to_f is not necessary

Time.now = 3.282 microsec

Delay = 1.0 sec

>Exit code: 0
>ruby TimeTest.rb

Time for measuring Time.now
Time.now = 3.421 microsec
3.4210000038147

to_f is not necessary

Time.now = 3.283 microsec

Delay = 1.0 sec

>Exit code: 0
```

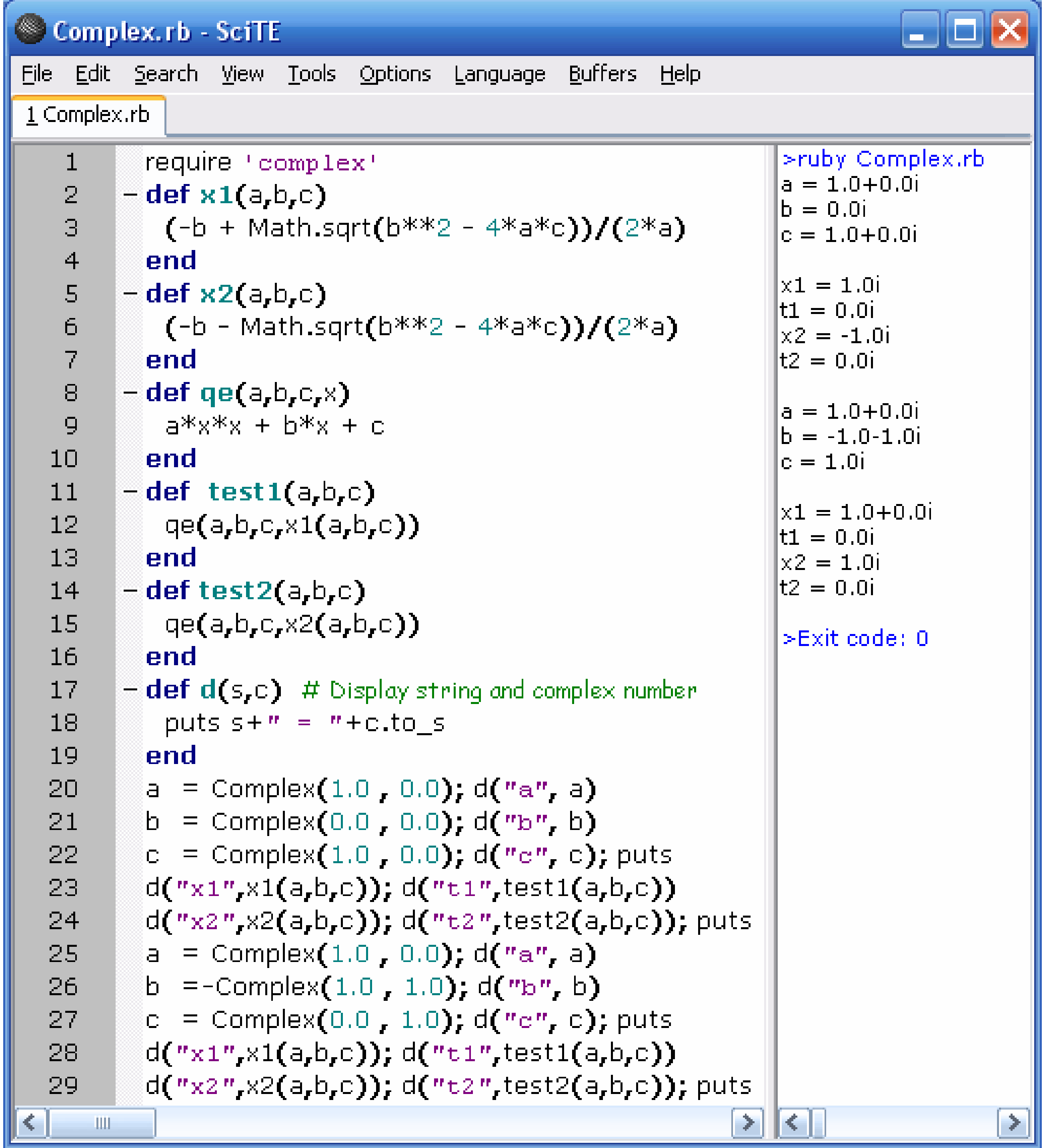
Matrix operations



```ruby
# Basic Matrix Operations (matrices are stored row-wise)
require 'matrix'
# Matrix must be defined with real numbers. If we use integers
# then matrix inversion will be performed using integer
# arithmetic and create wrong integer results
m1 = Matrix[ [1.0, 2.0], [3.0, 4.0] ]
print "\nm1 = " ; p m1
print "Determinant(m1) = ", m1.determinant, "\n"
puts "Matrix m1 is regular" if m1.regular?
print "Rows(m1): "; p m1.row_vectors
print "Columns(m1): "; p m1.column_vectors
print "\nInverse(m1) = "; p m1.inverse
print "m1*inv(m1)= "; p m1* m1.inverse
print "inv(m1)*m1= "; p m1.inverse*m1
print "inv(inv(m1))= "; p m1.inverse.inverse
puts; m2 = m1.transpose
print "m2 = ";   p m2
m3 = m1 + m2; print "m1 + m2 = "; p m3
m3 = m1 * m2; print "m1 * m2 = "; p m3
puts ; puts "Solving liner equations"
puts "x1    + 2*x2 =  5    :  AX=B;  X=A**-1 * B"
puts "3*x1 + 4*x2 = 11   :  x1=1, x2=2"
print "[x1, x2]= "; p m1.inverse * Matrix[[5.0], [11.0]]
print "\n\nRow [5.0, 11.0] transpose=";
p Matrix[[5.0, 11.0]].transpose ; puts
```

```
>ruby MMult.rb

m1 = Matrix[[1.0, 2.0], [3.0, 4.0]]
Determinant(m1) = -2.0
Matrix m1 is regular
Rows(m1): [Vector[1.0, 2.0], Vector[3.0, 4.0]]
Columns(m1): [Vector[1.0, 3.0], Vector[2.0, 4.0]]

Inverse(m1) = Matrix[[-2.0, 1.0], [1.5, -0.5]]
m1*inv(m1)= Matrix[[1.0, 0.0], [0.0, 1.0]]
inv(m1)*m1= Matrix[[1.0, 0.0], [0.0, 1.0]]
inv(inv(m1))= Matrix[[1.0, 2.0], [3.0, 4.0]]

m2 = Matrix[[1.0, 3.0], [2.0, 4.0]]
m1 + m2 = Matrix[[2.0, 5.0], [5.0, 8.0]]
m1 * m2 = Matrix[[5.0, 11.0], [11.0, 25.0]]

Solving liner equations
x1   + 2*x2 =  5   :  AX=B;  X=A**-1 * B
3*x1 + 4*x2 = 11  :  x1=1, x2=2
[x1, x2]= Matrix[[1.0], [2.0]]


Row [5.0, 11.0] transpose=Matrix[[5.0], [11.0]]

>Exit code: 0
```

Solving a quadratic equation with complex coefficients



```ruby
require 'complex'
def x1(a,b,c)
  (-b + Math.sqrt(b**2 - 4*a*c))/(2*a)
end
def x2(a,b,c)
  (-b - Math.sqrt(b**2 - 4*a*c))/(2*a)
end
def qe(a,b,c,x)
  a*x*x + b*x + c
end
def  test1(a,b,c)
  qe(a,b,c,x1(a,b,c))
end
def test2(a,b,c)
  qe(a,b,c,x2(a,b,c))
end
def d(s,c)  # Display string and complex number
  puts s+" = "+c.to_s
end
a  = Complex(1.0 , 0.0); d("a", a)
b  = Complex(0.0 , 0.0); d("b", b)
c  = Complex(1.0 , 0.0); d("c", c); puts
d("x1",x1(a,b,c)); d("t1",test1(a,b,c))
d("x2",x2(a,b,c)); d("t2",test2(a,b,c)); puts
a  = Complex(1.0 , 0.0); d("a", a)
b  =-Complex(1.0 , 1.0); d("b", b)
c  = Complex(0.0 , 1.0); d("c", c); puts
d("x1",x1(a,b,c)); d("t1",test1(a,b,c))
d("x2",x2(a,b,c)); d("t2",test2(a,b,c)); puts
```

```
>ruby Complex.rb
a = 1.0+0.0i
b = 0.0i
c = 1.0+0.0i

x1 = 1.0i
t1 = 0.0i
x2 = -1.0i
t2 = 0.0i

a = 1.0+0.0i
b = -1.0-1.0i
c = 1.0i

x1 = 1.0+0.0i
t1 = 0.0i
x2 = 1.0i
t2 = 0.0i

>Exit code: 0
```

Highlights and Conclusions

- Ruby is a multi-paradigm language: it is good for procedural, functional and object oriented programming
- Comfortable: Ruby is a dynamically typed language – no type definitions before we start running code
- Rich: 98 standard libraries and 9000+ methods
- Relaxed approach to syntax – many alternative ways to achieve goals; no rigid rules
- Strictly object oriented – that yields rich control structures and flexibility in building objects
- Simplicity in defining and expanding classes
- Free and growing: web development
- Performance awareness: benchmarks, YARV
- Tools: debugger, profiler and much more

Next steps

- Ruby on rails – web development framework: http://www.rubyonrails.org
- Common Gateway Interface (CGI) web programming toolkit http://www.spice-of-life.net/cgikit/index_en.html