introduction to jruby

NEAL FORD software architect / meme wrangler

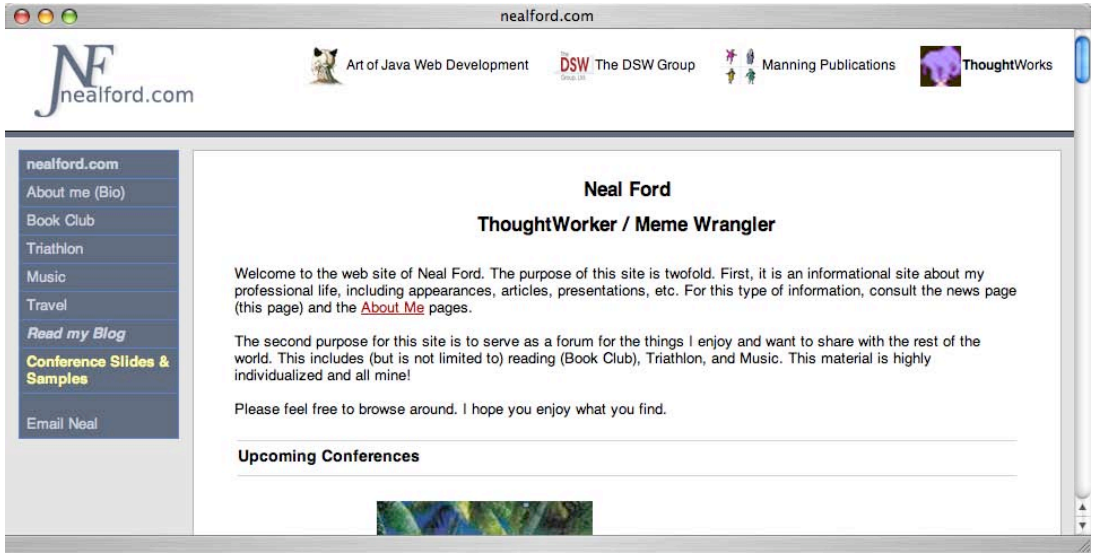
ThoughtWorks

nford@thoughtworks.com
3003 Summit Boulevard, Atlanta, GA 30319
www.nealford.com
www.thoughtworks.com
memeagora.blogspot.com

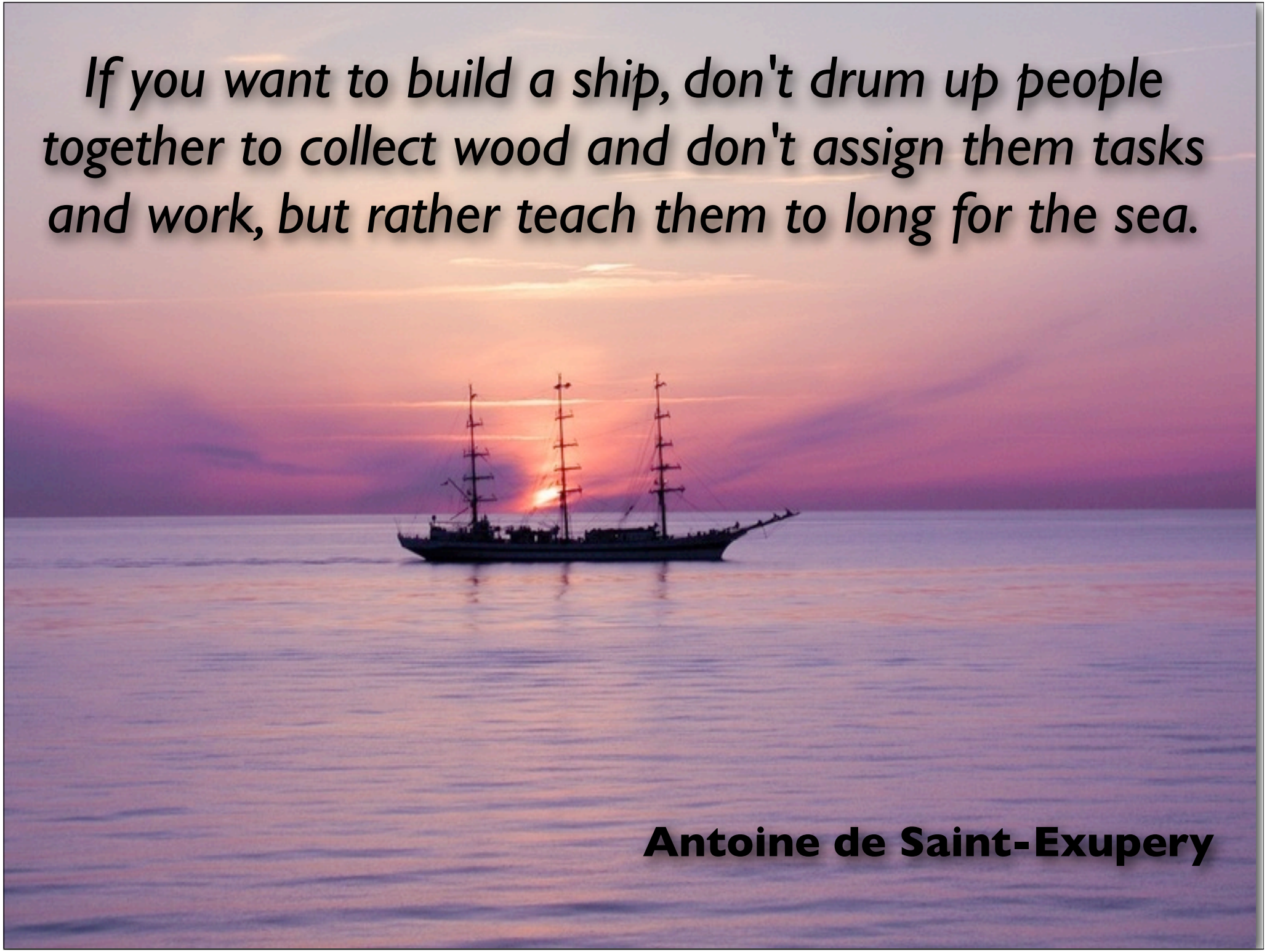
housekeeping

ask questions anytime

download slides from nealford.com



download samples from `github.com/nealford`


If you want to build a ship, don't drum up people together to collect wood and don't assign them tasks and work, but rather teach them to long for the sea.
Antoine de Saint-Exupery

why ruby?

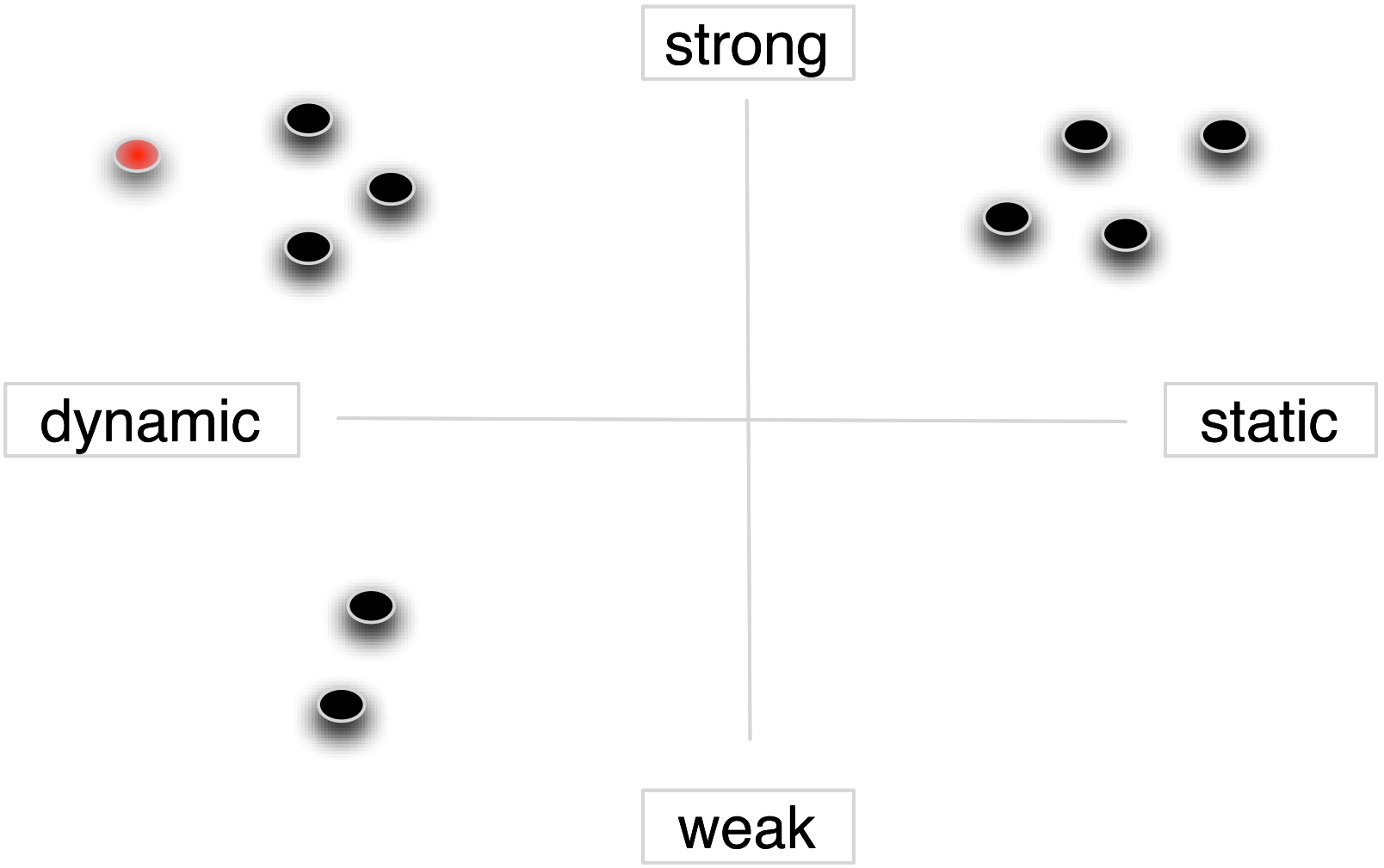
purely object-oriented

compact syntax

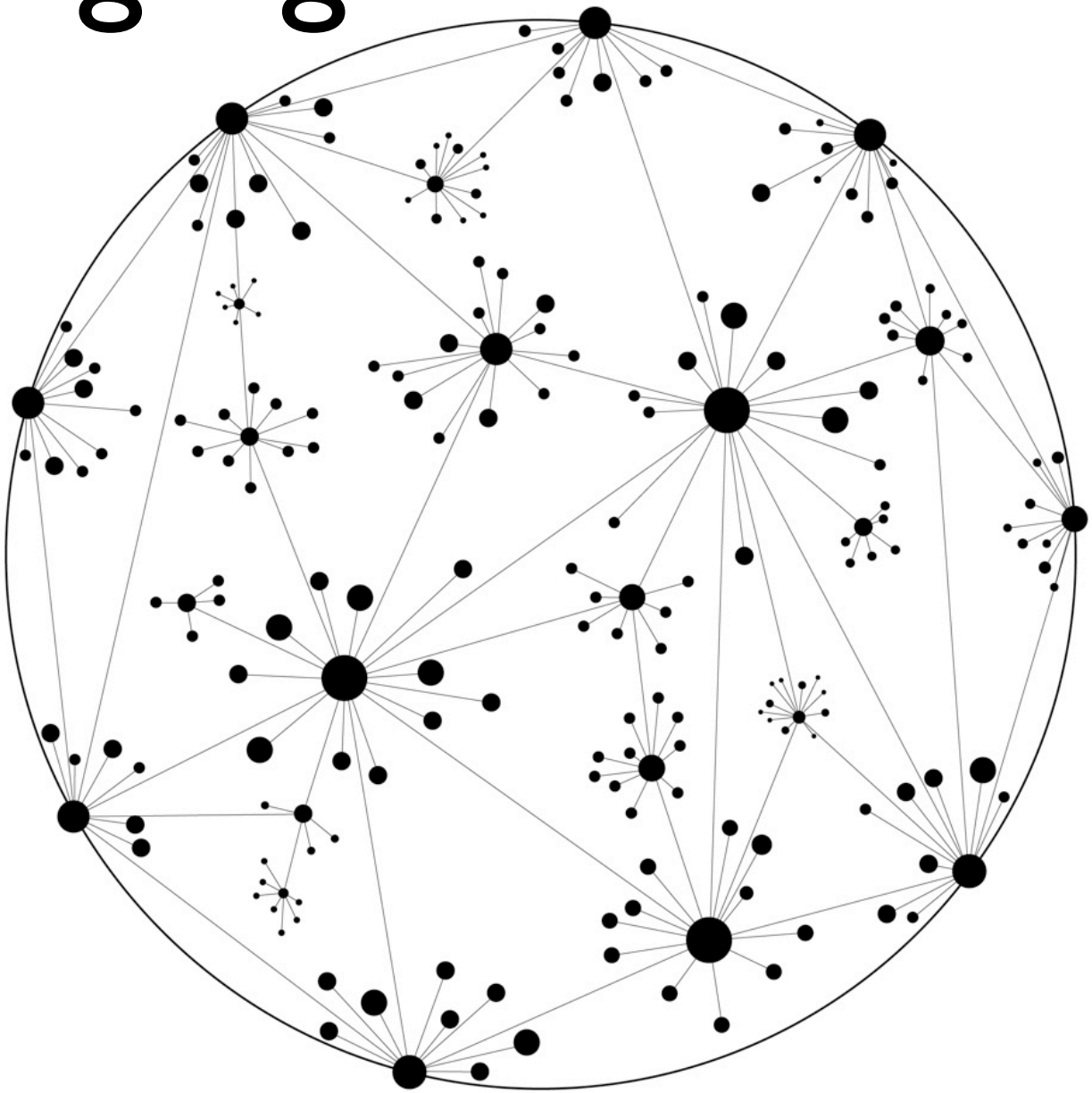
advanced language features

rails!

dynamically typed



language surface area



what is jruby?

sophisticated port of ruby to the java platform

mri: matz reference implementation

written in c & ruby

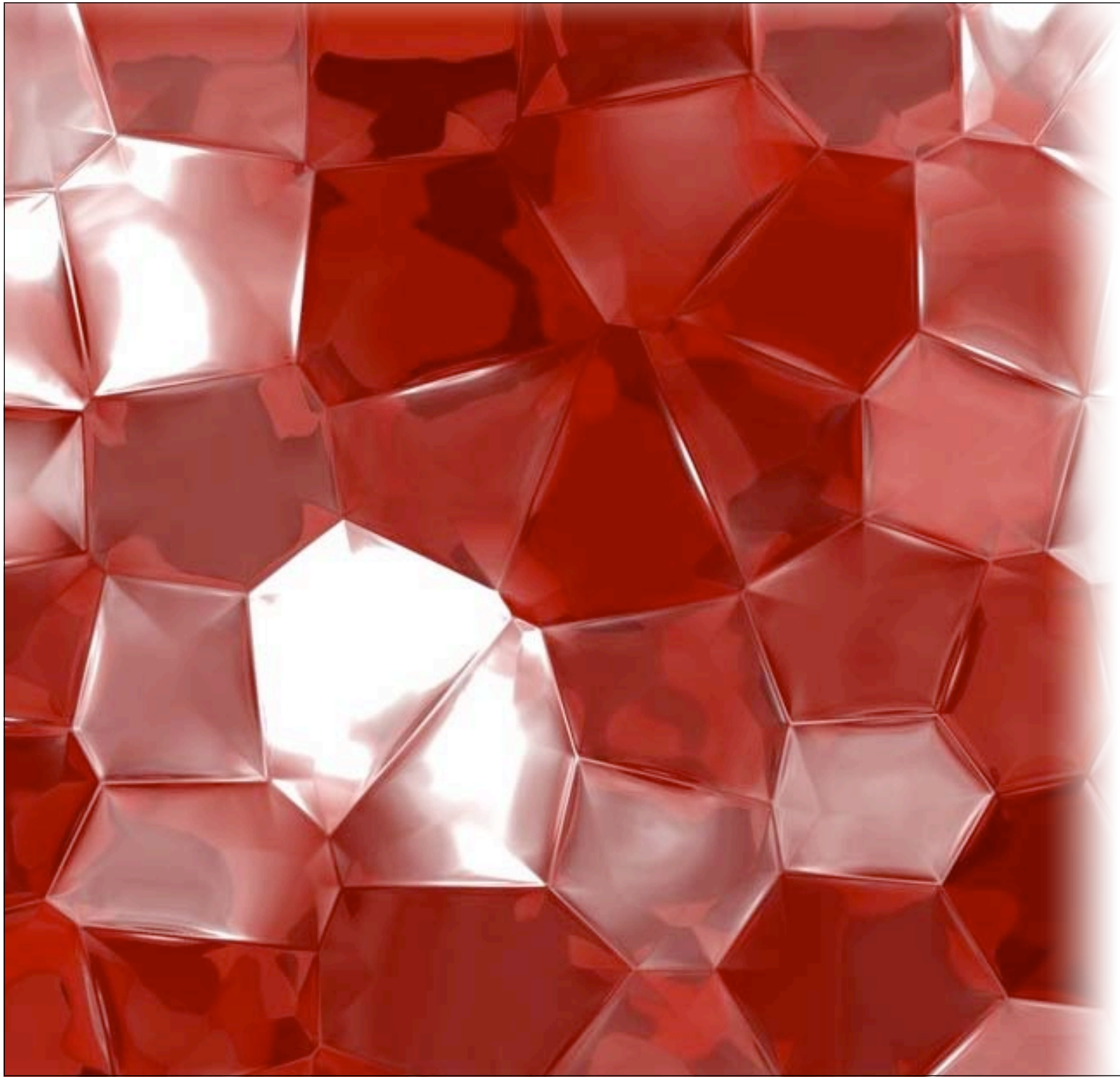
ported to all major (os) platforms

what is jruby?

jruby 1.0 ported interpreter to java

jruby 1.1 created a compiler

jruby is now the fastest version of ruby



ruby facets


```java
public class Employee {
    private String _name;
    private double _salary;
    private int _hireYear;

    public Employee(String name, double salary, int hireYear) {
        _name = name;
        _salary = salary;
        _hireYear = hireYear;
    }

    public String getName() {
        return _name;
    }

    public Double getSalary() {
        return _salary;
    }

    public int getHireYear() {
        return _hireYear;
    }

    public String toString() {
        return "Name is " + _name + ", salary is " + _salary +
                ", hire year is " + _hireYear;
    }

    public String toS() {
        return toString();
    }

    public void raiseSalary(int percentage) {
        _salary += (_salary * (percentage * 0.01));
    }
}
```

```ruby
class Employee
  def initialize(name, salary, hire_year=2007)
    @name = name
    @salary = salary
    @hire_year = hire_year
  end

  attr_reader :name, :salary, :hire_year

  def to_s
    "Name: #{@name}, salary: #{@salary}, " +
    "hire year: #{@hire_year}"
  end
  alias_method :to_string, :to_s

  def raise_salary_by(perc)
    @salary += (@salary * (perc * 0.01))
  end
end
```

```
public class Manager extends Employee{
    private Employee _assistant;

    public Manager(String name, double salary, int hireYear, Employee assistant) {
        super(name, salary, hireYear);
        _assistant = assistant;
    }

    public Employee getAssistant() {
        return _assistant;
    }

    public String toString() {
        return super.toString() + ", assistant is " + _assistant.toString();
    }

    public String toS() {
        return toString();
    }

    public void raiseSalary(int percentage) {
        percentage += 2005 - getHireYear();
        super.raiseSalary(percentage);
    }
}
```

```ruby
class Manager < Employee
  def initialize(name, salary, hire_year, asst)
    super(name, salary, hire_year)
    @asst = asst
  end

  def to_s
    super + ",\tAssistant info: #{@asst}"
  end

  def raise_salary_by(perc)
    perc += 2005 - @hire_year
    super(perc)
  end
end
```

```
public class HrRunner {

    public static void main(String[] args) {
        new HrRunner();
    }

    public void show(List<Employee> emps) {
        for (Employee e : emps)
            System.out.println(e);
    }

    public HrRunner() {
        List<Employee> employees = new ArrayList<Employee>();
        employees.add(new Employee("Homer", 200.0, 1995));
        employees.add(new Employee("Lenny", 150.0, 2000));
        employees.add(new Employee("Carl", 250.0, 1999));
        employees.add(new Manager("Monty", 3000.0, 1950, employees.get(2)));

        show(employees);
        for (Employee e : employees)
            e.raiseSalary(10);

        show(employees);
    }
}
```

```ruby
require 'hr'

def show(emps)
  emps.each { |emp| puts emp  }
end

employees = Array.new
employees[0] = Employee.new("Homer", 200.0, 1995)
employees[1] = Employee.new("Lenny", 150.0, 2000)
employees[2] = Employee.new("Carl", 250.0, 1999)
employees[3] = Manager.new("Monty", 3000.0, 1950, employees[2])


show(employees)

employees.each { |e| e.raise_salary_by(10) }
puts "\nGive everyone a raise\n\n"

show employees
```

blocks

delimited with either { } or do..end

both support parameters

closures

```java
public class EmployeeList {
    private List<Employee> _employees;

    public List<Employee> getEmployees() {
        return _employees;
    }

    public EmployeeList() {
        _employees = new ArrayList<Employee>();
    }

    public void add(Employee e) {
        _employees.add(e);
    }

    public void deleteFirst() {
        _employees.remove(0);
    }

    public void deleteLast() {
        _employees.remove(_employees.size() - 1);
    }

    public void show() {
        for (Employee e : _employees)
            System.out.println(e.toString());
    }
}
```

list access

```
public Employee get(int key) {
    return _employees.get(key);
}

public Employee get(String name) {
    for (Employee e : _employees)
        if (e.getName().equals(name))
            return e;
    return null;
}
```

access tests

```
@Test public void get_with_int() {
    assertThat(_list.get(0),
            sameInstance(_list.getEmployees().get(0)));
}

@Test public void get_with_string() {
    assertThat(_list.get("Second"),
            sameInstance(_list.getEmployees().get(1)));
    assertNull(_list.get("Homer"));
}
```

```ruby
class EmployeeList
  def initialize
    @employees = Array.new
  end

  attr_reader :employees

  def add(an_employee)
    @employees.push(an_employee)
  end
  alias_method :<<, :add

  def delete_first
    @employees.shift
  end

  def delete_last
    @employees.pop
  end

  def show
    @employees.each { |e| puts e }
  end
```

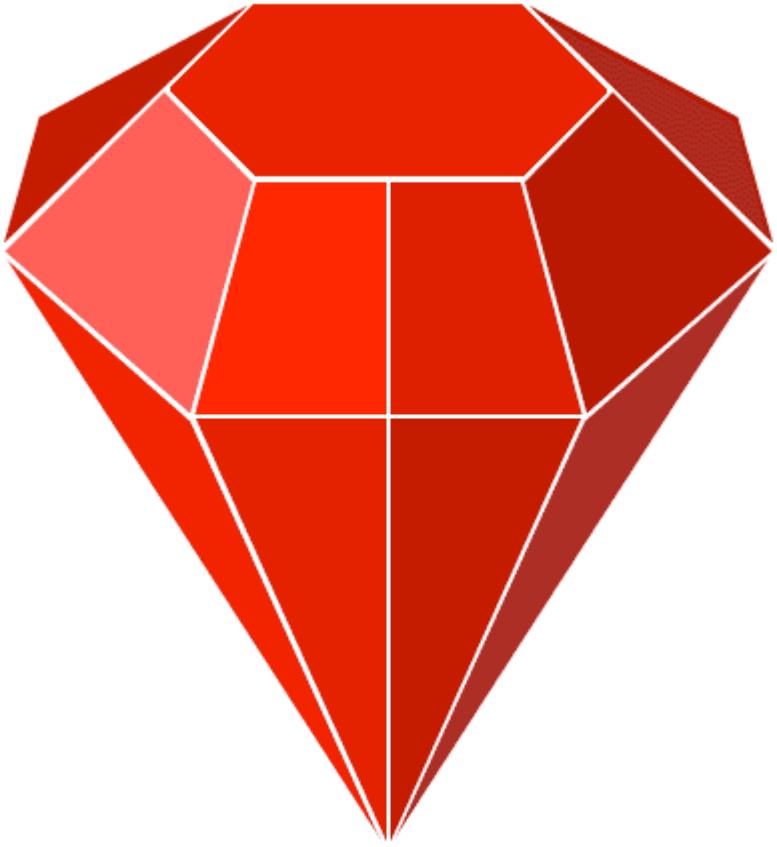
ruby and []

```ruby
def [](key)
  return @employees[key] if key.kind_of? Integer
  return @employees.find {|anEmp| key == anEmp.name }
end
```

ruby tests

```ruby
def test_int_braces
  for i in 0..@list.employees.size - 1 do
    assert @list[i] == @list.employees[i]
  end
end

def test_string_braces
  %w(First Second Third).each_with_index do |i, n|
    assert @list[n] == @list[i]
  end
  assert @list['foo'] == nil
  assert @list['foo'].nil?
end
```


JAVA™



java classes in jruby

```ruby
require 'java'

frame = javax.swing.JFrame.new("My Title")

JFrame = javax.swing.JFrame
frame = JFrame.new("My Title")

include_class "javax.swing.JFrame"
frame = JFrame.new("My Title")

include_class("java.lang.String") do |pkg_name, class_name|
  "J#{class_name}"
end
msg = JString.new("My Message")
```

substitute names programmatically

calling semantics

call methods like ruby methods

get/set/is methods invoked ala ruby

```
emp.getName()
```

```
emp.name
```

```
emp.setName("Homer")
```

```
emp.name = "Homer"
```

```
emp.isManager()
```

```
emp.manager?
```

calling semantics

camelcase java names may be called with underscores

```
require "java"
url = java.net.URL.new("http://www.nealford.com")
puts url.to_external_form     toExternalForm()
puts url.to_uri               toURI()
```

closures

a function evaluated in an environment
containing one or more bound variables

can be passed as data

instances of Proc

closure

```
def paid_more(amount)
  lambda { |e| e.salary > amount }
end
is_high_paid = paid_more(60000)

is_high_paid.call(employees[0])
```

the big deal

```ruby
def make_counter
  var = 0
  proc { var += 1 }
end

c1 = make_counter
c1.call                    # => 1
c1.call                    # => 2
c1.call                    # => 3

c2 = make_counter

puts "c1 = #{c1.call}, c2 = #{c2.call}"
# >> c1 = 4, c2 = 1
```

the big deal

executable data

compact syntax

crucial because of pervasiveness

heavily used in infrastructure

open classes

about classpaths

```
>> puts $:
/Library/Ruby/Site/1.8
/Library/Ruby/Site/1.8/powerpc-darwin9.0
/Library/Ruby/Site/1.8/universal-darwin9.0
/Library/Ruby/Site
/System/Library/Frameworks/Ruby.framework/Versions/1.8/usr/lib/ruby/1.8
/System/Library/Frameworks/Ruby.framework/Versions/1.8/usr/lib/ruby/1.8/powerpc-darwin9.0
/System/Library/Frameworks/Ruby.framework/Versions/1.8/usr/lib/ruby/1.8/universal-darwin9.0
.
=> nil
```

```
[nealford| ~ ]=> jirb
irb(main):001:0> puts $:
/Users/nealford/bin/jruby-1.1RC2/lib/ruby/site_ruby/1.8
/Users/nealford/bin/jruby-1.1RC2/lib/ruby/site_ruby
/Users/nealford/bin/jruby-1.1RC2/lib/ruby/1.8
/Users/nealford/bin/jruby-1.1RC2/lib/ruby/1.8/java
lib/ruby/1.8
.
```

open classes

a class definition for a class that already appears on the classpath reopens the class

allows

- adding new methods
- overriding existing methods
- removing methods

open employee

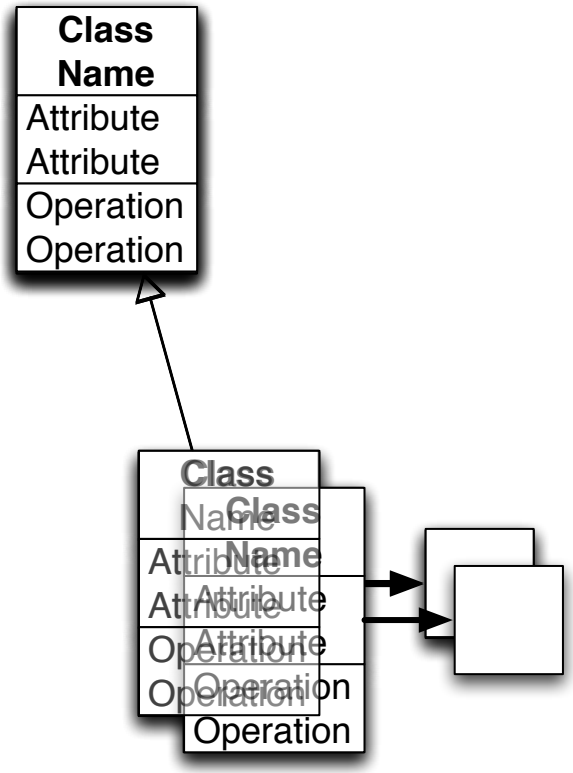
```
require File.dirname(__FILE__) + "/../01. classes/hr"

class Employee
  attr_accessor :birth_year

  def age
    Time.now.year - birth_year;
  end

end
```

open employee

```
e = Employee.new "Homer", 20000
e.birth_year = 1950
puts "Age is #{e.age}"
```

open classes redux

shadow meta-class



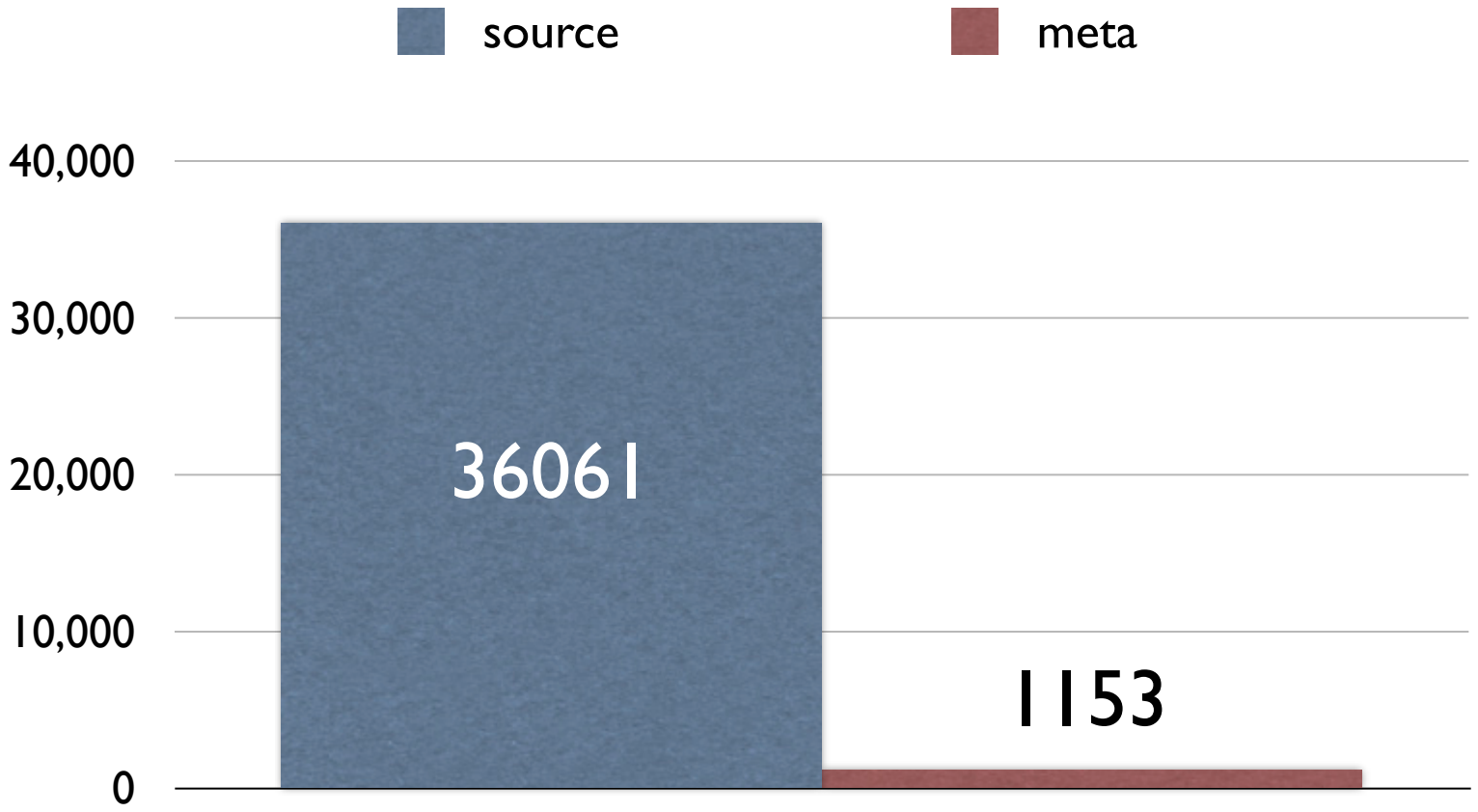
shadow meta-class

```
e = Employee.new "Homer", 20000
e.birth_year = 1950
puts "Age is #{e.age}"

def e.big_raise(by_perc)
    @salary += (@salary * (by_perc * 0.1))
end

old_salary = e.salary
e.big_raise(5)
puts "Big money! From #{old_salary} to #{e.salary}"
```

shadow meta-class

```ruby
def e.raise_salary_by(perc)
  @salary -= (@salary * (perc * 0.01))
end

old_salary = e.salary
e.raise_salary_by(5)
puts "Small money! From #{old_salary} to #{e.salary}"
```

meta-programming



modules (aka mixins)

allows logical grouping of classes, methods, and constants

namespaces

```
class TestEmployee < Test::Unit::TestCase
```

```
module Test
  module Unit
    class TestCase
```

mixins

when you **include** a module into a class, the module's methods are "mixed into" the class

methods defined in the module may interact with the class's parts

mixin

```ruby
module Debug
  def who_am_i
    "#{self.class.name} (\##{self.object_id}): #{self.to_s}"
  end
end
```

```ruby
class Employee
  include Debug
```

mixin

```
employees = Array.new
employees[0] = Employee.new("Homer", 200.0, 1995)
employees[3] = Manager.new("Monty", 3000.0, 1950,
        employees[2])

show(employees)

puts "\n\nWho are they?"
puts employees[0].who_am_i
puts employees[3].who_am_i
```

comparisons

```
class Employee
  include Comparable

  def <=>(other)
      name <=> other.name
  end
end

list = Array.new
list << Employee.new("Monty", 10000)
list << Employee.new("Homer", 50000)
list << Employee.new("Bart", 5000)
```

comparisons

```
puts list

list.sort!

puts list

puts "Monty vs. Homer", list[0] < list[1]
puts "Homer vs. Monty", list[0] > list[1]

puts "Homer is between Bart and Monty?",
    list[1].between?(list[0], list[2])
```


CONTRACT

comparisons

```
[nealford| ~/docs/dev/ruby/conf_jruby/10.mixins ]=> ruby comparisons.rb
Name: Monty, salary: 10000, hire year: 2007
Name: Homer, salary: 50000, hire year: 2007
Name: Bart, salary: 5000, hire year: 2007
Name: Bart, salary: 5000, hire year: 2007
Name: Homer, salary: 50000, hire year: 2007
Name: Monty, salary: 10000, hire year: 2007
Monty vs. Homer
true
Homer vs. Monty
false
Homer is between Bart and Monty?
true
```

violating handshakes

```
Name: Monty, salary: 10000, hire year: 2007
Name: Homer, salary: 50000, hire year: 2007
Name: Bart, salary: 5000, hire year: 2007
comparisons.rb:19:in `sort!': undefined method `<=>' for #<Employee:0x27650
@hire_year=2007, @salary=10000, @name="Monty"> (NoMethodError)
        from comparisons.rb:19
```

```ruby
puts list

list.sort!

puts list

puts "Monty vs. Homer", list[0] < list[1]
puts "Homer vs. Monty", list[0] > list[1]

puts "Homer is between Bart and Monty?",
    list[1].between?(list[0], list[2])
```

swing in jruby

just as ugly as in java!

```
require 'java'

import javax.swing.JFrame

class ClickAction
  include java.awt.event.ActionListener
  def actionPerformed(evt)
    msg = "<html>Hello from <b><u>JRuby</u></b><br>"
    javax.swing.JOptionPane.showMessageDialog(nil, msg)
  end
end

frame = JFrame.new("Hello Swing")
button = javax.swing.JButton.new("Click Me!")
button.add_action_listener(ClickAction.new)
frame.get_content_pane.add(button)
frame.set_default_close_operation(JFrame::EXIT_ON_CLOSE)
frame.pack
frame.visible = true
```

swing take 2

```
class BlockActionListener
  include java.awt.event.ActionListener

  def initialize(&block)
    super
    @block = block
  end

  def actionPerformed(e)
    @block.call(e)
  end
end
```

swing take 2

```
class JButton
  def initialize(name, &block)
    super(name)
    addActionListener(BlockActionListener.new(&block))
  end
end
```

```
clear_button = JButton.new("Clear") { name_field.text = "" }
```

swing, take 2

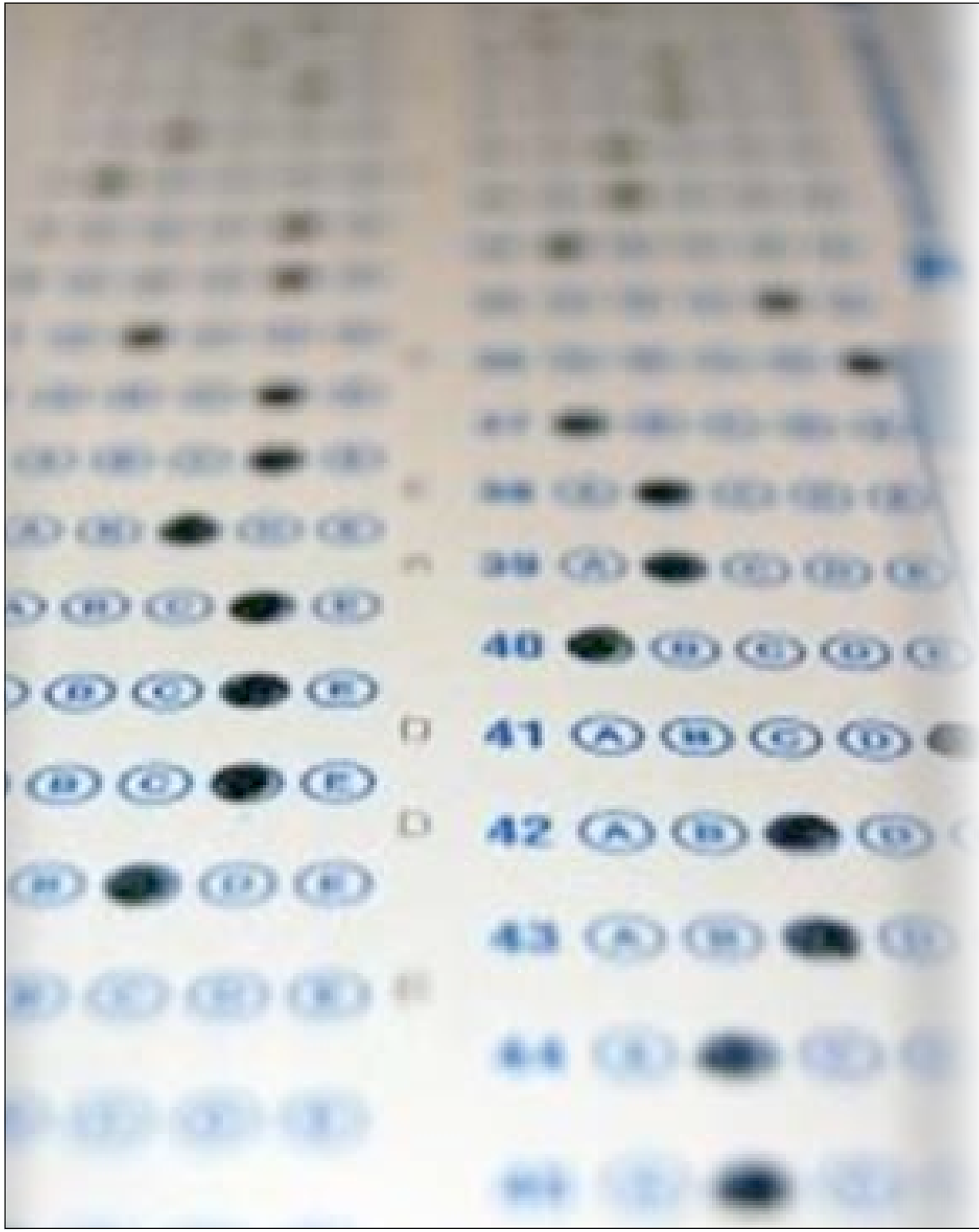
```
class HelloFrame < JFrame
  def initialize
    super("Hello Swing!")
    populate
    pack
    resizable = false
    defaultCloseOperation = JFrame::EXIT_ON_CLOSE
  end
```

```
HelloFrame.new.visible = true
```

```
def populate
  name_panel = JPanel.new
  name_panel.add JLabel.new("Name:")
  name_field = JTextField.new(20)
  name_panel.add name_field

  button_panel = JPanel.new
  greet_button = JButton.new "Greet" do
    name = name_field.text
    msg = %(<html>Hello <span style="color:red">#{name}</span>!</html>)
    JOptionPane.showMessageDialog self, msg
  end
  button_panel.add greet_button
  clear_button = JButton.new("Clear") { name_field.text = "" }

  button_panel.add clear_button

  contentPane.add name_panel, BorderLayout::CENTER
  contentPane.add button_panel, BorderLayout::SOUTH
end
```

jruby adds “humane interface” methods to standard java classes

`<=>`, `<<`, **`between?`**



testing java with jruby

the java part

```
public interface Order {
    void fill(Warehouse warehouse);

    boolean isFilled();
}
public interface Warehouse {
    public void add(String item, int quantity);

    int getInventory(String product);

    boolean hasInventory(String product, int quantity);

    void remove(String product, int quantity);
}
```

testing `fill()`

```
public void fill(Warehouse warehouse) {
    if (warehouse.hasInventory(_product, _quantity)) {
        warehouse.remove(_product, _quantity);
        _filled = true;
    } else
        _filled = false;

}
```

jmock

```
@RunWith(JMock.class)
public class OrderInteractionTester {
    private static String TALISKER = "Talisker";
    Mockery context = new JUnit4Mockery();

    @Test public void fillingRemovesInventoryIfInStock() {
        Order order = new OrderImpl(TALISKER, 50);
        final Warehouse warehouse = context.mock(Warehouse.class);

        context.checking(new Expectations() {{
            one (warehouse).hasInventory(TALISKER, 50); will(returnValue(true));
            one (warehouse).remove(TALISKER, 50);
        }});

        order.fill(warehouse);
        assertThat(order.isFilled(), is(true));
        context.assertIsSatisfied();
    }
```

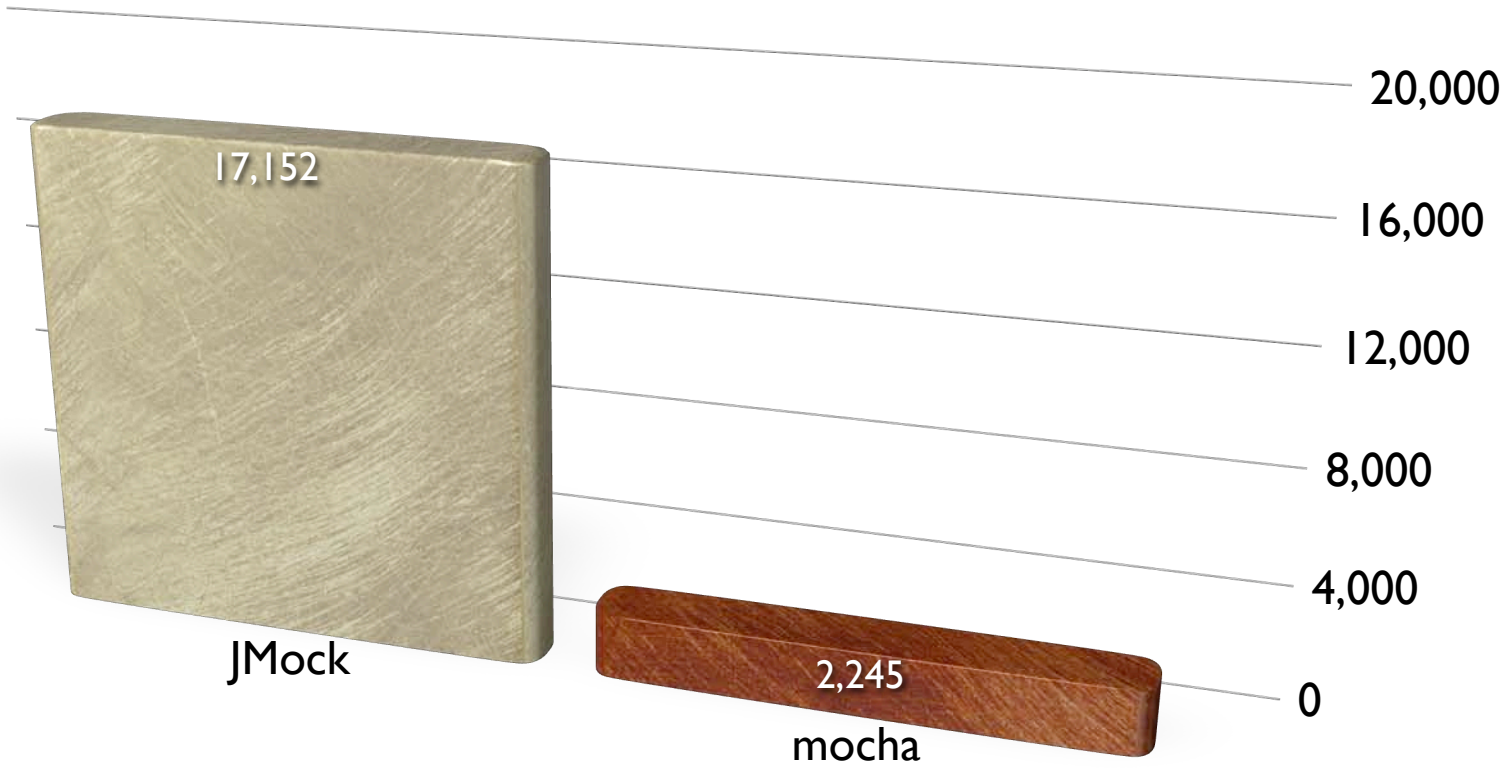
mocha

```
require "java"
require "Warehouse.jar"
%w(OrderImpl Order Warehouse WarehouseImpl).each { |f|
   include_class "com.nealford.conf.jmock.warehouse.#{f}"
}

class OrderInteractionTest < Test::Unit::TestCase
  TALISKER = "Talisker"

  def test_filling_removes_inventory_if_in_stock
    order = OrderImpl.new(TALISKER, 50)
    warehouse = Warehouse.new
    warehouse.stubs(:hasInventory).with(TALISKER, 50).returns(true)
    warehouse.stubs(:remove).with(TALISKER, 50)

    order.fill(warehouse)
    assert order.is_filled
  end
```

what does it take???

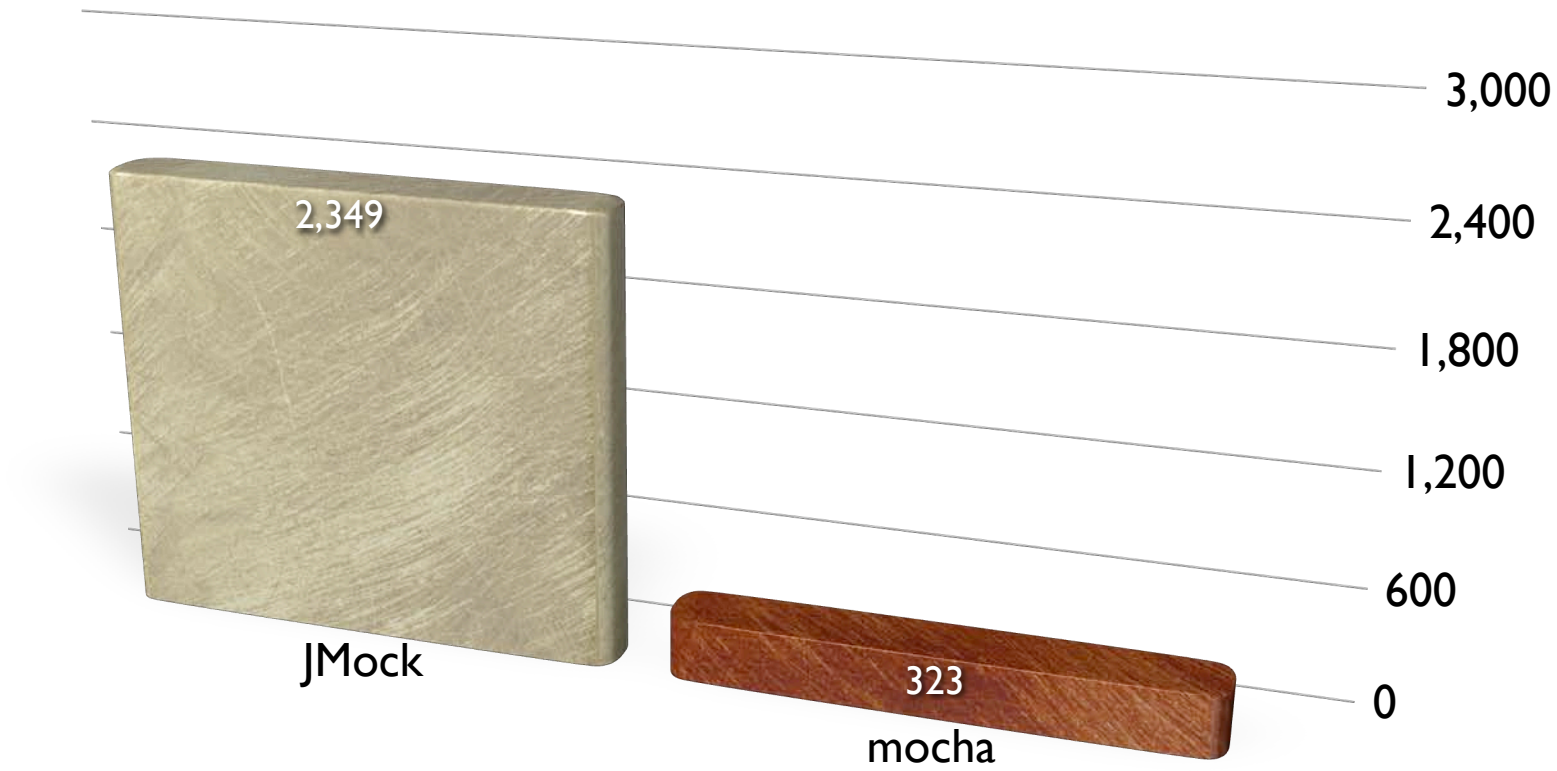
```
class Object

  def expects(symbol)
    method = stubba_method.new(stubba_object, symbol)
    $stubba.stub(method)
    mocha.expects(symbol, caller)
  end

  def stubs(symbol)
    method = stubba_method.new(stubba_object, symbol)
    $stubba.stub(method)
    mocha.stubs(symbol, caller)
  end

  def verify
    mocha.verify
  end

end
```

jmock vs mocha loc



jmock has 7.5 times as many lines of code

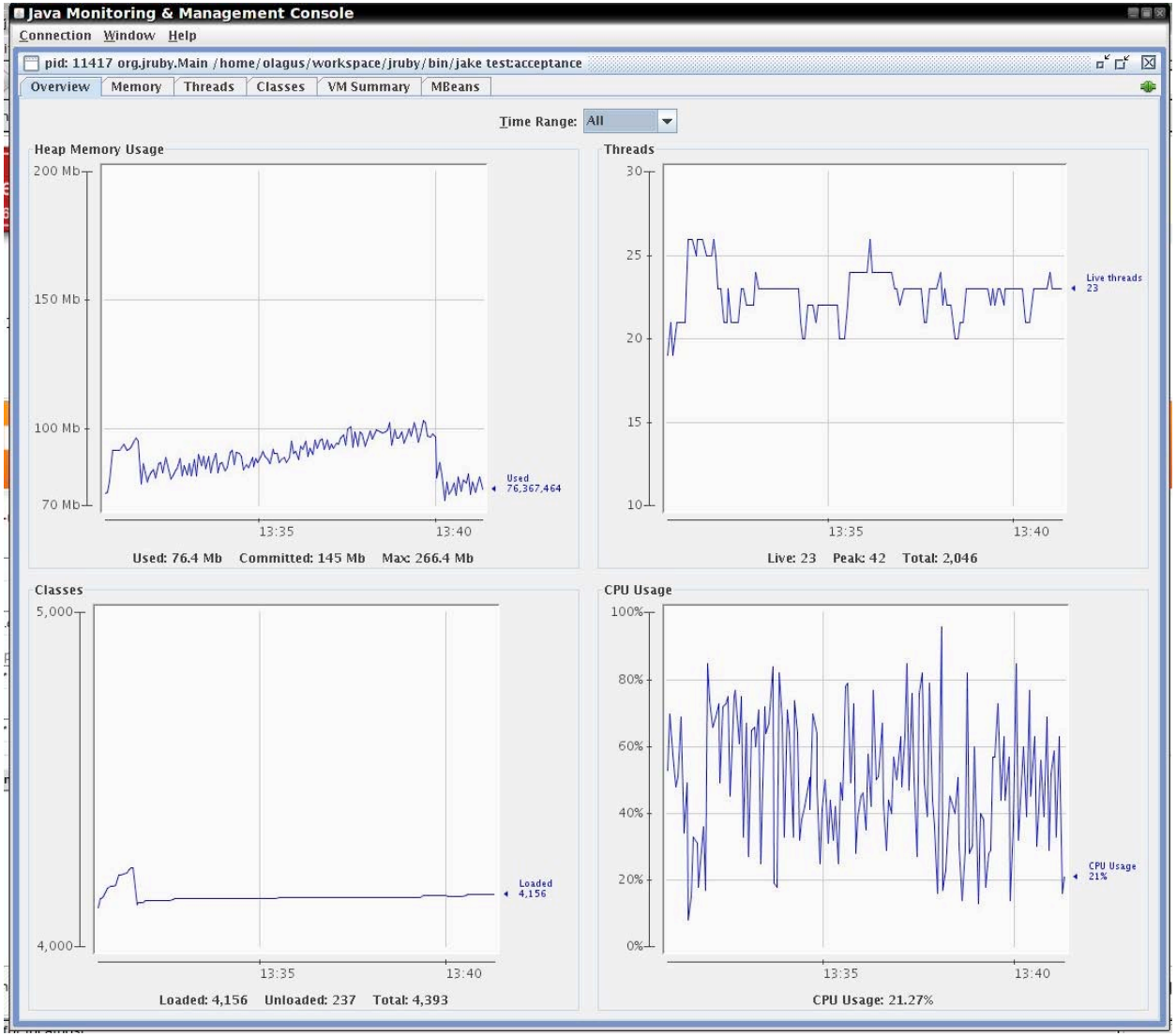
jmock vs mocha cc



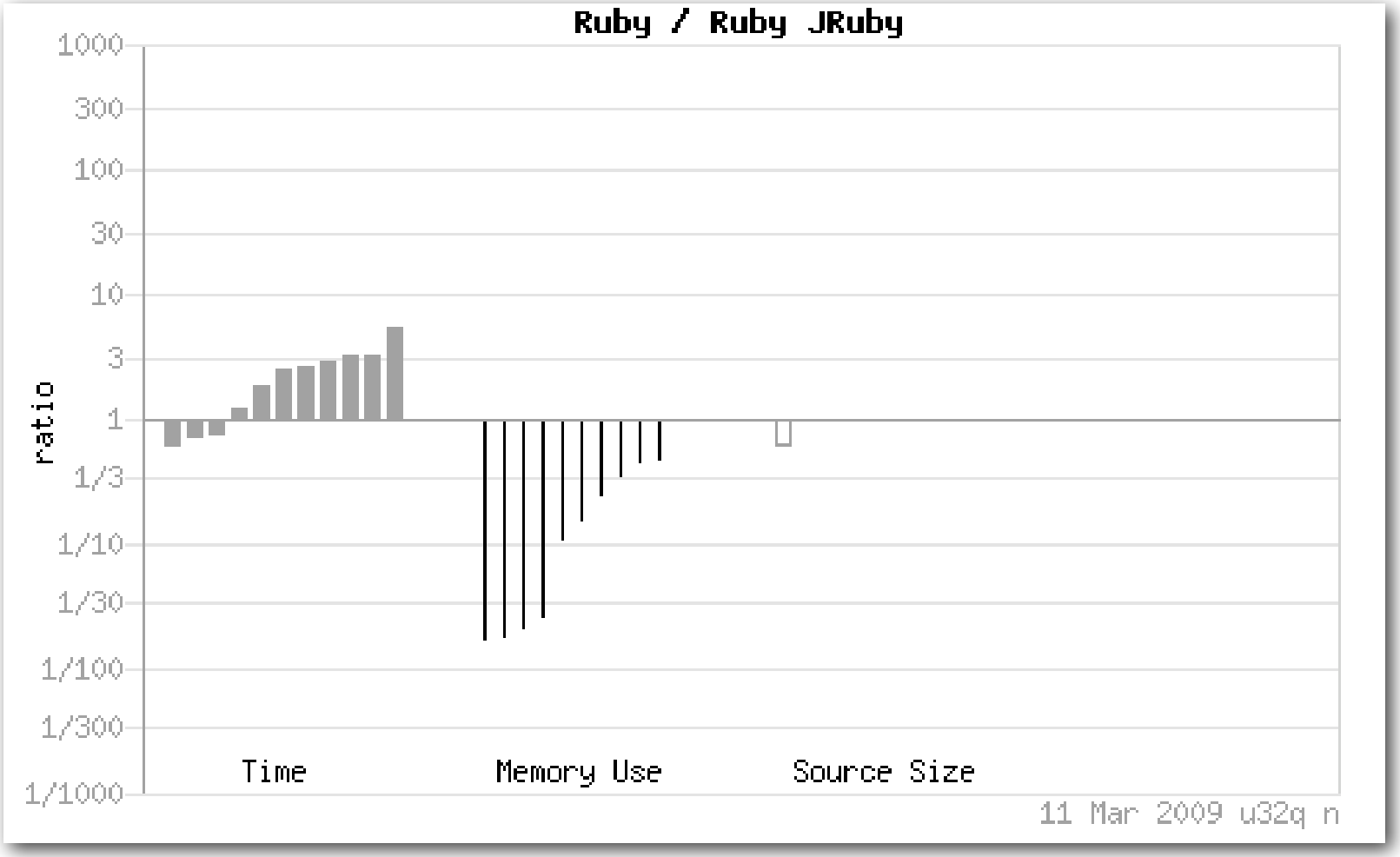
jmock has 7.2 times the complexity of mocha

jruby runs ruby on rails

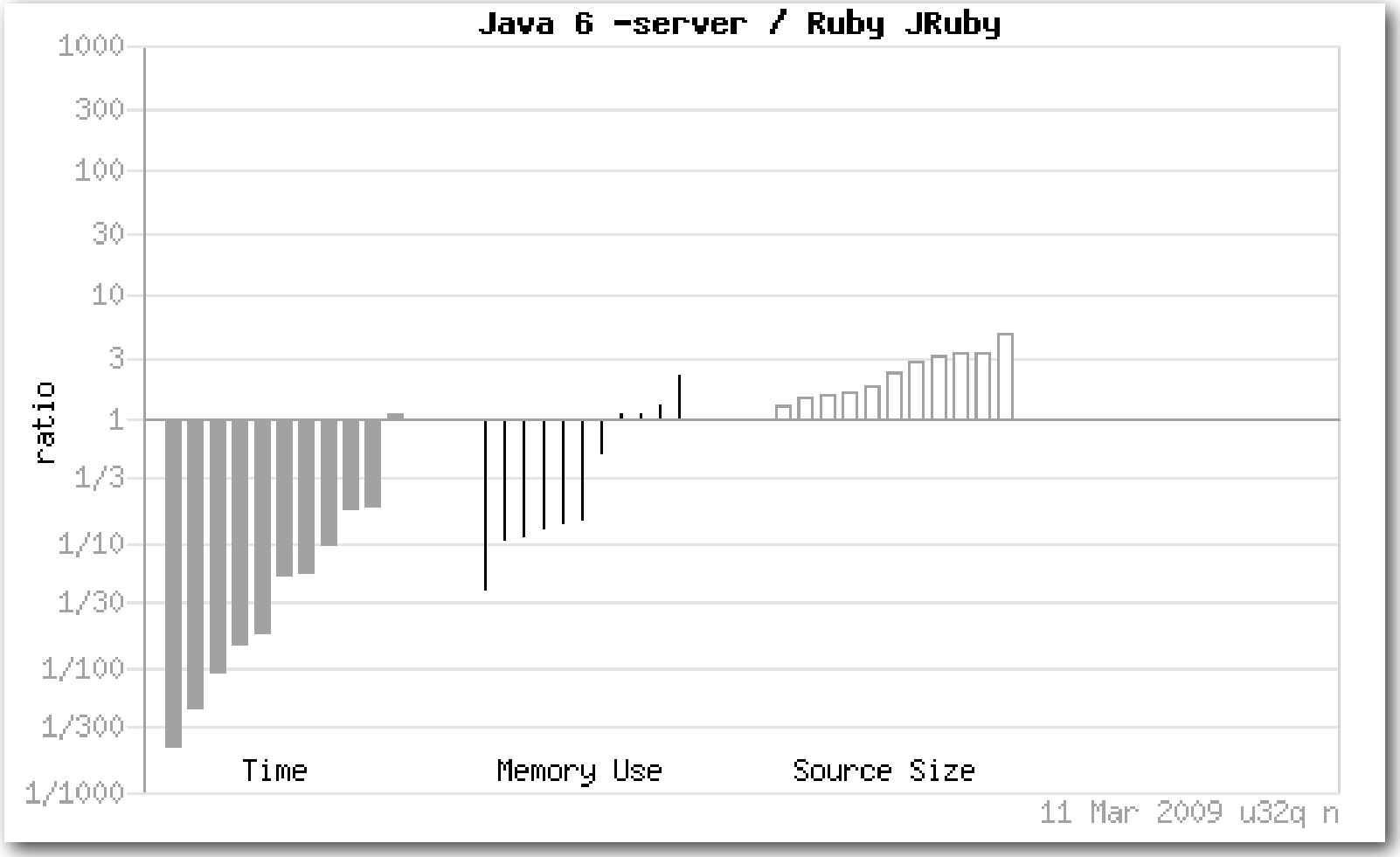
fun with jruby



benchmarks:
ruby vs jruby



benchmark:
java vs jruby



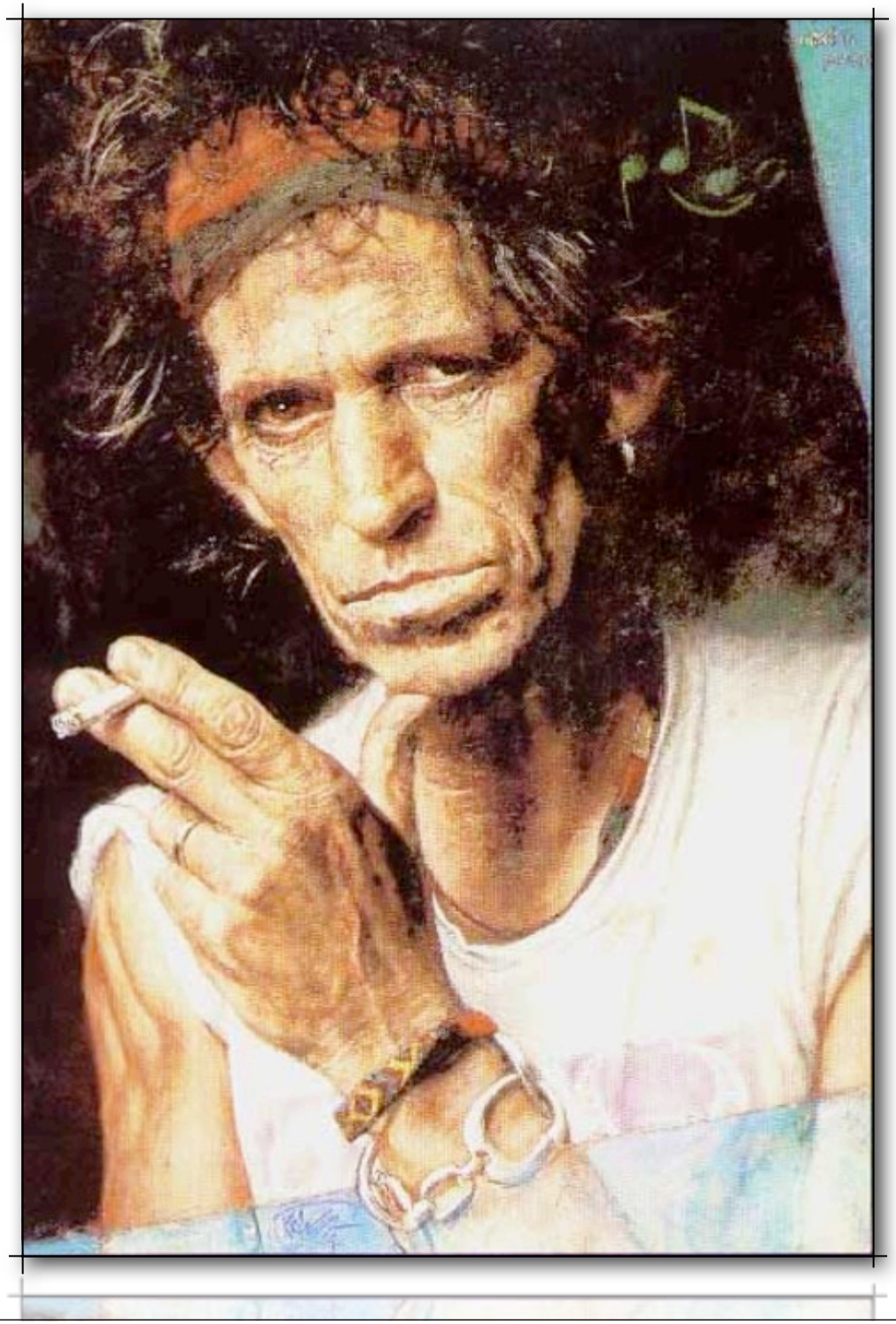
remember, back in 1997...

java was considered too slow for "serious development"

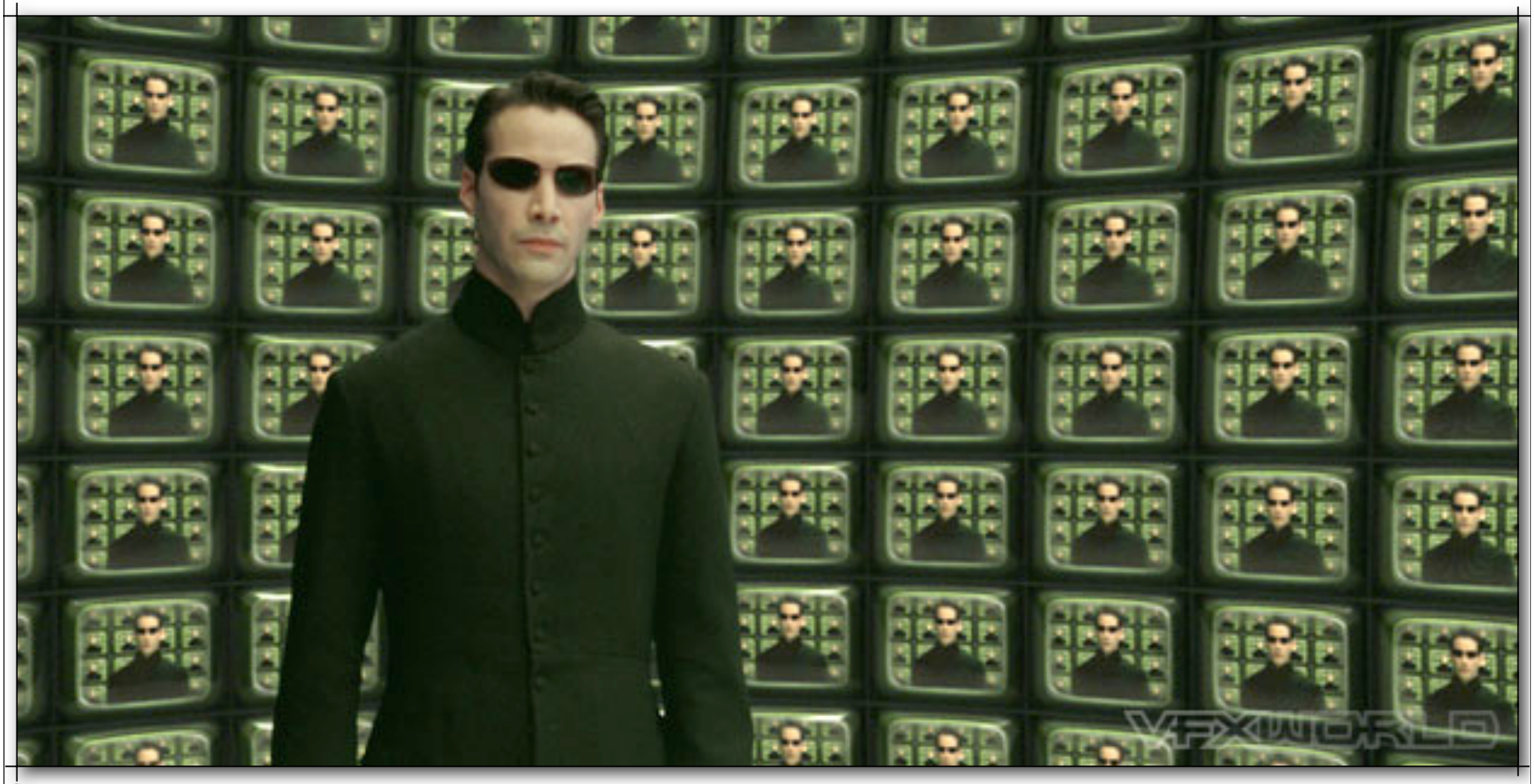
make it right...

...then make it fast

java in 2008 ==



/j?ruby/



?'s

please fill out the session evaluations

samples at `github.com/nealford`

NEAL FORD software architect / meme wrangler

ThoughtWorks

nford@thoughtworks.com
3003 Summit Boulevard, Atlanta, GA 30319
www.nealford.com
www.thoughtworks.com
memeagora.blogspot.com

resources

jruby site
http://jruby.codehaus.org/

charles nutter's blog
http://headius.blogspot.com/

ola bini's blog
http://ola-bini.blogspot.com/

benchmarks
http://shootout.alioth.debian.org/gp4sandbox/benchmark.php