



Joining up the dots

SYNTHESIS



Stuart Caborn
George Malamidis

“Synthesis”

the formation of something complex or
coherent by combining simpler things



Synthesized testing

Combine lightweight tests to build confidence
that our system is complete and reduce the
need for large, overarching tests



Test code is code, too

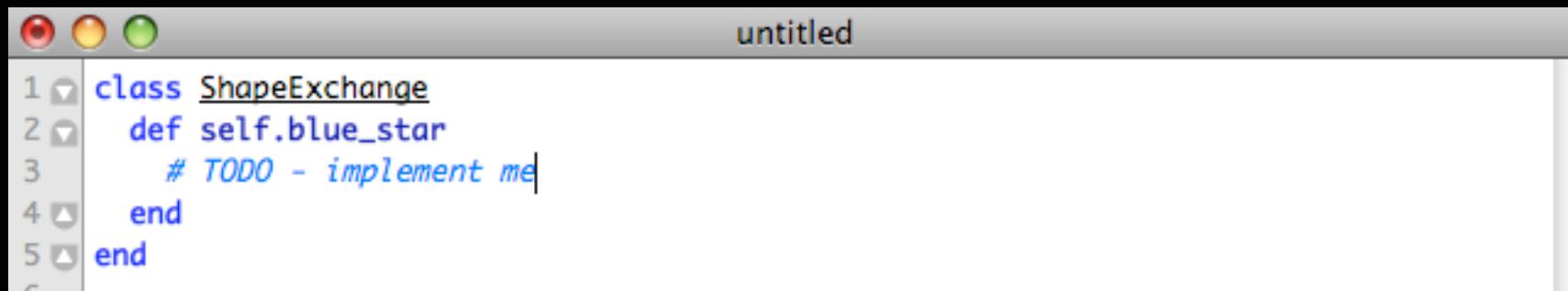




Technical debt



A story of shapes

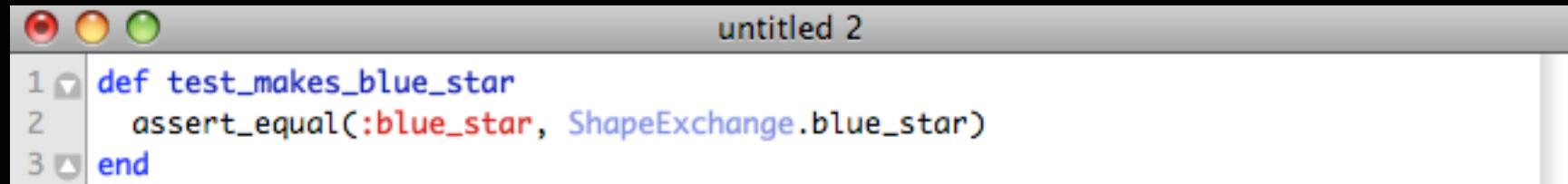


```
1 class ShapeExchange
2   def self.blue_star
3     # TODO - implement me
4   end
5 end
```



```
1 class ShapeShifter
2   def self.shift(*shapes)
3     really_complicated_shape_cooking(*shapes)
4   end
5 end
```

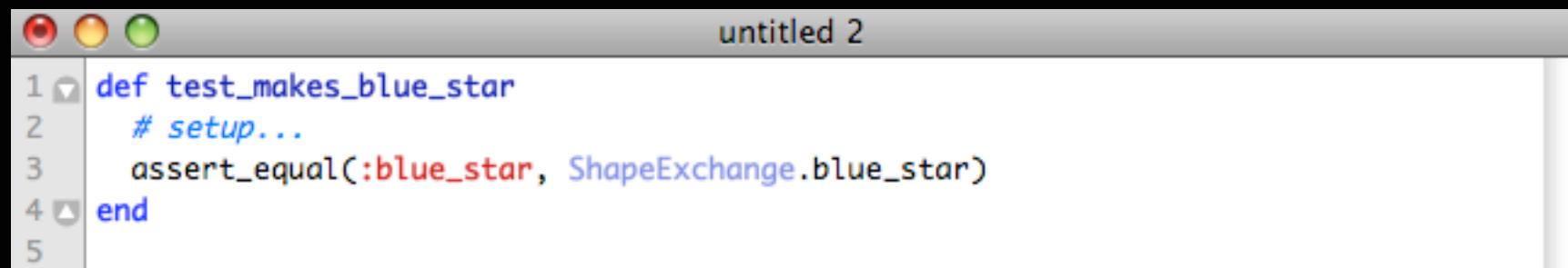
Simple?



A screenshot of a Mac OS X application window titled "untitled 2". The window contains the following Ruby code:

```
1 def test_makes_blue_star
2   assert_equal(:blue_star, ShapeExchange.blue_star)
3 end
```

Hang on...



A screenshot of a Mac OS X application window titled "untitled 2". The window contains the following Ruby code:

```
1 def test_makes_blue_star
2   # setup...
3   assert_equal(:blue_star, ShapeExchange.blue_star)
4 end
5
```

Just a little setup



A screenshot of a text editor window titled "untitled 2". The code in the editor is:

```
1 def test_makes_blue_star
2   # setup...
3   connect_to_db
4
5   assert_equal(:blue_star, ShapeExchange.blue_star)
6 end
```

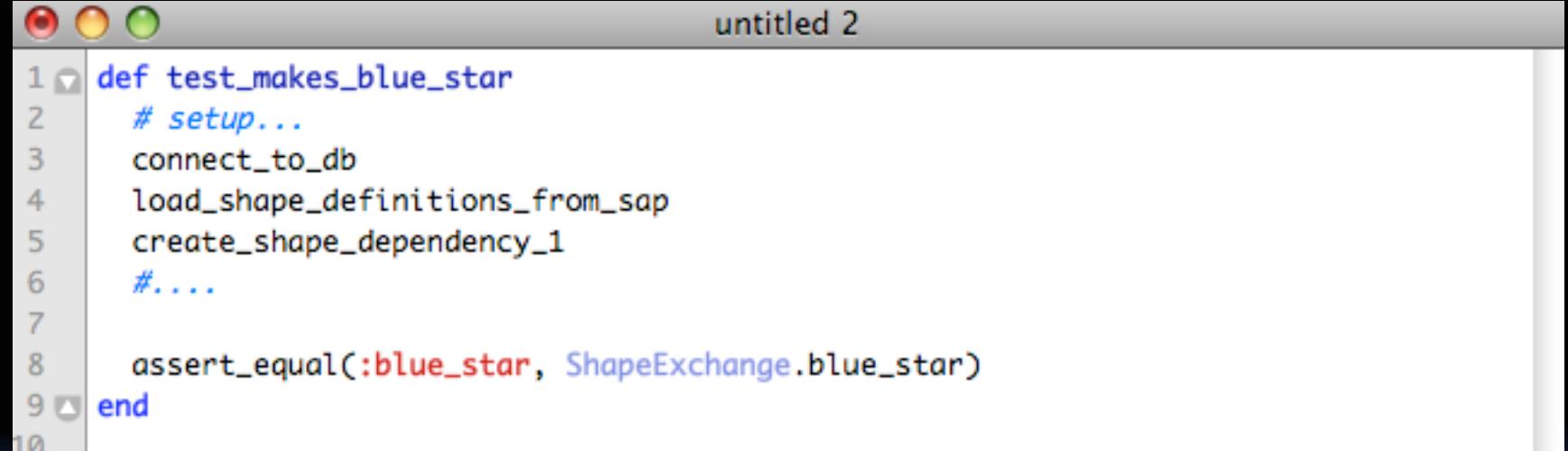
Just a little more setup



A screenshot of a Mac OS X application window titled "untitled 2". The window contains the following Ruby code:

```
1 def test_makes_blue_star
2   # setup...
3   connect_to_db
4   load_shape_definitions_from_sap
5
6   assert_equal(:blue_star, ShapeExchange.blue_star)
7 end
```

You get the idea

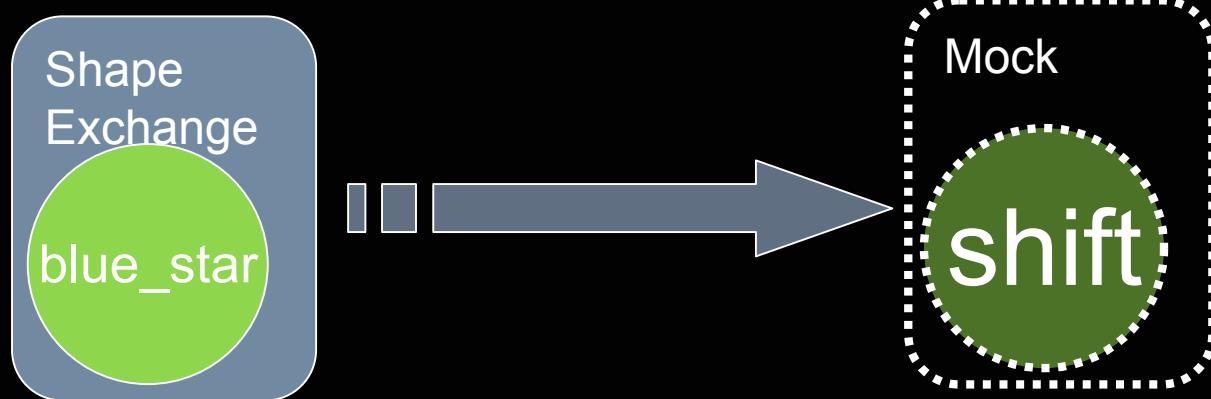


A screenshot of a code editor window titled "untitled 2". The window contains a Ruby script with line numbers 1 through 10 on the left. The code defines a test method named "test_makes_blue_star" which includes setup steps like connecting to a database and loading shape definitions, followed by an assertion that a shape named "blue_star" is equal to one from the "ShapeExchange" module.

```
1 def test_makes_blue_star
2   # setup...
3   connect_to_db
4   load_shape_definitions_from_sap
5   create_shape_dependency_1
6   #....
7
8   assert_equal(:blue_star, ShapeExchange.blue_star)
9 end
10
```

Mock the interaction?

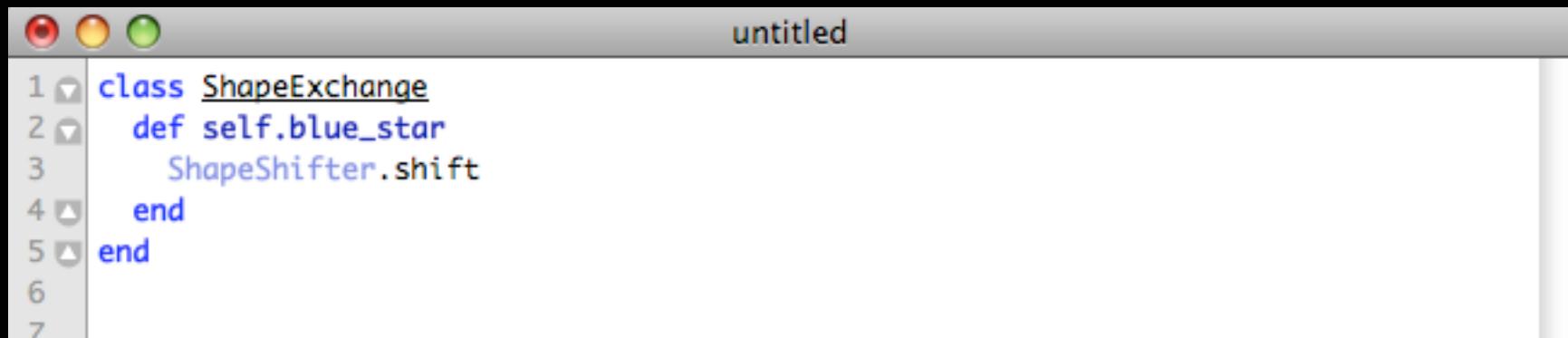
Red



A screenshot of a code editor window titled "untitled 2". The code is as follows:

```
1 def test_makes_blue_star
2   ShapeShifter.expects(:shift)
3   ShapeExchange.blue_star
4 end
```

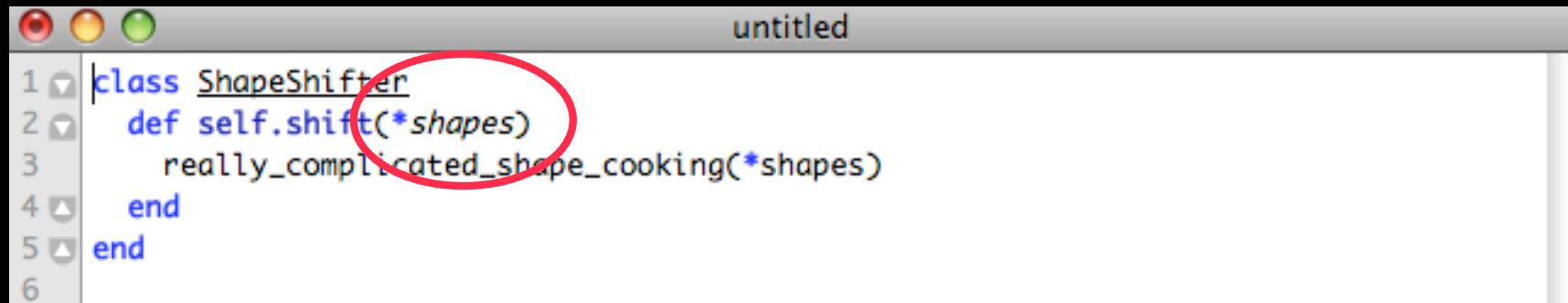
Green



A screenshot of a Mac OS X application window titled "untitled". The window contains the following Ruby code:

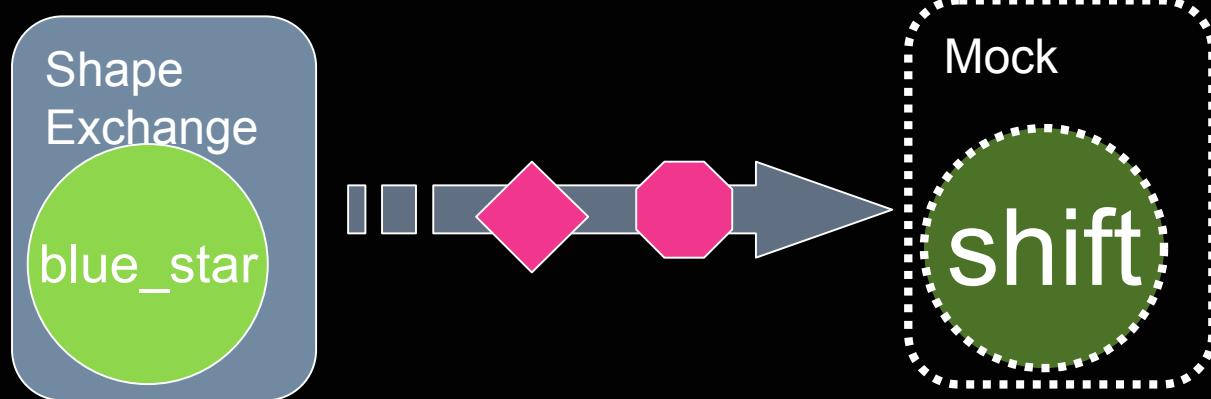
```
1 class ShapeExchange
2   def self.blue_star
3     ShapeShifter.shift
4   end
5 end
6
7
```

Crash



```
1 class ShapeShifter
2   def self.shift(*shapes)
3     really_complicated_shape_cooking(*shapes)
4   end
5 end
6
```

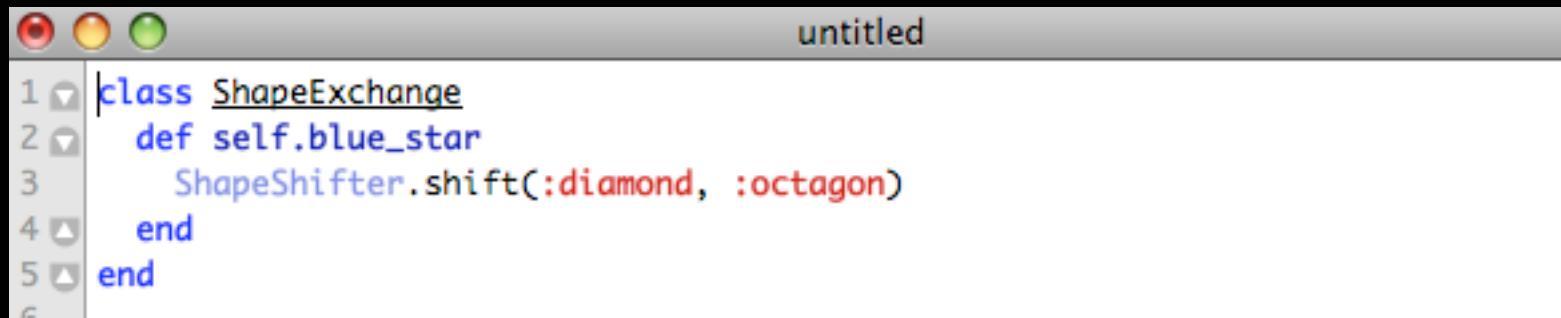
Red



A screenshot of an IDE window titled 'untitled 2'. The code editor displays the following Ruby test script:

```
1 def test_makes_blue_star
2   ShapeShifter.expects(:shift).with(:diamond, :octagon)
3   ShapeExchange.blue_star
4 end
```

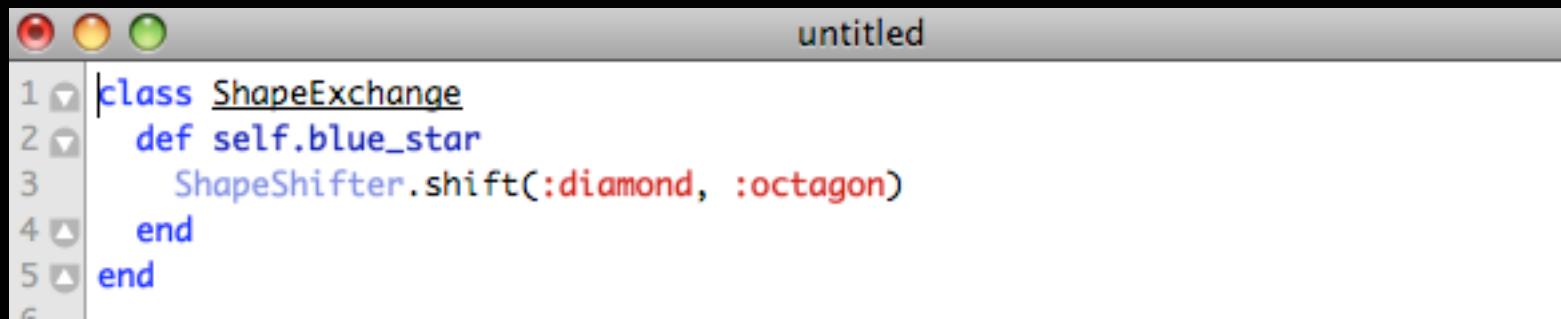
Green



A screenshot of a Ruby code editor window titled "untitled". The window has three colored buttons (red, yellow, green) in the top-left corner. The code in the editor is:

```
1 class ShapeExchange
2   def self.blue_star
3     ShapeShifter.shift(:diamond, :octagon)
4   end
5 end
```

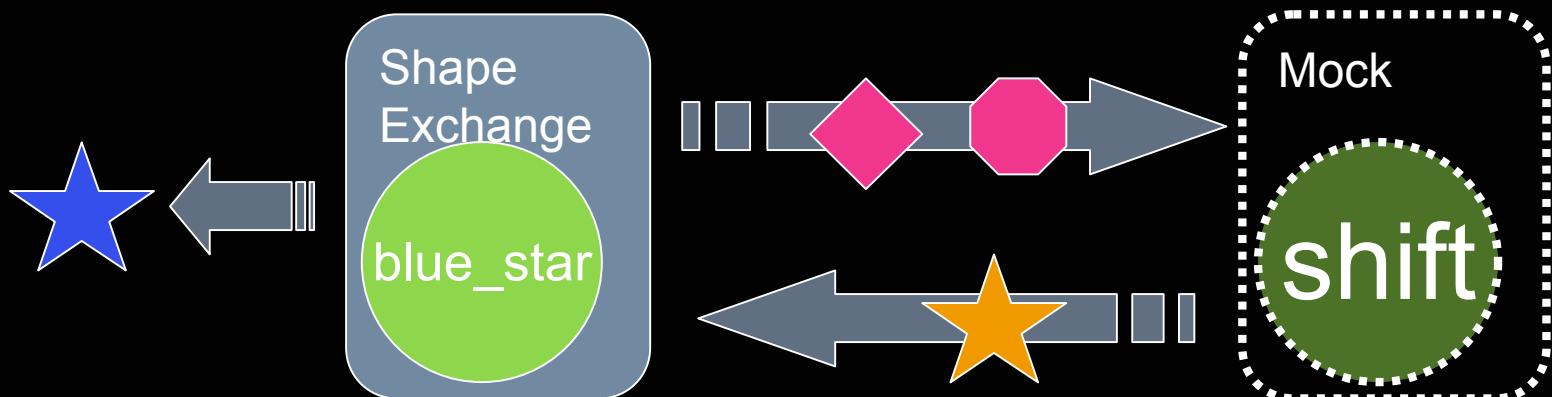
Incomplete



A screenshot of a text editor window titled "untitled". The code is incomplete:

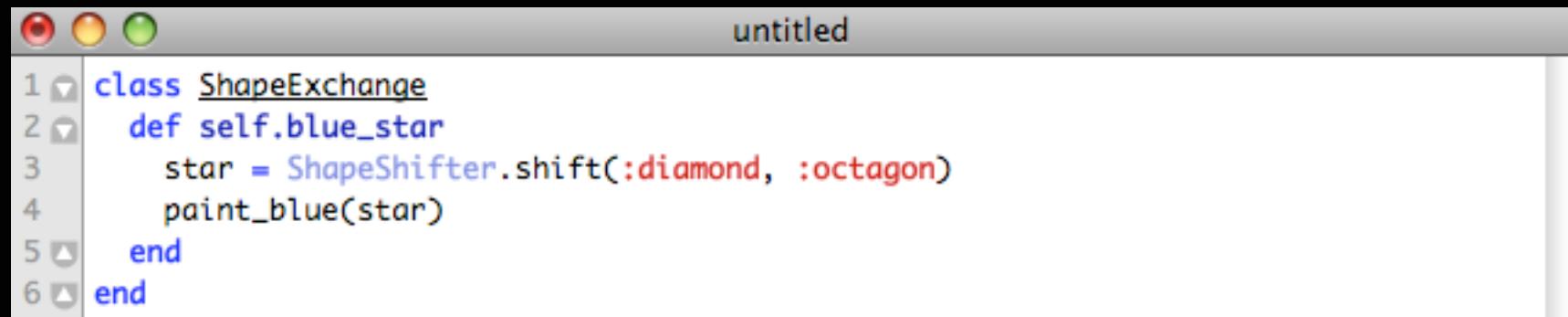
```
1 class ShapeExchange
2   def self.blue_star
3     ShapeShifter.shift(:diamond, :octagon)
4   end
5 end
```

Red



```
untitled 2
1 def test_makes_blue_star
2   ShapeShifter.expects(:shift).with(:diamond, :octagon).returns(:star)
3   assert_equal(:blue_star, ShapeExchange.blue_star)
4 end
```

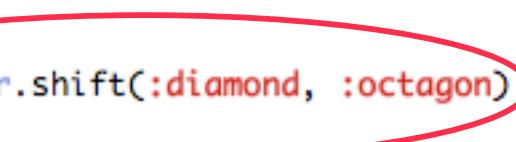
Green



A screenshot of a Ruby code editor window titled "untitled". The window has three colored icons (red, yellow, green) in the top-left corner. The code in the editor is as follows:

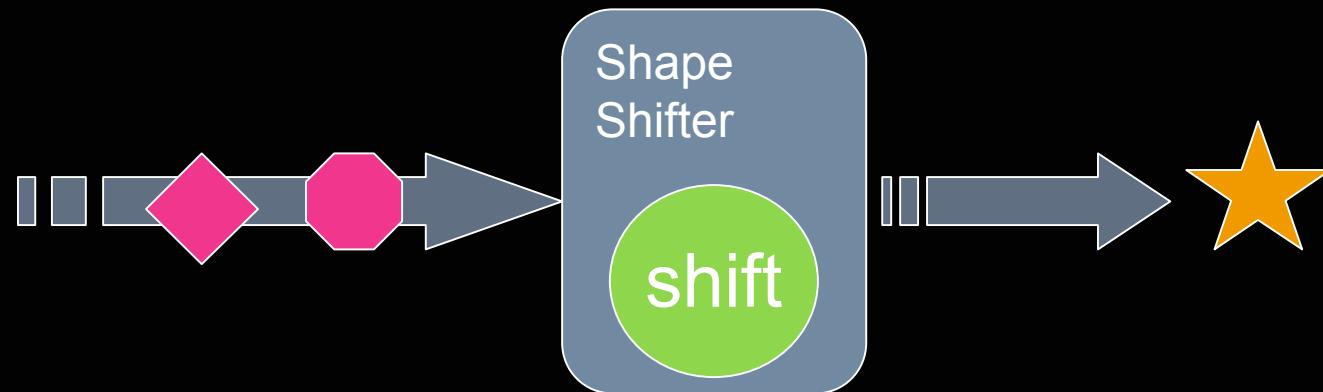
```
1 class ShapeExchange
2   def self.blue_star
3     star = ShapeShifter.shift(:diamond, :octagon)
4     paint_blue(star)
5   end
6 end
```

Where?



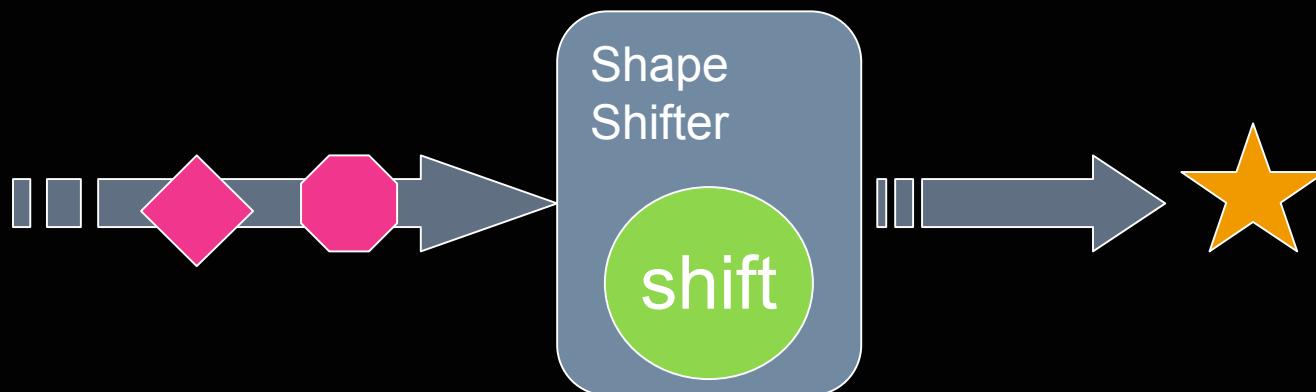
```
untitled
1 class ShapeExchange
2   def self.blue_star
3     star = ShapeShifter.shift(:diamond, :octagon)
4     paint_blue(star)
5   end
6 end
```

Can we prove this happens?



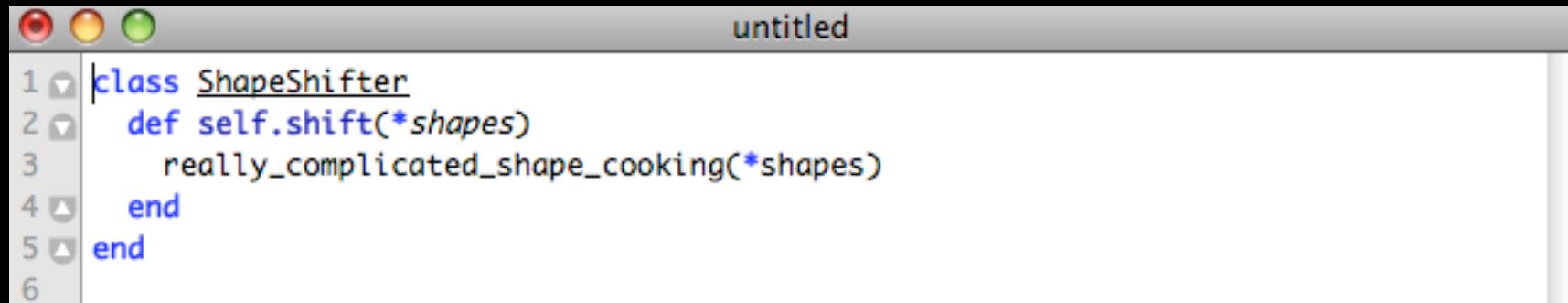
?

Red



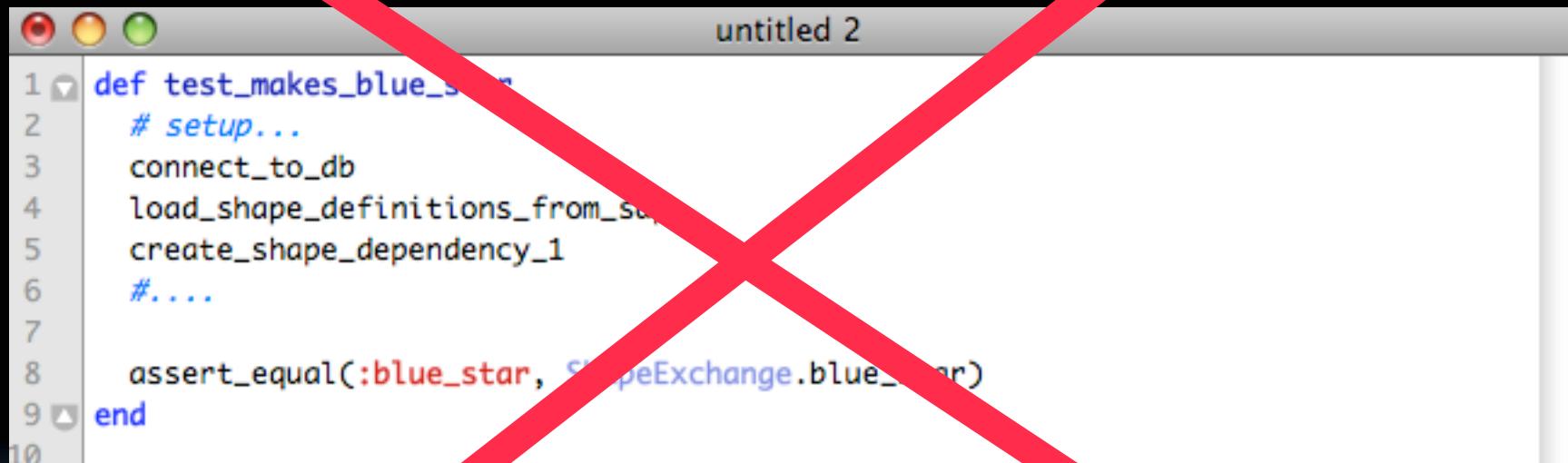
```
untitled 2
1 def test_shifts_start_from_diamond_and_octagon
2   assert_equal(:star, ShapeShifter.shift(:diamond, :octagon))
3 end
4
```

Green



```
1 | class ShapeShifter
2 |   def self.shift(*shapes)
3 |     really_complicated_shape_cooking(*shapes)
4 |   end
5 | end
6 |
```

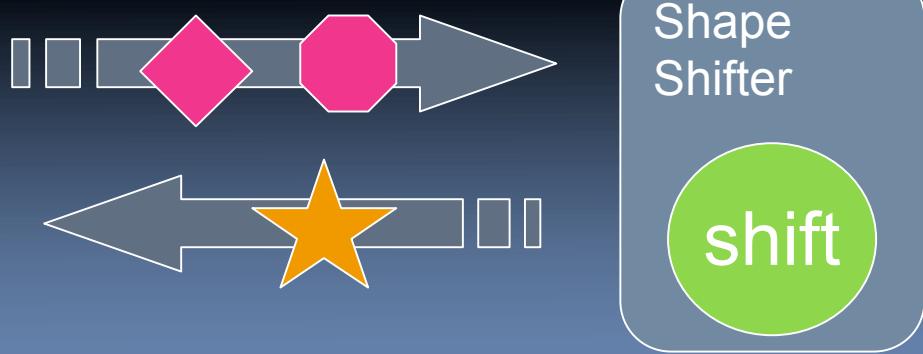
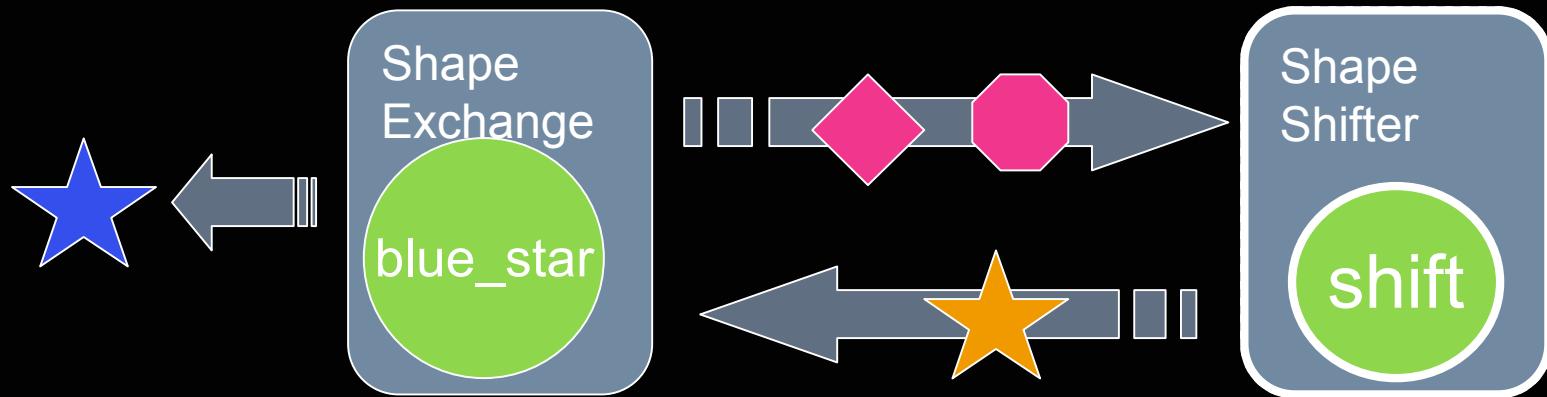
Reduced technical debt



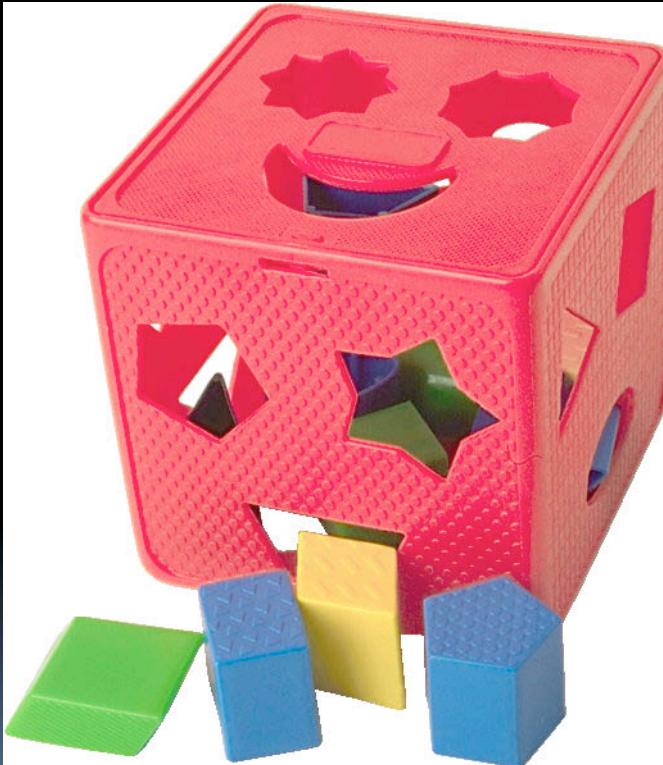
A screenshot of a code editor window titled "untitled 2". The code in the editor is a Ruby test script:

```
1 def test_makes_blue_star
2   # setup...
3   connect_to_db
4   load_shape_definitions_from_database
5   create_shape_dependency_1
6   #...
7
8   assert_equal(:blue_star, ShapeExchange.blue_star)
9 end
10
```

Synthesizing a bigger test



Synthesis



Works well with

Existing tools

- RSpec
- Mocha
- zentest
- autotest

Testing Practices

- TDD
- BDD

Synthesis overview

Used through the Rake task Synthesis::Task

```
33 Synthesis::Task.new('synthesis:test:rspec') do |t|
34   t.adapter = :rspec
35   t.pattern = 'test_project/rspec/*_spec.rb'
36 end
```

Synthesis overview

First pass

Collect mocked expectations

Second Pass

Correlate mocked expectations with REAL
calls to REAL objects.

```
1 require File.dirname(__FILE__) + "/helper"
2
3 class DataBranderTest < Test::Unit::TestCase
4   def test_saves_branded_with_storage
5     storage = Storage.new('whatever')
6     storage.expects(:save).with('Hello from Prague')
7     data_brander = DataBrander.new(storage)
8     data_brander.save_branded('Hello')
9   end
10 end
11
```

```
1 class DataBrander
2   def initialize(storage)
3     @storage = storage
4   end
5
6   def save_branded(data)
7     @storage.save("#{data} from Prague")
8   end
9 end
```

```
1 require File.dirname(__FILE__) + "/helper"
2
3 class StorageTest < Test::Unit::TestCase
4   def test_saves_to_file
5     Storage.new('test-file.txt').save('hello')
6     assert_equal('hello', File.read('test-file.txt'))
7   ensure
8     FileUtils.rm_f('test-file.txt') if File.file?('test-file.txt')
9   end
10 end
```

```
1 class Storage
2   def initialize(filename)
3     @filename = filename
4   end
5
6   def save(data)
7     File.open(@filename, 'w') {|f| f << data}
8   end
9 end
```

```
RakeMate v2.0.0 running on Ruby v1.8.6 (/usr/bin/ruby)
>>> /Users/gmalamid-devel/ruby/whatever_code/synthesis_examples/Rakefile

(in /Users/gmalamid-devel/ruby/whatever_code/synthesis_examples)
Loaded suite /Library/Ruby/Gems/1.8/gems/rake-0.8.1/lib/rake/rake_test_loader
Started
..
Finished in 0.002593 seconds.

2 tests, 2 assertions, 0 failures, 0 errors
```

```
1 require "simple"
2
3 storage = Storage.new("hello.txt")
4 data_brandner = DataBrander.new(storage)
5 data_brandner.save_branded("Hello")
```

```
1 class Storage
2   def initialize(filename)
3     @filename = filename
4   end
5
6   def save(data)
7     File.open(@filename, 'w') { |f| f << data}
8   end
9 end
```

```
1 class Storage
2   def initialize(filename)
3     @filename = filename
4   end
5
6   def save(data, mode)
7     File.open(@filename, mode) { |f| f << data}
8   end
9 end
```

```
1 require File.dirname(__FILE__) + "/helper"
2
3 class StorageTest < Test::Unit::TestCase
4   def test_saves_to_file
5     Storage.new('test-file.txt').save('hello', 'w')
6     assert_equal('hello', File.read('test-file.txt'))
7   ensure
8     FileUtils.rm_f('test-file.txt') if File.file?('test-file.txt')
9   end
10 end
```

```
RakeMate v2.0.0 running on Ruby v1.8.6 (/usr/bin/ruby)
>>> /Users/gmalamid-devel/ruby/whatever_code/synthesis_examples/Rakefile

(in /Users/gmalamid-devel/ruby/whatever_code/synthesis_examples)
Loaded suite /Library/Ruby/Gems/1.8/gems/rake-0.8.1/lib/rake/rake_test_loader
Started
..
Finished in 0.001725 seconds.

2 tests, 2 assertions, 0 failures, 0 errors
```

```
1 require "simple"
2
3 storage = Storage.new("hello.txt")
4 data_brandner = DataBrander.new(storage)
5 data_brandner.save_branded("Hello")
```

ArgumentError: wrong number of arguments (1 for 2)

method save in **data_brander.rb** at line 7
method save_branded in **data_brander.rb** at line 7
at top level in **main.rb** at line 5

Program exited.

```
1 class DataBrander
2   def initialize(storage)
3     @storage = storage
4   end
5
6   def save_branded(data)
7     @storage.save("#{data} from Prague")
8   end
9 end
```

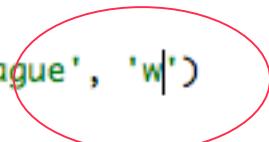
```
17     desc "Synthesize Simple"
18   Synthesis::Task.new('simple') do |t|
19     t.pattern = 'simple/test/*_test.rb'
20   end
```

```
[Synthesis] Collecting expectations...
Loaded suite /usr/bin/rake
Started
...
Finished in 0.004425 seconds.

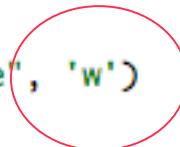
2 tests, 2 assertions, 0 failures, 0 errors
[Synthesis] Verifying expectation invocations...
Loaded suite /usr/bin/rake
Started
...
Finished in 0.002309 seconds.

2 tests, 2 assertions, 0 failures, 0 errors
[Synthesis]
[Synthesis] Tested Expectations:
[Synthesis]
[Synthesis] Untested Expectations:
[Synthesis] Storage.new.save(String) in ./simple/test/data_brander_test.rb:6:in `test_saves_branded_with_storage'
[Synthesis]
[Synthesis] Ignoring:
[Synthesis]
[Synthesis] FAILED.
```

```
1 require File.dirname(__FILE__) + "/helper"
2
3 class DataBranderTest < Test::Unit::TestCase
4   def test_saves_branded_with_storage
5     storage = Storage.new('whatever')
6     storage.expects(:save).with('Hello from Prague', 'w|')
7     data_brander = DataBrander.new(storage)
8     data_brander.save_branded('Hello')
9   end
10 end
11
```



```
1 class DataBrander
2   def initialize(storage)
3     @storage = storage
4   end
5
6   def save_branded(data)
7     @storage.save("#{data} from Prague", 'w')
8   end
9 end
```



```
[Synthesis] Collecting expectations...
Loaded suite /usr/bin/rake
Started
..
Finished in 0.002872 seconds.

2 tests, 2 assertions, 0 failures, 0 errors
[Synthesis] Verifying expectation invocations...
Loaded suite /usr/bin/rake
Started
..
Finished in 0.001661 seconds.

2 tests, 2 assertions, 0 failures, 0 errors
[Synthesis]
[Synthesis] Verified 1 expectations
[Synthesis] SUCCESS.
```

```
1 class CreatePeople < Sequel::Migration
2   def up
3     create_table :people do
4       primary_key :id
5       varchar :first_name
6       varchar :last_name
7     end
8   end
9
10  def down
11    drop_table :people
12  end
13 end
14
15 DB = Sequel.sqlite
16 CreatePeople.apply(DB, :up)
```

```
1 require File.dirname(__FILE__) + "/helper"
2
3 class GreeterTest < Test::Unit::TestCase
4   def test_says_hello
5     person = stub(:first_name => 'John', :last_name => 'Doe')
6     Person.expects(:find_by_first_name).with("John").returns(person)
7     assert_equal("Hello, John Doe", Greeter.say_hello("John"))
8   end
9 end
```

```
1 class Greeter
2   def self.say_hello(first_name)
3     person = Person.find_by_firstname(first_name)
4     "Hello, #{person.first_name} #{person.last_name}"
5   end
6 end
```

```
1 require File.dirname(__FILE__) + "/helper"
2
3 class PersonTest < Test::Unit::TestCase
4   def test_find_by_name
5     Person.create(:first_name => "John", :last_name => "Doe")
6     assert_equal("Doe", Person.find_by_first_name("John").last_name)
7   end
8 end
```

```
1 class Person < Sequel::Model  
2 end
```

```
RakeMate v2.0.0 running on Ruby v1.8.6 (/usr/bin/ruby)
>>> /Users/gmalamid-devel/ruby/whatever_code/synthesis_examples/Rakefile

(in /Users/gmalamid-devel/ruby/whatever_code/synthesis_examples)
Loaded suite /Library/Ruby/Gems/1.8/gems/rake-0.8.1/lib/rake/rake_test_loader
Started
..
Finished in 0.002974 seconds.

2 tests, 3 assertions, 0 failures, 0 errors
```

```
1  #!/usr/bin/env ruby
2  require File.dirname(__FILE__) + "/sql"
3  require File.dirname(__FILE__) + "/sql/database"
4
5  Person.create(:first_name => 'John', :last_name => 'Doe')
6
7  p Greeter.say_hello("John")
```

```
RubyMate r8136 running Ruby r1.8.6  
(/System/Library/Frameworks/Ruby.framework/Versions/1.8/usr/bin  
/ruby)  
>>> main.rb  
  
"Hello, John Doe"
```

```
1 class CreatePeople < Sequel::Migration
2   def up
3     create_table :people do
4       primary_key :id
5       varchar :first_name
6       varchar :last_name
7     end
8   end
9
10  def down
11    drop_table :people
12  end
13 end
14
15 DB = Sequel.sqlite
16 CreatePeople.apply(DB, :up)
```

```
1 class CreatePeople < Sequel::Migration
2   def up
3     create_table :people do
4       primary_key :id
5       varchar :given_name
6       varchar :last_name
7     end
8   end
9
10  def down
11    drop_table :people
12  end
13 end
14
15 DB = Sequel.sqlite
16 CreatePeople.apply(DB, :up)
```

```
1 require File.dirname(__FILE__) + "/helper"
2
3 class PersonTest < Test::Unit::TestCase
4   def test_find_by_name
5     Person.create(:given_name => "John", :last_name => "Doe")
6     assert_equal("Doe", Person.find_by_given_name("John").last_name)
7   end
8 end
```

```
RakeMate v2.0.0 running on Ruby v1.8.6 (/usr/bin/ruby)
>>> /Users/gmalamid-devel/ruby/whatever_code/synthesis_examples/Rakefile

(in /Users/gmalamid-devel/ruby/whatever_code/synthesis_examples)
Loaded suite /Library/Ruby/Gems/1.8/gems/rake-0.8.1/lib/rake/rake_test_loader
Started
..
Finished in 0.003556 seconds.

2 tests, 3 assertions, 0 failures, 0 errors
```

`Sequel::Error::InvalidStatement: INSERT INTO people (first_name, last_name) VALUES ('John', 'Doe')` table people has no column named first_name

```
method execute_insert in sqlite.rb at line 50
method insert          in sqlite.rb at line 171
method save!           in record.rb at line 238
method save            in validations.rb at line 12
method create          in record.rb at line 133
method transaction     in sqlite.rb at line 124
method transaction     in database.rb at line 594
method transaction     in sqlite.rb at line 124
method hold            in connection_pool.rb at line 61
method transaction     in sqlite.rb at line 118
method create          in record.rb at line 131
at top level           in main.rb at line 5
```

```
[Synthesis] Collecting expectations...
Loaded suite /usr/bin/rake
Started
..
Finished in 0.008957 seconds.

2 tests, 3 assertions, 0 failures, 0 errors
[Synthesis] Verifying expectation invocations...
Loaded suite /usr/bin/rake
Started
..
Finished in 0.00263 seconds.

2 tests, 3 assertions, 0 failures, 0 errors
[Synthesis]
[Synthesis] Tested Expectations:
[Synthesis]
[Synthesis] Untested Expectations:
[Synthesis] Person.find_by_first_name(String) in ./sql/test/greeter_test.rb:6:in `test_says_hello'
[Synthesis]
[Synthesis] Ignoring:
[Synthesis]
[Synthesis] FAILED.
```

```
1 require File.dirname(__FILE__) + "/helper"
2
3 class GreeterTest < Test::Unit::TestCase
4   def test_says_hello
5     person = stub(:given_name => 'John', :last_name => 'Doe')
6     Person.expects(:find_by_given_name).with("John").returns(person)
7     assert_equal("Hello, John Doe", Greeter.say_hello("John"))
8   end
9 end
```

```
1 class Greeter
2   def self.say_hello(given_name)
3     person = Person.find_by_given_name(given_name)
4     "Hello, #{person.given_name} #{person.last_name}"
5   end
6 end
```

```
RakeMate v2.0.0 running on Ruby v1.8.6 (/usr/bin/ruby)
>>> /Users/gmalamid-devel/ruby/whatever_code/synthesis_examples/Rakefile

(in /Users/gmalamid-devel/ruby/whatever_code/synthesis_examples)
[Synthesis] Collecting expectations...
Loaded suite /usr/bin/rake
Started
..
Finished in 0.004433 seconds.

2 tests, 3 assertions, 0 failures, 0 errors
[Synthesis] Verifying expectation invocations...
Loaded suite /usr/bin/rake
Started
..
Finished in 0.00482 seconds.

2 tests, 3 assertions, 0 failures, 0 errors
[Synthesis]
[Synthesis] Verified 1 expectations
[Synthesis] SUCCESS.
```

```
RubyMate r8136 running Ruby r1.8.6  
(/System/Library/Frameworks/Ruby.framework/Versions/1.8/usr/bin  
/ruby)  
>>> main.rb  
  
"Hello, John Doe"
```

Synthesis

- <http://synthesis.rubyforge.org/>
- stuart.caborn@thoughtworks.com
- george.malamidis@thoughtworks.com