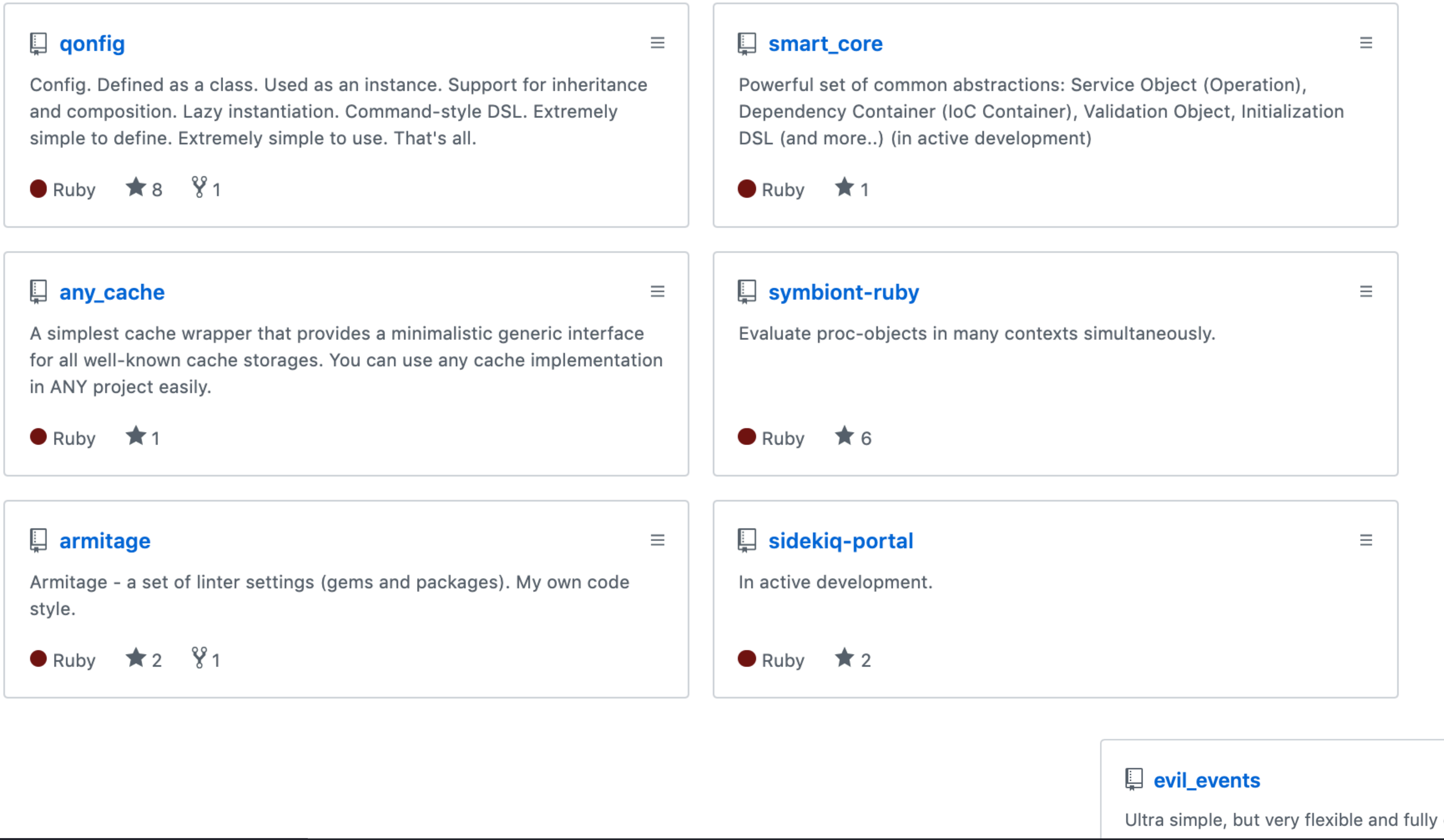
SmartCore

https://github.com/0exp/smart_core

Абстракции, которые вам понравятся :)



qonfig

Config. Defined as a class. Used as an instance. Support for inheritance and composition. Lazy instantiation. Command-style DSL. Extremely simple to define. Extremely simple to use. That's all.

● Ruby ★ 8 1

smart_core

Powerful set of common abstractions: Service Object (Operation), Dependency Container (IoC Container), Validation Object, Initialization DSL (and more..) (in active development)

● Ruby ★ 1

any_cache

A simplest cache wrapper that provides a minimalistic generic interface for all well-known cache storages. You can use any cache implementation in ANY project easily.

● Ruby ★ 1

symbiont-ruby

Evaluate proc-objects in many contexts simultaneously.

● Ruby ★ 6

armitage

Armitage - a set of linter settings (gems and packages). My own code style.

● Ruby ★ 2 1

sidekiq-portal

In active development.

● Ruby ★ 2

evil_events

Ultra simple, but very flexible and fully

https://github.com/0exp
https://github.com/0exp/smart_core

Для кого?

SmartCore::Validator

прикольно валидируем

SmartCore::Initializer

круто инстанцируем

SmartCore::Operation

клево исполняем

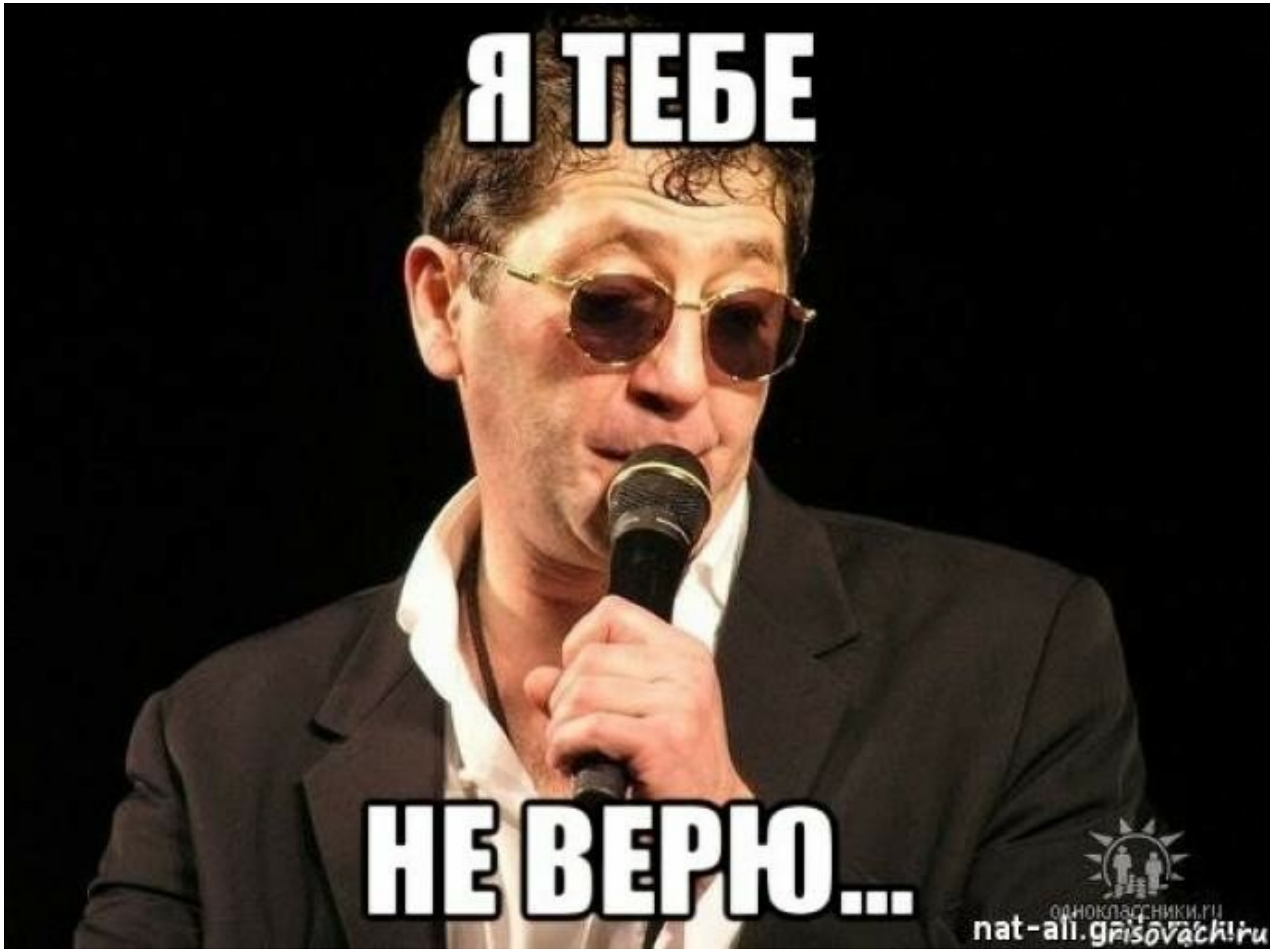
SmartCore::Container

а это зачем?

SmartCore::Injector

exclusive, перелогинься

SmartCore::Validator



SmartCore::Validator

Синтаксис, схожий с ActiveModel::Validations

Nested Validations

Нормальная композиция валидаторов

Ошибка - это **error code** (но уже хочу чуток по другому)

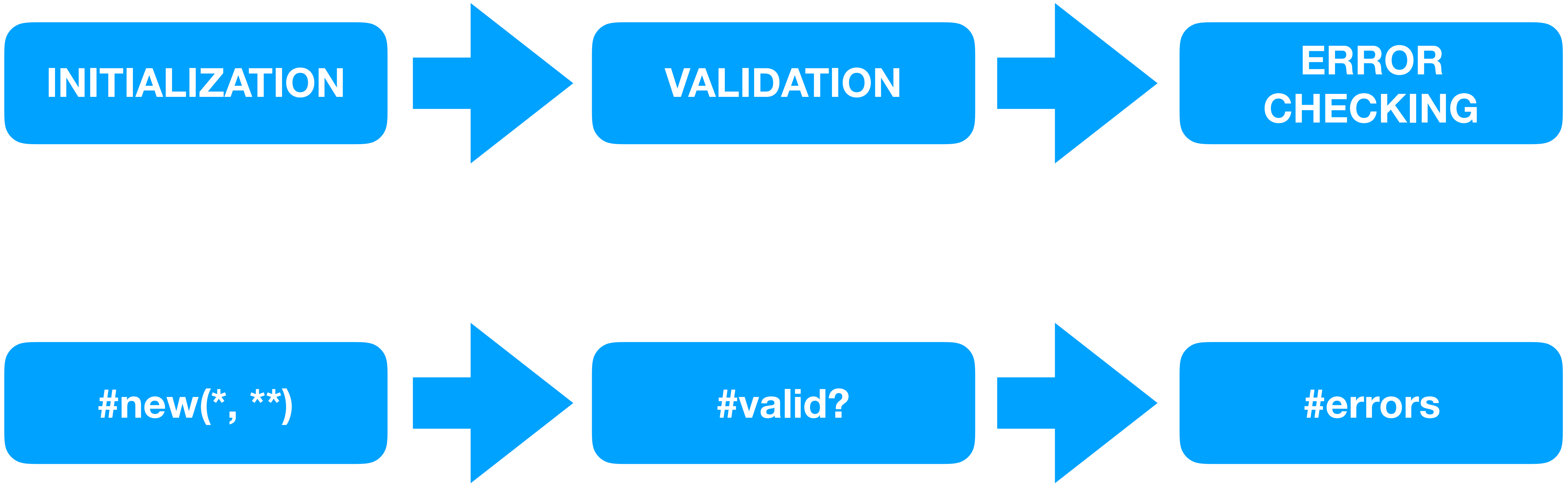
Предсказуемый source code

SmartCore::Validator

- ✓ - **nested validations;**
- ✓ - validator - as an instance;
- ✓ - validation - as a method;
- ✓ - validator composition;
- ✓ - attribute definition DSL
- ✓ - **errors - collection of error codes**;
- ✓ - simple and concise API;
- ✓ - no dependencies;
- ✓ - **maintainable source code**;

- ✓ - [DSL] **.attribute**
- ✓ - [DSL] **.validate**
- ✓ - [DSL] **.validate_with(Validator)**
- ✓ - **#error(:error_code)**
- ✓ - **#fatal(:error_code)**
- ✓ - **#valid?**
- ✓ - **#errors**
- ✓ - **attribute readers**

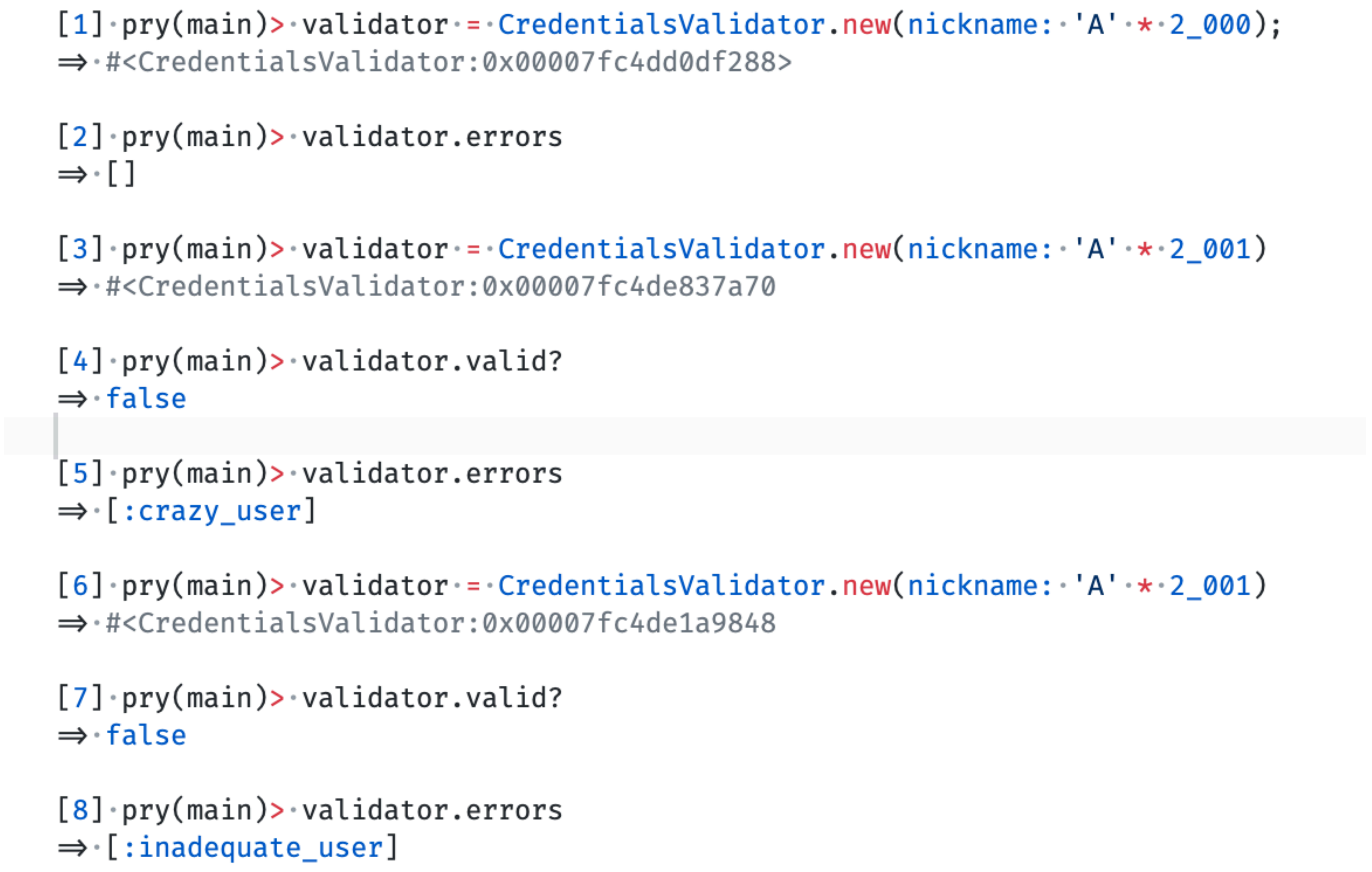
SmartCore::Validator - Execution Flow



SmartCore::Validator - Definition

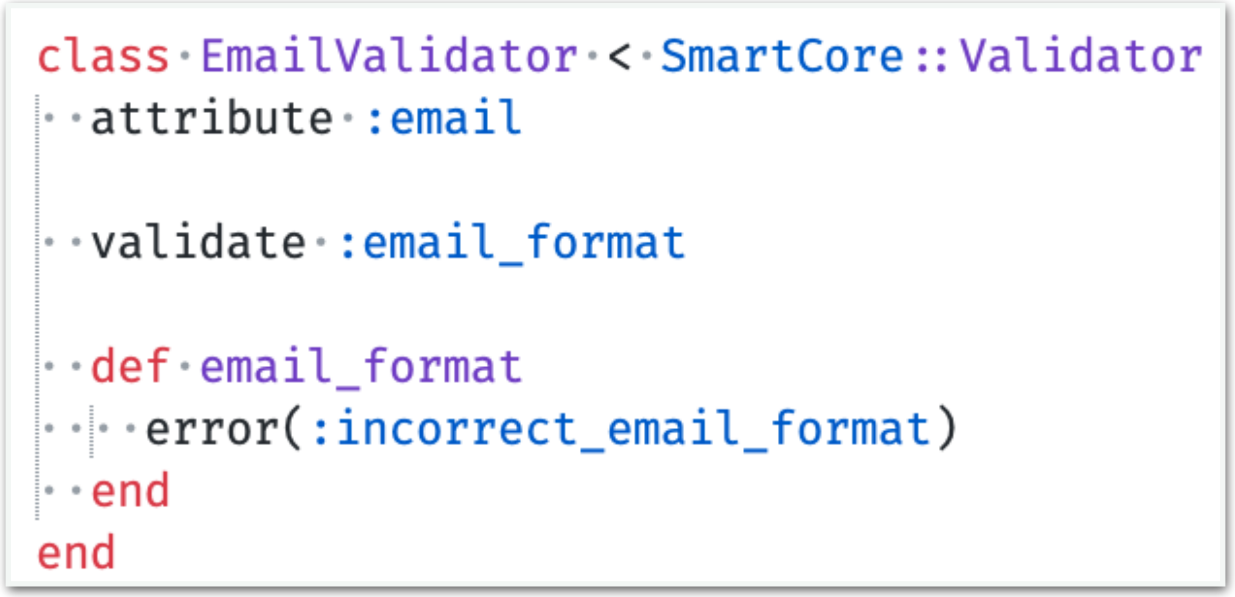
```ruby
class CredentialsValidator < SmartCore::Validator
  attribute :nickname

  validate :nickname_correctness

  validate :adequacy do
    validate :psychopathy
  end

  private

  def nickname_correctness
    error(:incorrect_nickname) unless nickname.is_a?(String)
  end

  def adequacy
    error(:inadequate_user) if [true, false].sample
  end

  def psychopaty
    error(:crazy_user) if nickname.size > 2_000
  end
end
```

SmartCore::Validator - Usage

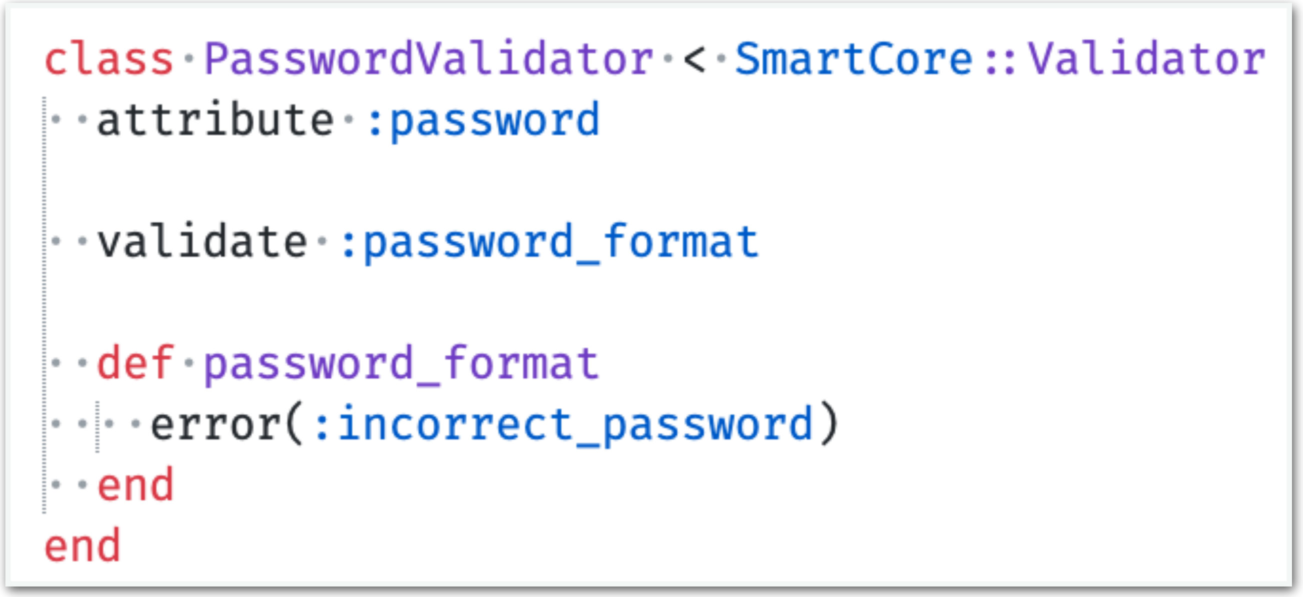


```
[1] pry(main)> validator = CredentialsValidator.new(nickname: 'A' * 2_000);
=> #<CredentialsValidator:0x00007fc4dd0df288>

[2] pry(main)> validator.errors
=> []

[3] pry(main)> validator = CredentialsValidator.new(nickname: 'A' * 2_001)
=> #<CredentialsValidator:0x00007fc4de837a70

[4] pry(main)> validator.valid?
=> false

[5] pry(main)> validator.errors
=> [:crazy_user]

[6] pry(main)> validator = CredentialsValidator.new(nickname: 'A' * 2_001)
=> #<CredentialsValidator:0x00007fc4de1a9848

[7] pry(main)> validator.valid?
=> false

[8] pry(main)> validator.errors
=> [:inadequate_user]
```

SmartCore::Validator - Composition

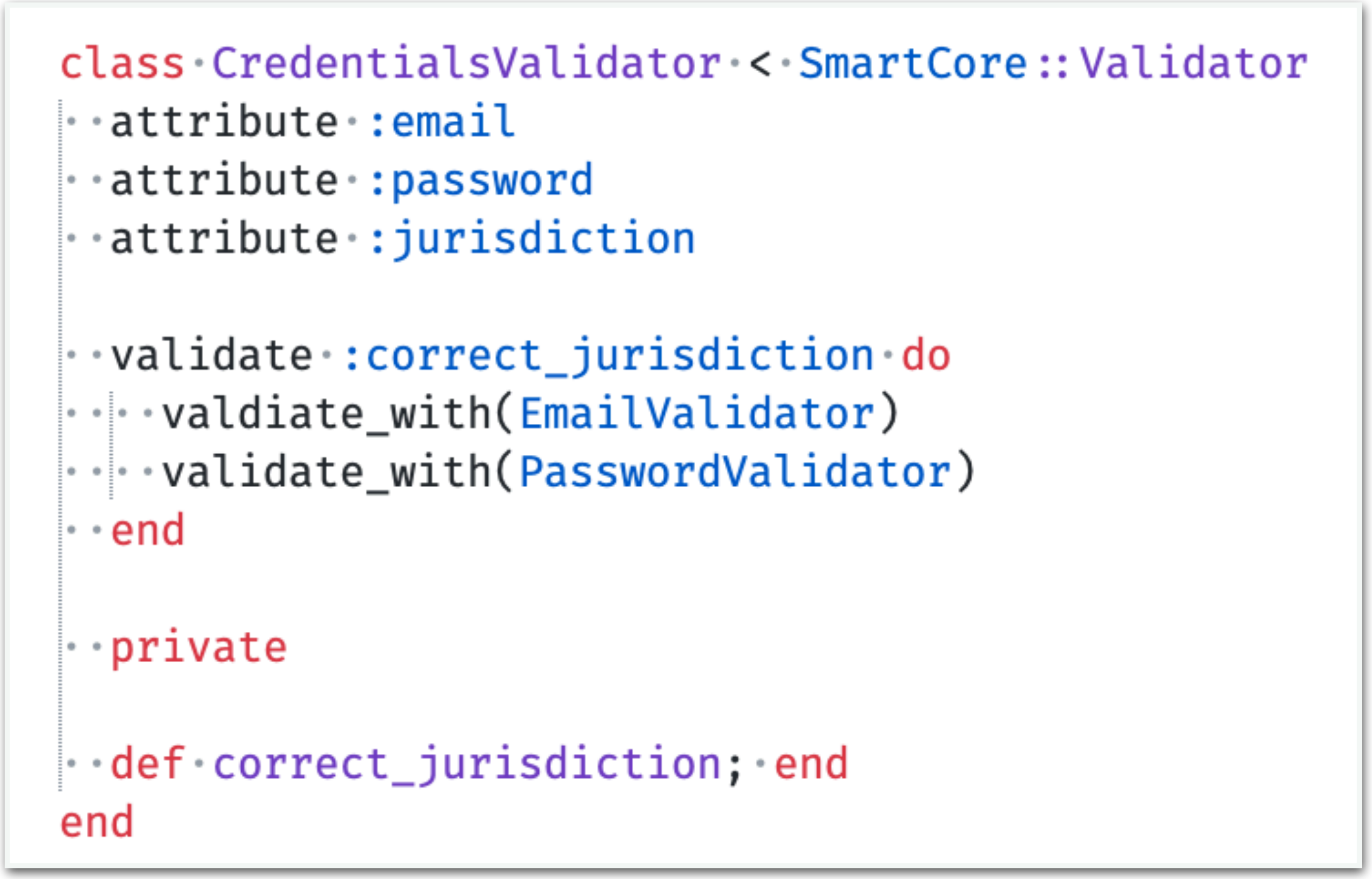


```ruby
class EmailValidator < SmartCore::Validator
  attribute :email

  validate :email_format

  def email_format
    error(:incorrect_email_format)
  end
end
```



```ruby
class PasswordValidator < SmartCore::Validator
  attribute :password

  validate :password_format

  def password_format
    error(:incorrect_password)
  end
end
```

COMPOSE THEM ALL:



```ruby
class CredentialsValidator < SmartCore::Validator
  attribute :email
  attribute :password
  attribute :jurisdiction

  validate :correct_jurisdiction do
    valdiate_with(EmailValidator)
    validate_with(PasswordValidator)
  end

  private

  def correct_jurisdiction; end
end
```

SmartCore::Initializer

SmartCore::Initializer

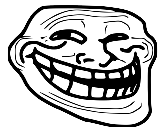
Простой DSL

Дай DSL для параметра, дай DSL для kwarg

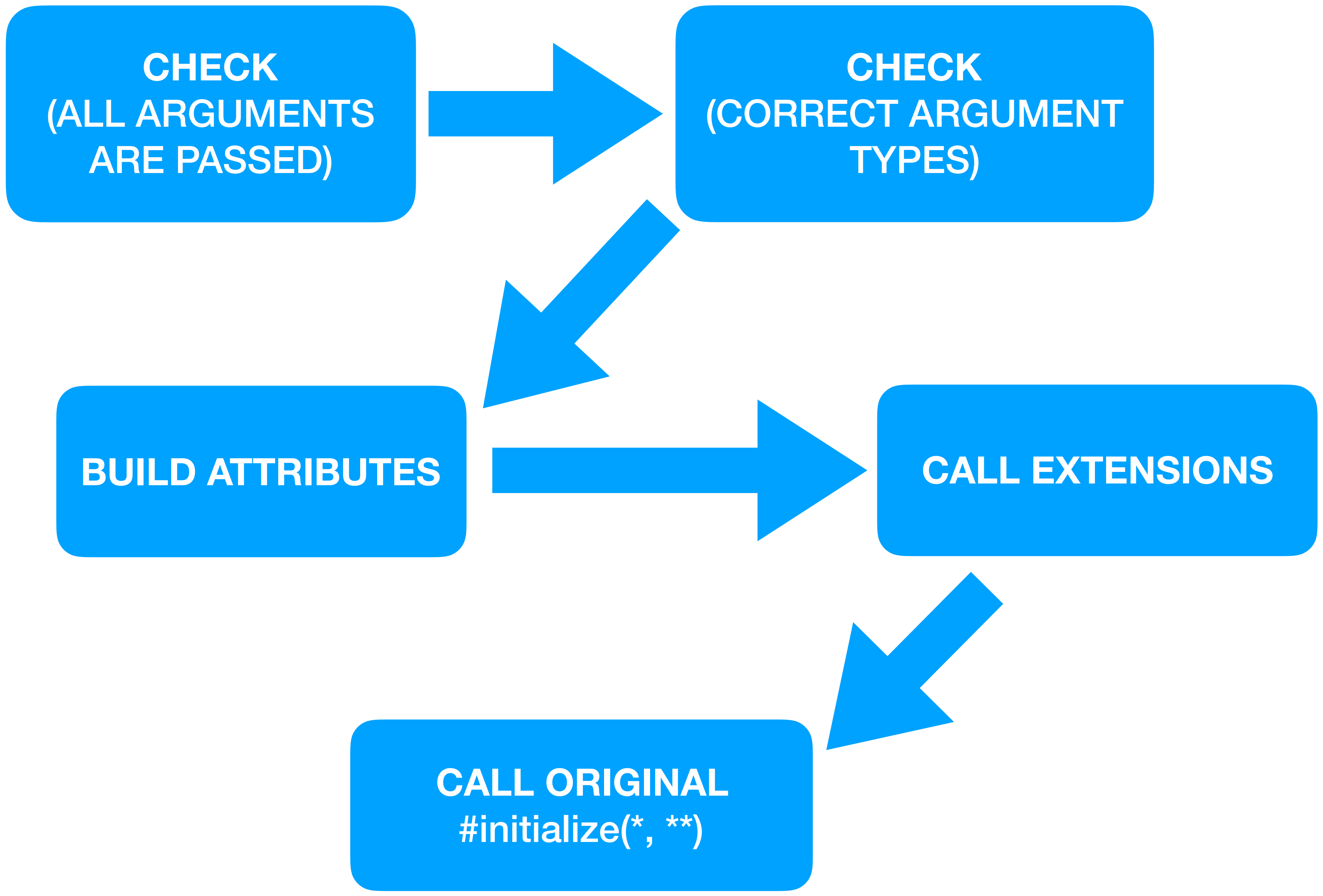
Проверка типов

Всякие плюшечки, когда придумаю
(захотел :default заиметь, например, или ошибки нормальные)

SmartCore::Initializer

- ✓ - mix and use**;**
- ✓ - attribute definition DSL (**.param**, **.params**, **.option**, **.options**);
- ✓ - ivars (under the hood);
- ✓ - **type annotations**;
- ✓ - positional attributes (**.param**, **.params**);
- ✓ - keyword attributes (**.option**, **.options**);
- ✓ - attribute visibility (**:privacy => ...**);
- ✓ - default values (**:default => ...**);
- ✓ - typed / semantic exceptions;
- ✓ - thread safe;
- ✓ - **drop tainbox;** 
- ✓ - **maintainable source code**;

SmartCore::Initializer - Execution Flow

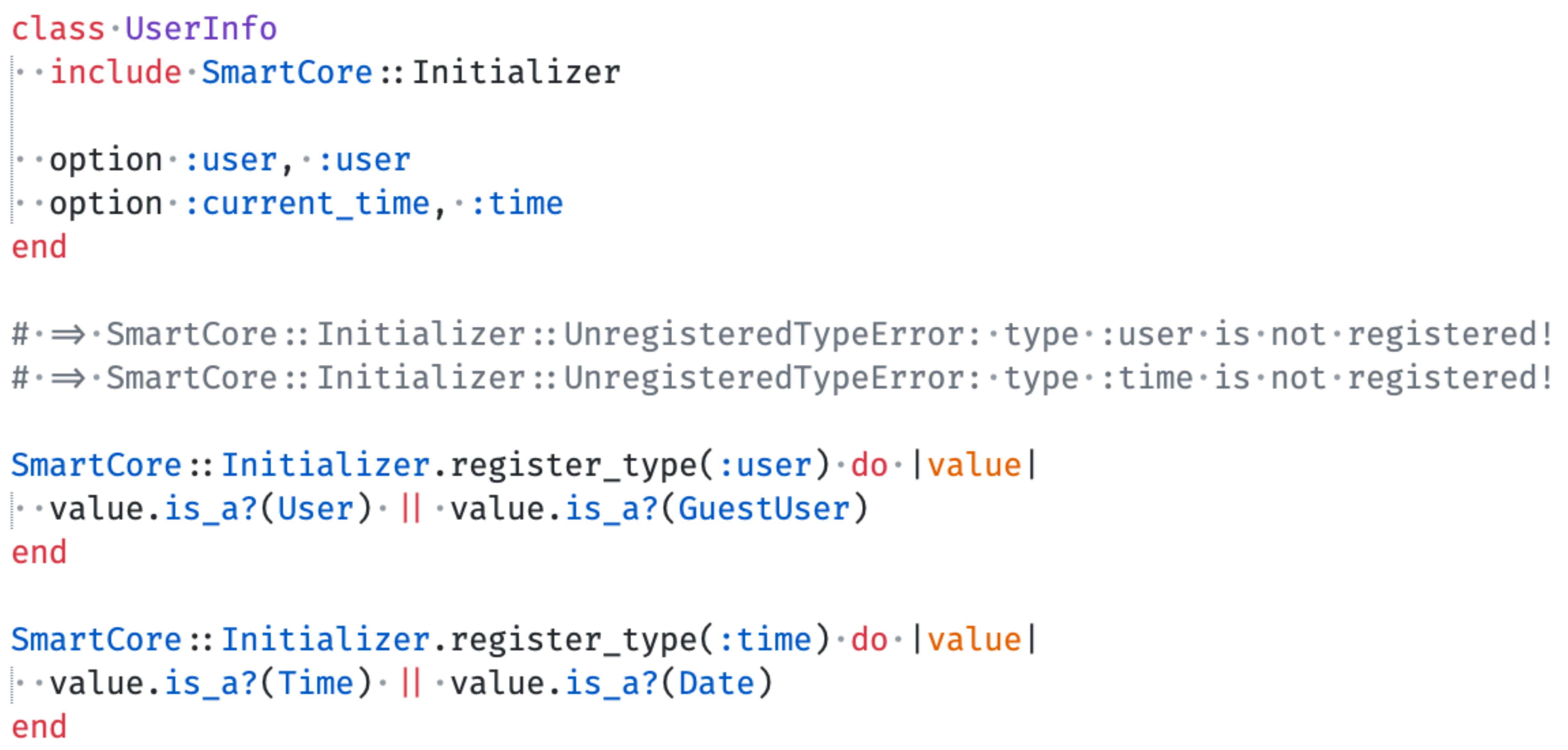

CHECK (ALL ARGUMENTS ARE PASSED)
CHECK (CORRECT ARGUMENT TYPES)
BUILD ATTRIBUTES
CALL EXTENSIONS
CALL ORIGINAL #initialize(*, **)

SmartCore::Initializer - Usage

```
[1] pry(main)> User.new
# => SmartCore::Initializer::ParameterError:
# => Wrong number of parameters (given 0, expected 2)

[2] pry(main)> User.new(1, 2)
# => SmartCore::Initializer::ArgumentError:
# => Incorrect type of <nickname> attribute (given: Integer, expected: :string)

[3] pry(main)> User.new('exclusive', 5, admin: 'test')
# => SmartCore::Initializer::ArgumentError:
# => Incorrect type of <admin> attribute (given: String, expected: :boolean)

[4] pry(main)> user = User.new('exclusive', 5, admin: true)
# => #<User:0x0 @admin=true, @age=5, @nickname="exclusive", @time=2019-06-02 23:01:14 +0300>

[5] pry(main)> user.nickname
# => "exclusive"

[6] pry(main)> user.age
# => NoMethodError: private method `age' called for #<

[7] pry(main)> user.options
# => {:admin=>true, :time=>2019-06-02 23:01:14 +0300}

[8] pry(main)> user.params
# => {:nickname=>"exclusive", :age=>5}

[9] pry(main)> user.attributes
# => {:nickname=>"exclusive", :age=>5, :admin=>true, :time=>2019-06-02 23:01:14 +0300}
```

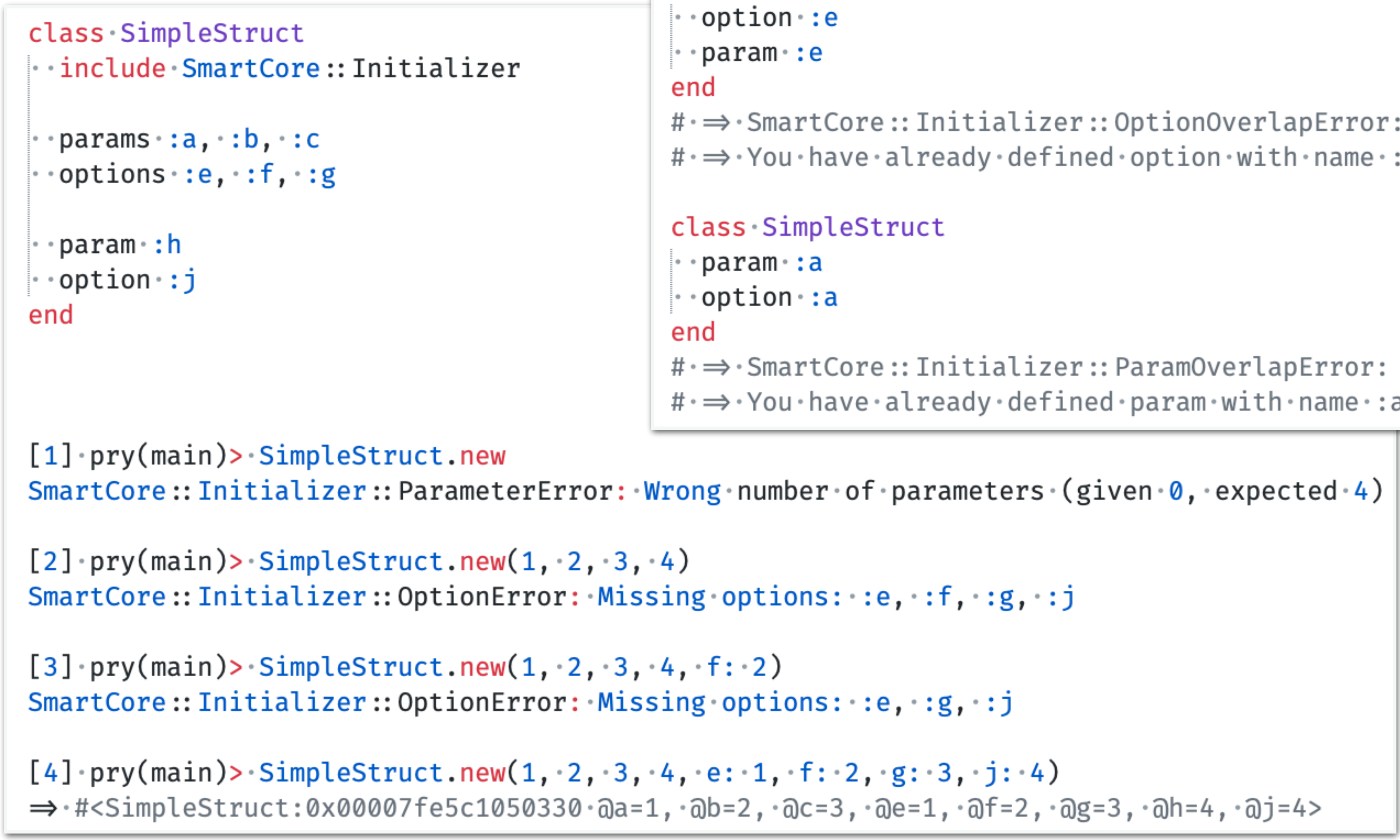
```ruby
class User
  include SmartCore::Initializer

  param :nickname, :string
  param :age, :integer, privacy: :private

  option :admin, :boolean, default: false
  option :time, default: -> { Time.current }
end
```

SmartCore::Initializer - Type Checker API



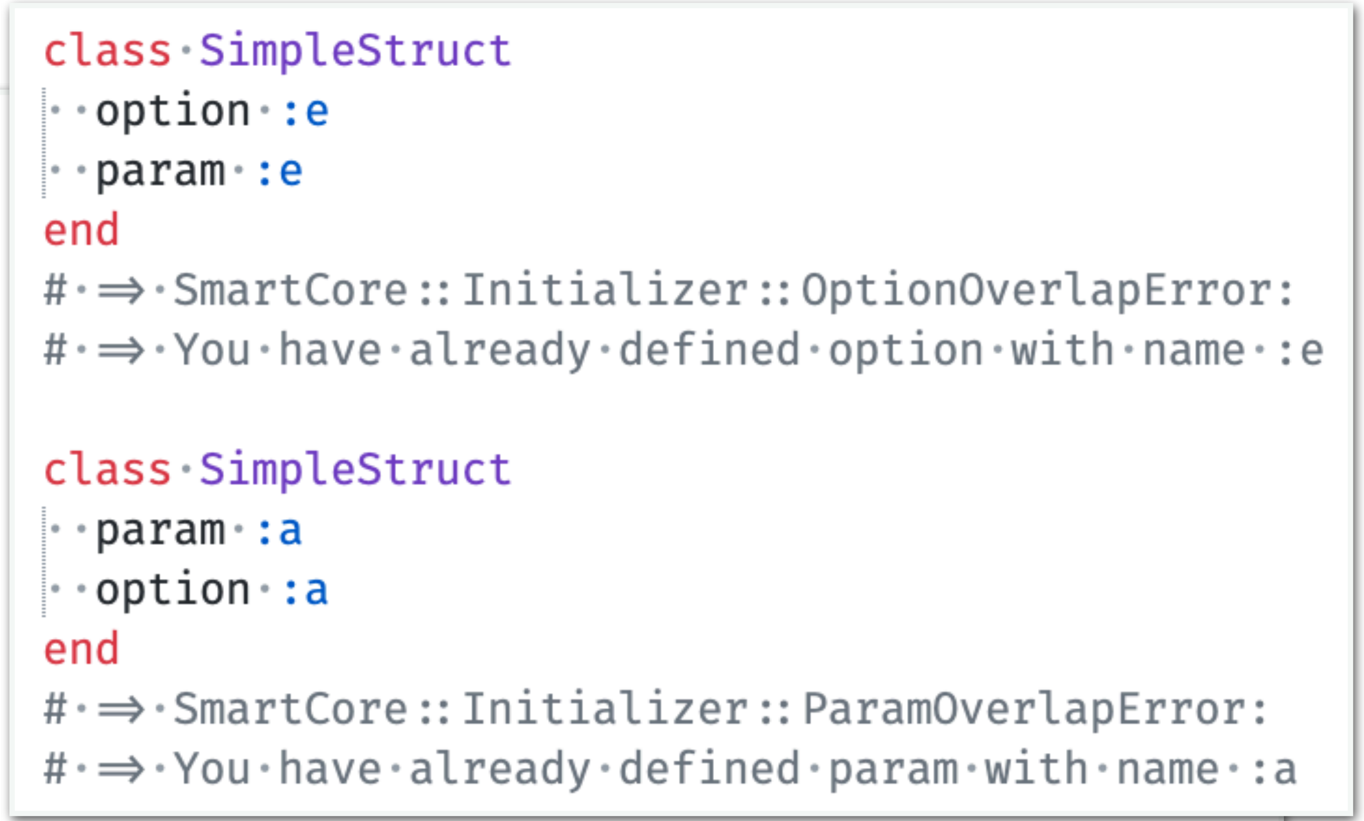
```ruby
class UserInfo
  include SmartCore::Initializer

  option :user, :user
  option :current_time, :time
end

# => SmartCore::Initializer::UnregisteredTypeError: type :user is not registered!
# => SmartCore::Initializer::UnregisteredTypeError: type :time is not registered!

SmartCore::Initializer.register_type(:user) do |value|
  value.is_a?(User) || value.is_a?(GuestUser)
end

SmartCore::Initializer.register_type(:time) do |value|
  value.is_a?(Time) || value.is_a?(Date)
end
```

SmartCore::Initializer - Semantic Exceptions



```ruby
class SimpleStruct
  include SmartCore::Initializer

  params :a, :b, :c
  options :e, :f, :g

  param :h
  option :j
end
```

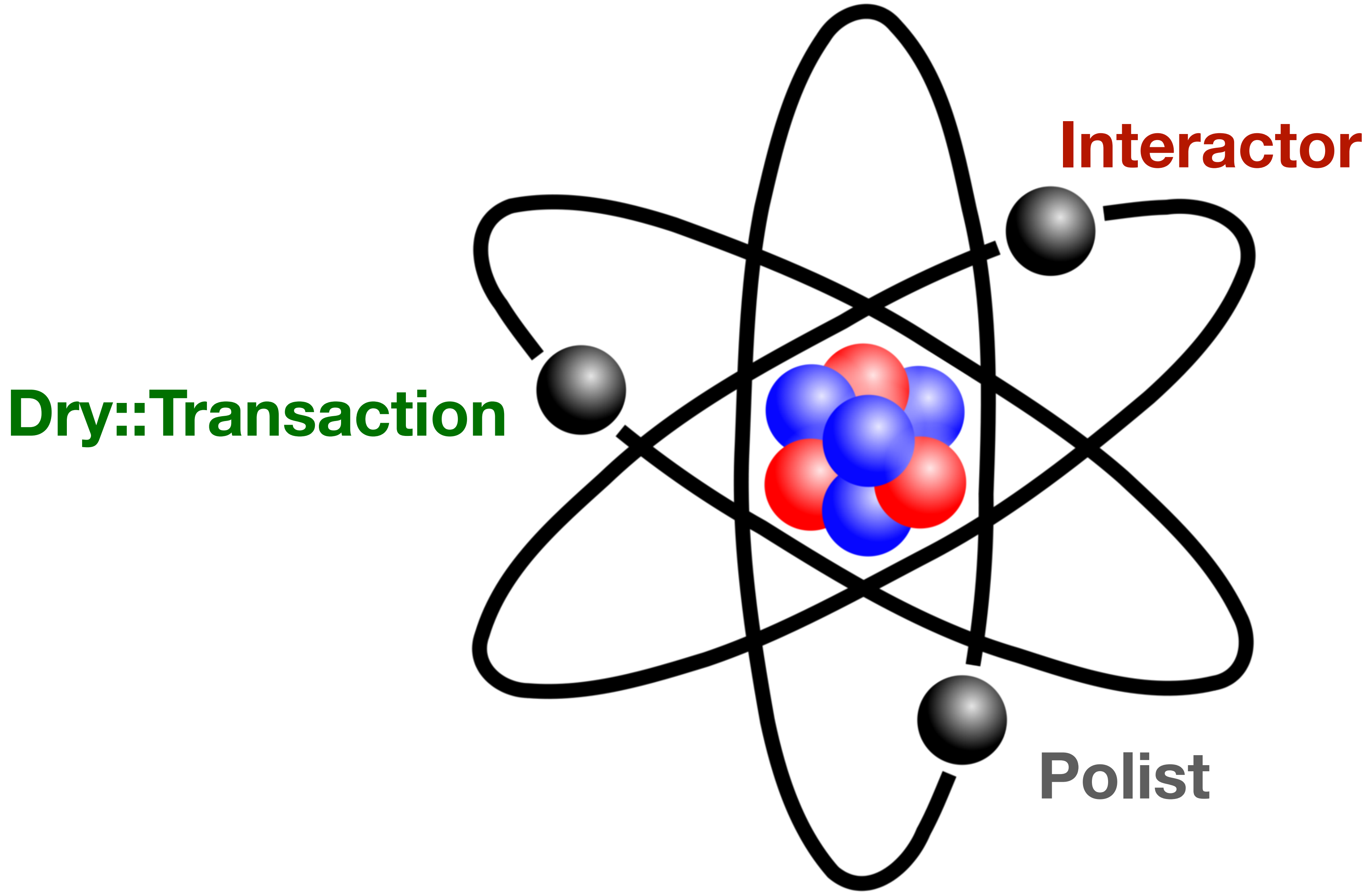
```
[1] pry(main)> SimpleStruct.new
SmartCore::Initializer::ParameterError: Wrong number of parameters (given 0, expected 4)

[2] pry(main)> SimpleStruct.new(1, 2, 3, 4)
SmartCore::Initializer::OptionError: Missing options: :e, :f, :g, :j

[3] pry(main)> SimpleStruct.new(1, 2, 3, 4, f: 2)
SmartCore::Initializer::OptionError: Missing options: :e, :g, :j

[4] pry(main)> SimpleStruct.new(1, 2, 3, 4, e: 1, f: 2, g: 3, j: 4)
=> #<SimpleStruct:0x00007fe5c1050330 @a=1, @b=2, @c=3, @e=1, @f=2, @g=3, @h=4, @j=4>
```



```ruby
class SimpleStruct
  option :e
  param :e
end
# => SmartCore::Initializer::OptionOverlapError:
# => You have already defined option with name :e

class SimpleStruct
  param :a
  option :a
end
# => SmartCore::Initializer::ParamOverlapError:
# => You have already defined param with name :a
```

SmartCore::Operation

SmartCore::Operation



SmartCore::Operation

call(*).new(*).call

Вызывать без инстанцирования

Attribute DSL

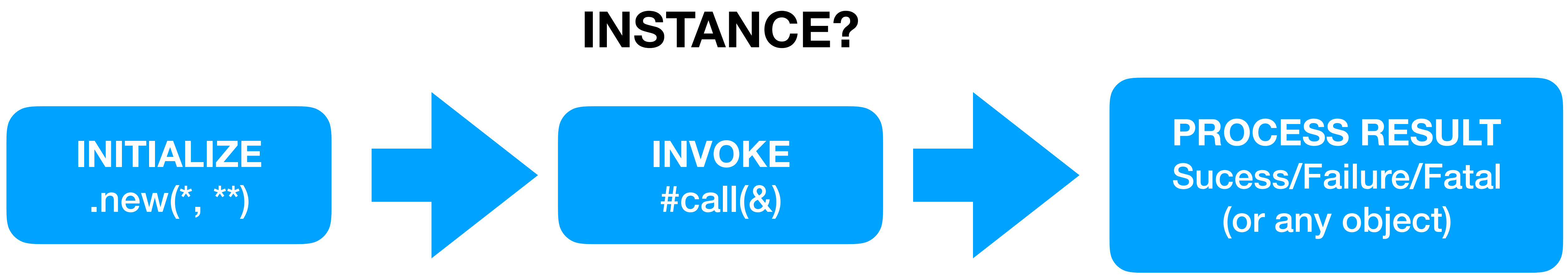
Результат - это объект (или что сам захочешь)

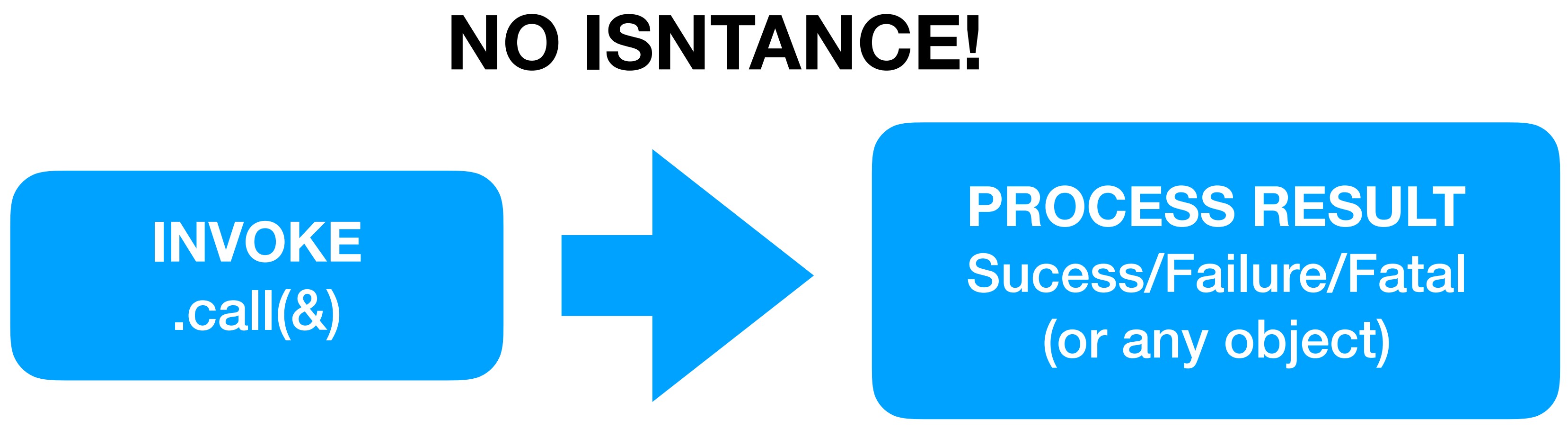
predictable source code

SmartCore::Operation

- ✓ - attribute definition DSL (**SmartCore::Initializer**);
- ✓ **- result as an object;**
  - ✓ **- Success(**data)**, **Failure(*errors), Fatal(*errors)**
  - ✓ **- #success?**
  - ✓ **- #failure?**
  - ✓ **- #fatal?**
- ✓ - yieldable result (**#success?(&)**, **#failure?(&)**, **#fatal?(&)**);
- ✓ - yieldable **#call** (and **.call**);
- ✓ - **CALL.NEW.CALL**;
- ✓ - no external dependencies;
- ✓ - maintainable source code;

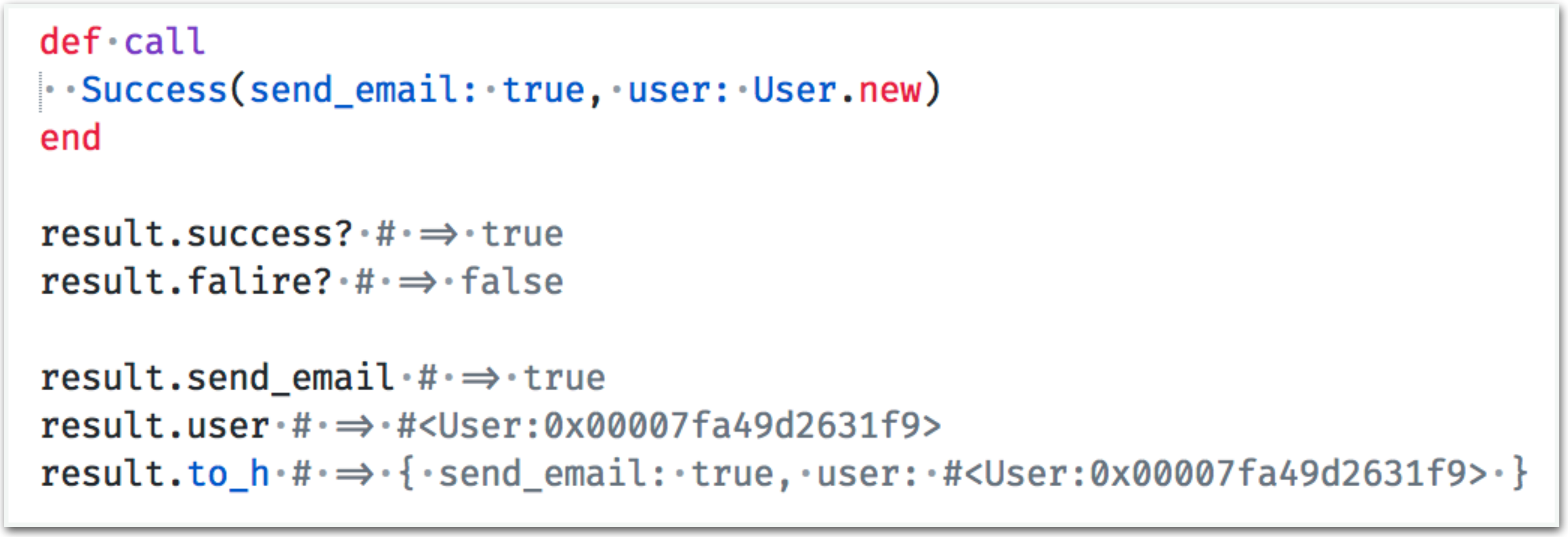
SmartCore::Operation - Execution Flow



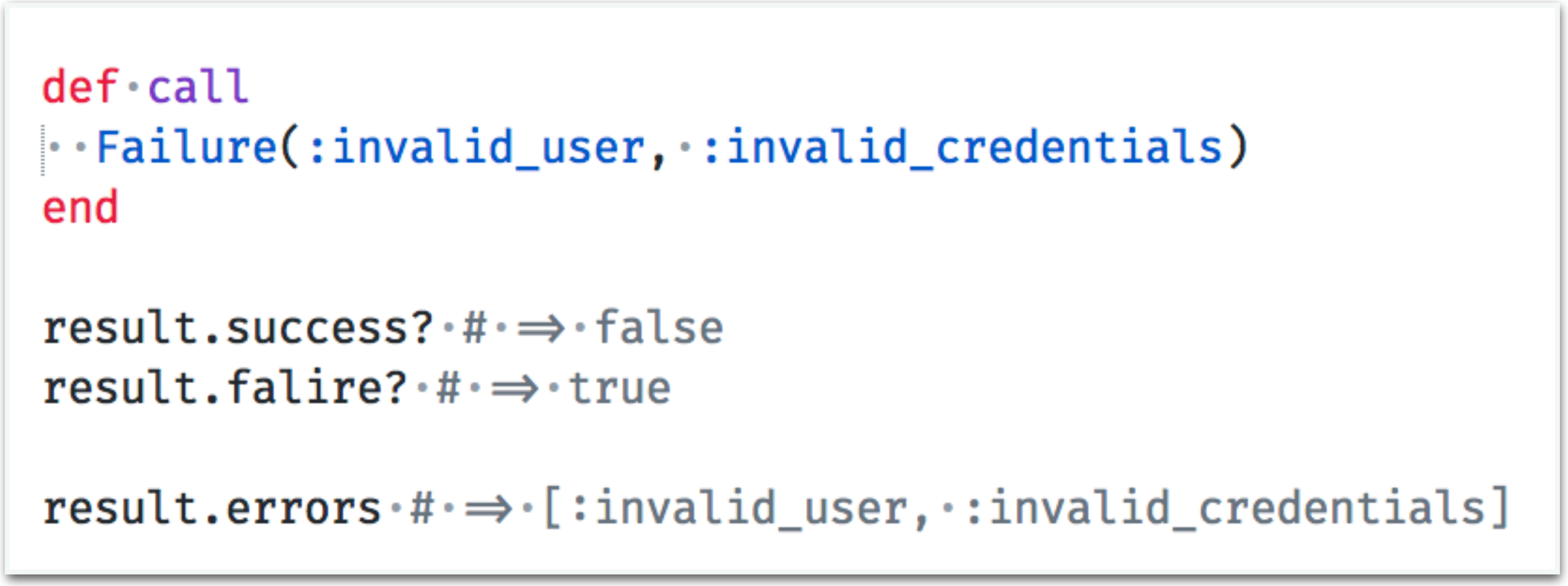


SmartCore::Operation - Result Object (Success / Failure / Fatal)

- #success?
- #failure?
- #fatal?
- #errors
- yield !!!

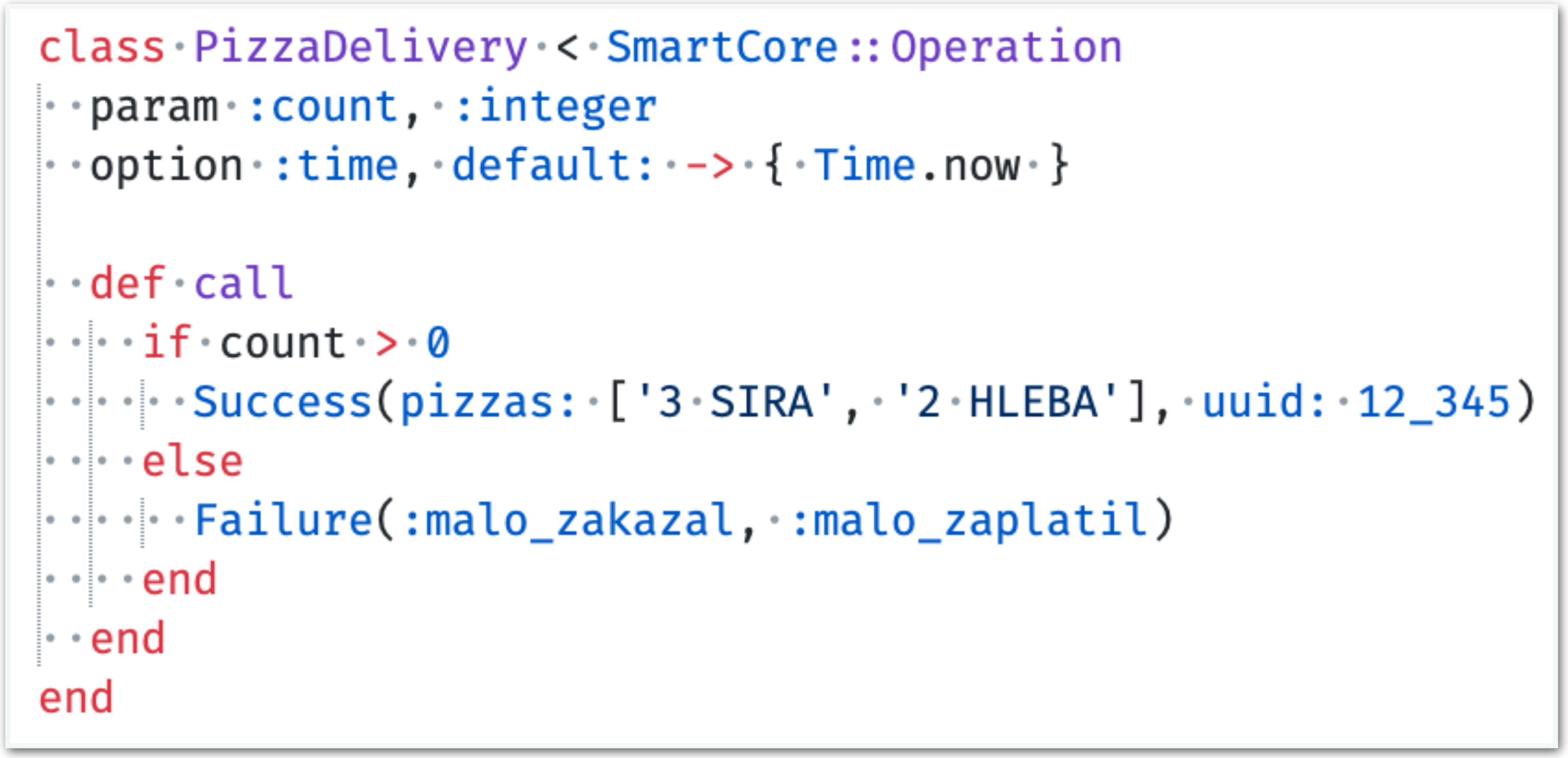


```ruby
def call
  Success(send_email: true, user: User.new)
end

result.success? # => true
result.falire? # => false

result.send_email # => true
result.user # => #<User:0x00007fa49d2631f9>
result.to_h # => { send_email: true, user: #<User:0x00007fa49d2631f9> }
```

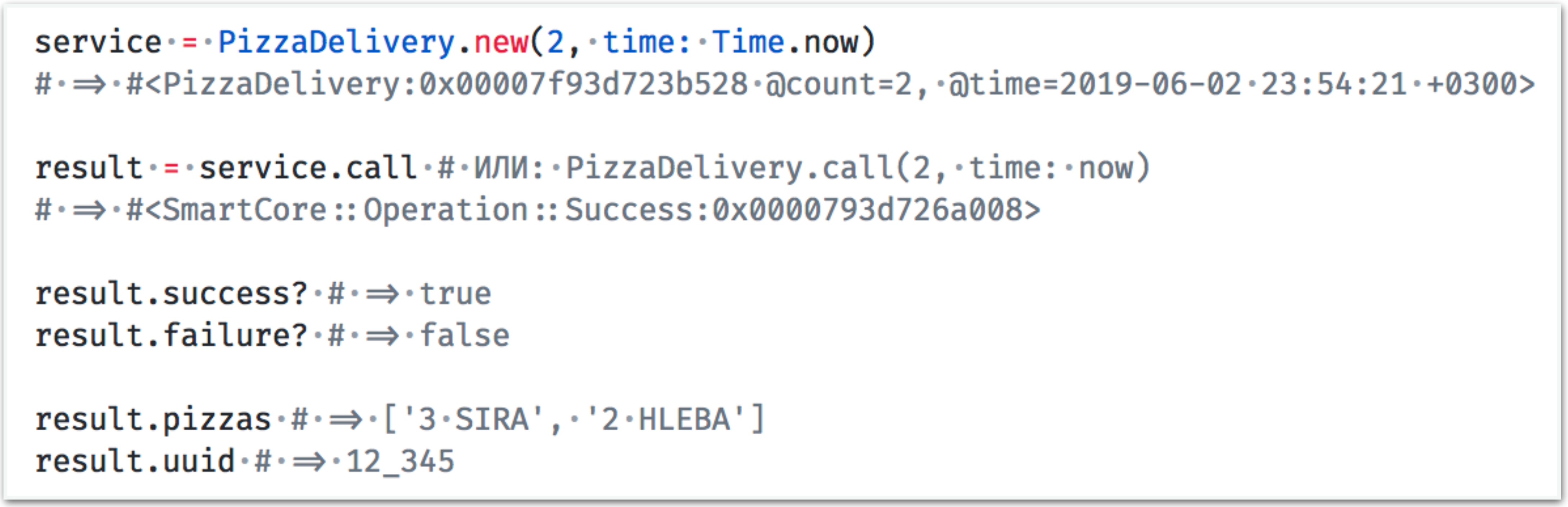


```ruby
def call
  Failure(:invalid_user, :invalid_credentials)
end

result.success? # => false
result.falire? # => true

result.errors # => [:invalid_user, :invalid_credentials]
```

SmartCore::Operation - Basic Usage (Success)

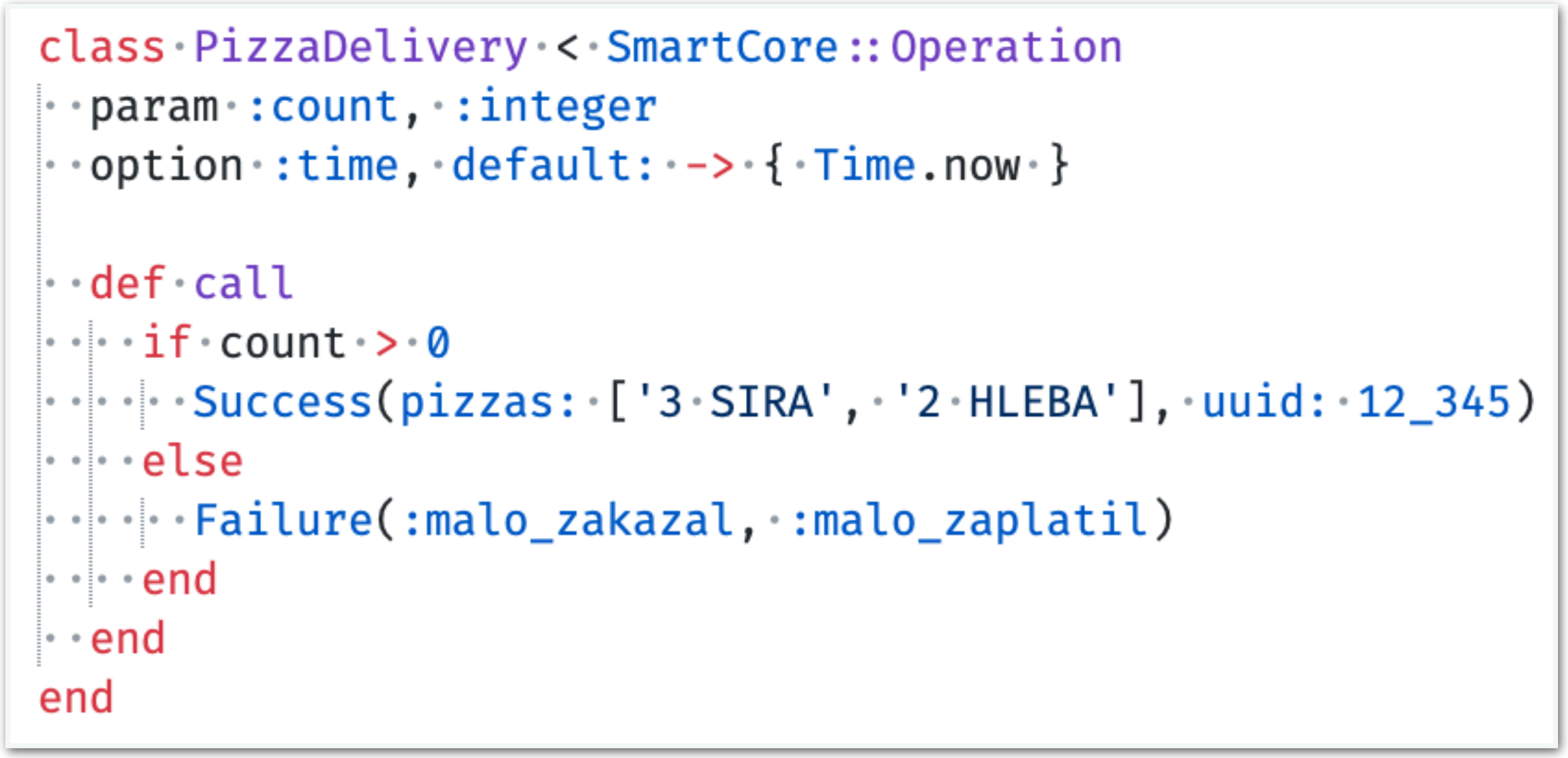


```ruby
class PizzaDelivery < SmartCore::Operation
  param :count, :integer
  option :time, default: -> { Time.now }

  def call
    if count > 0
      Success(pizzas: ['3 SIRA', '2 HLEBA'], uuid: 12_345)
    else
      Failure(:malo_zakazal, :malo_zaplatil)
    end
  end
end
```

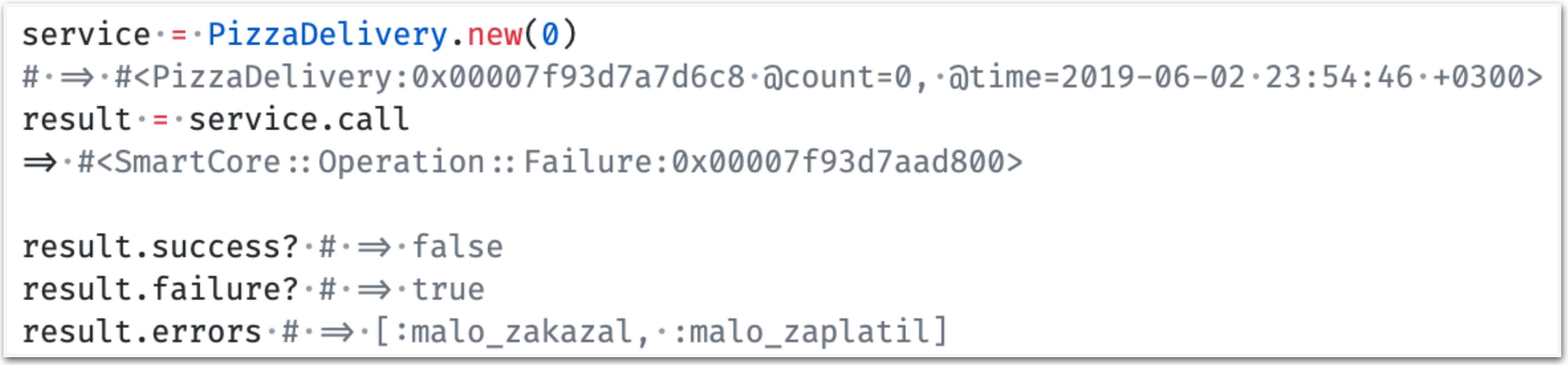


```ruby
service = PizzaDelivery.new(2, time: Time.now)
# => #<PizzaDelivery:0x00007f93d723b528 @count=2, @time=2019-06-02 23:54:21 +0300>

result = service.call # ИЛИ: PizzaDelivery.call(2, time: now)
# => #<SmartCore::Operation::Success:0x0000793d726a008>

result.success? # => true
result.failure? # => false

result.pizzas # => ['3 SIRA', '2 HLEBA']
result.uuid # => 12_345
```

SmartCore::Operation - Basic Usage (Success)



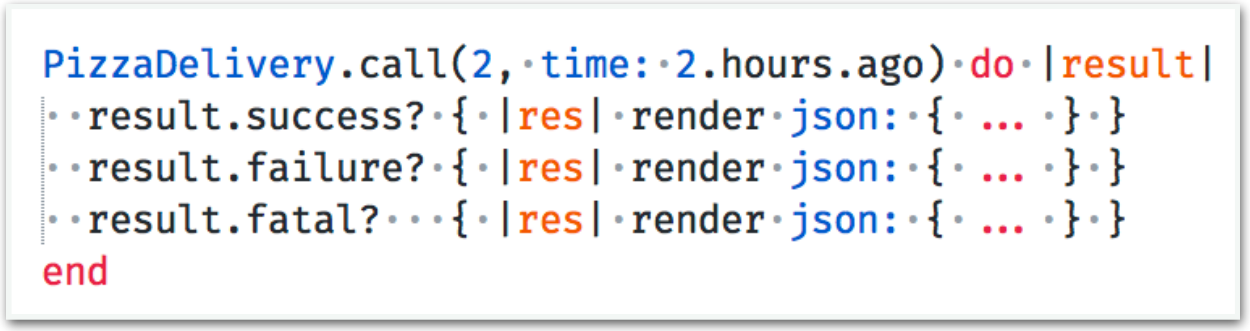
```ruby
class PizzaDelivery < SmartCore::Operation
  param :count, :integer
  option :time, default: -> { Time.now }

  def call
    if count > 0
      Success(pizzas: ['3 SIRA', '2 HLEBA'], uuid: 12_345)
    else
      Failure(:malo_zakazal, :malo_zaplatil)
    end
  end
end
```



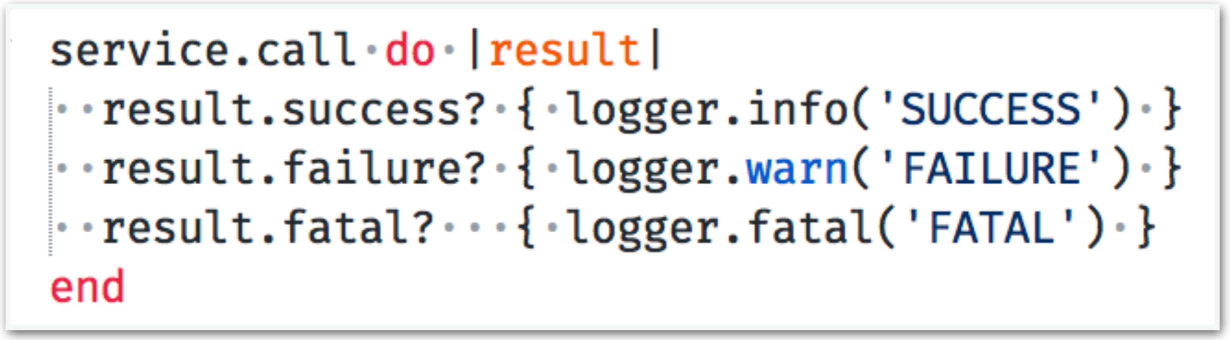
```ruby
service = PizzaDelivery.new(0)
# => #<PizzaDelivery:0x00007f93d7a7d6c8 @count=0, @time=2019-06-02 23:54:46 +0300>
result = service.call
=> #<SmartCore::Operation::Failure:0x00007f93d7aad800>

result.success? # => false
result.failure? # => true
result.errors # => [:malo_zakazal, :malo_zaplatil]
```

SmartCore::Operation - Exclusive API

RESULT MATCHER (yield!)

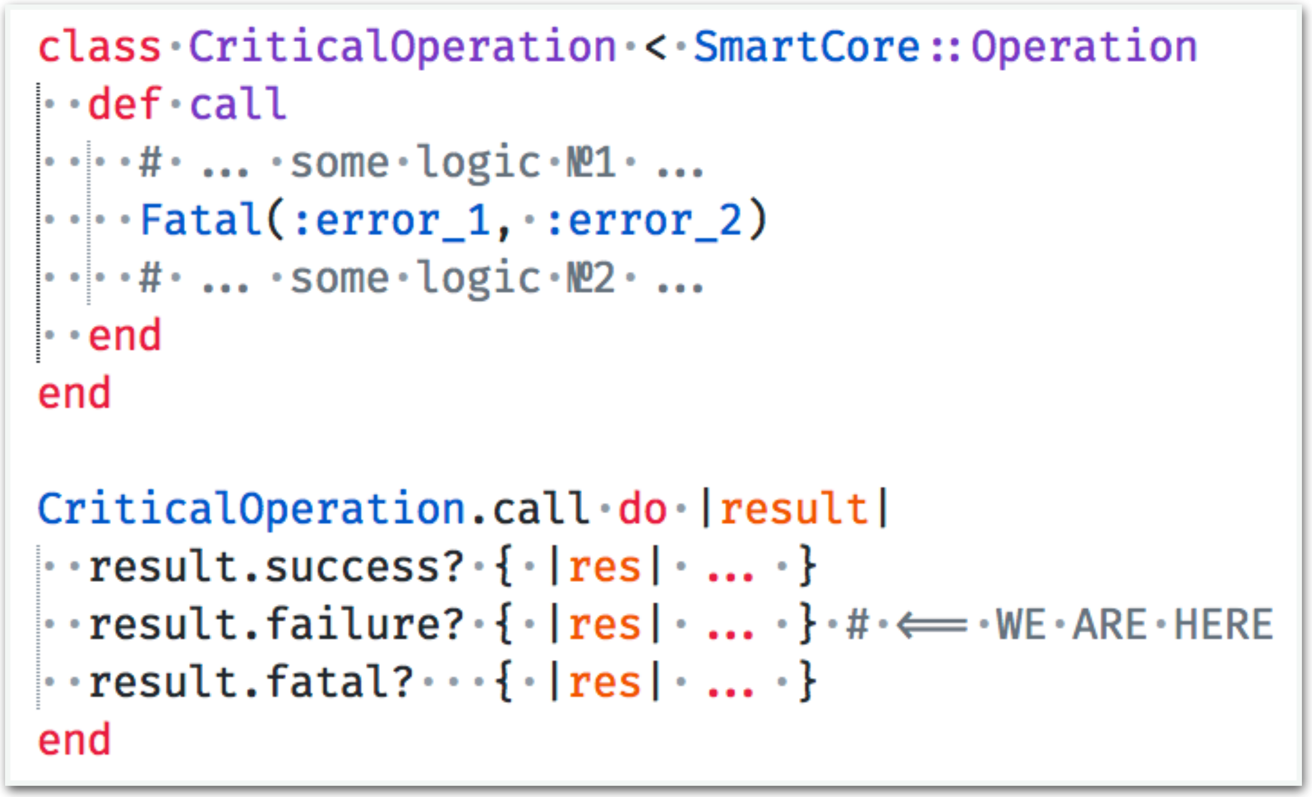


```ruby
PizzaDelivery.call(2, time: 2.hours.ago) do |result|
  result.success? { |res| render json: { ... } }
  result.failure? { |res| render json: { ... } }
  result.fatal?   { |res| render json: { ... } }
end
```



```ruby
service.call do |result|
  result.success? { logger.info('SUCCESS') }
  result.failure? { logger.warn('FAILURE') }
  result.fatal?   { logger.fatal('FATAL') }
end
```

STOP EXECUTION FLOW



```ruby
class CriticalOperation < SmartCore::Operation
  def call
    # ... some logic №1 ...
    Fatal(:error_1, :error_2)
    # ... some logic №2 ...
  end
end

CriticalOperation.call do |result|
  result.success? { |res| ... }
  result.failure? { |res| ... } # <== WE ARE HERE
  result.fatal?   { |res| ... }
end
```

SmartCore::Container

SmartCore::Container

Namespaces

Работает как Instance

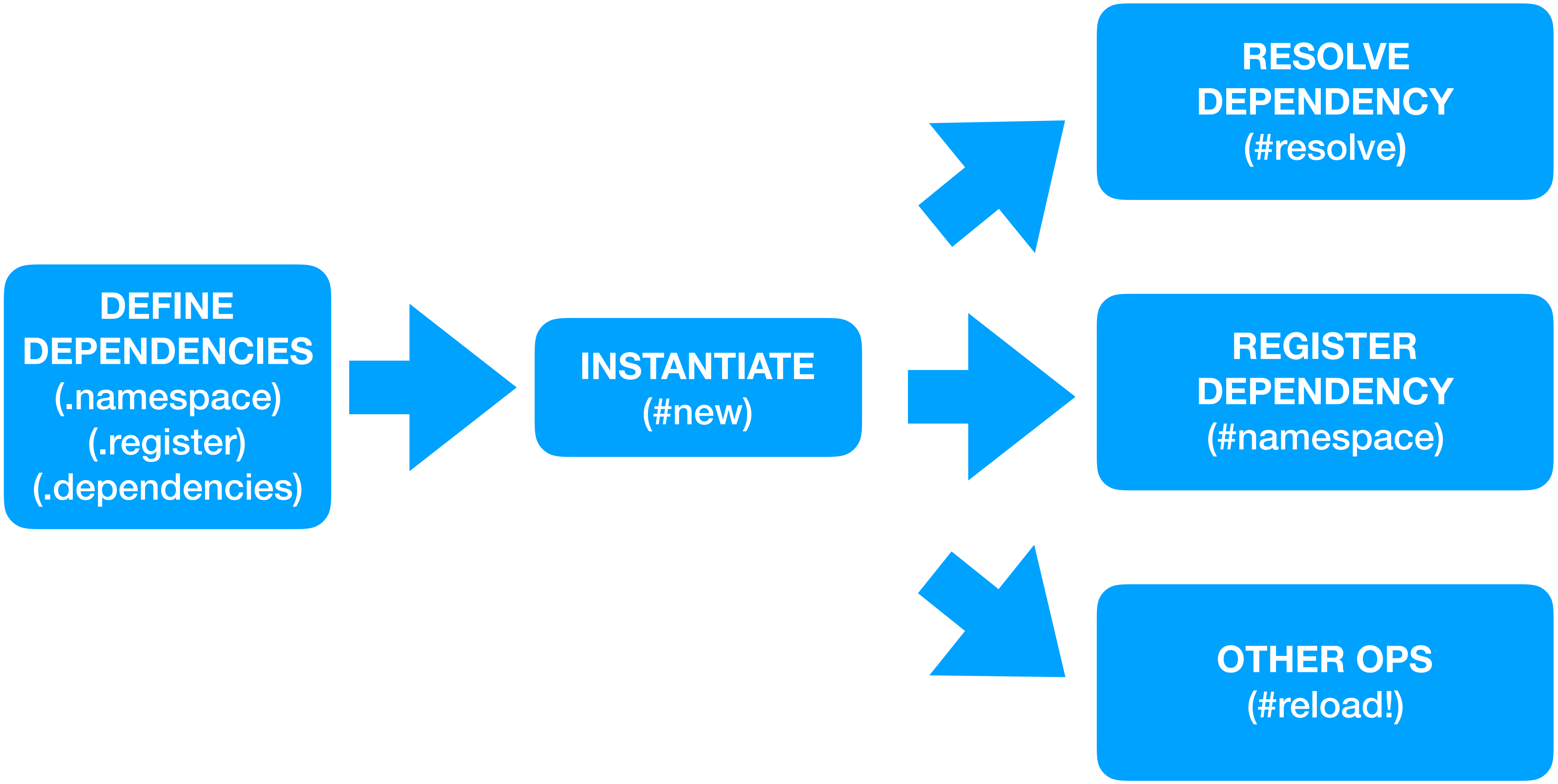
Mixin-реализация

predictable source code

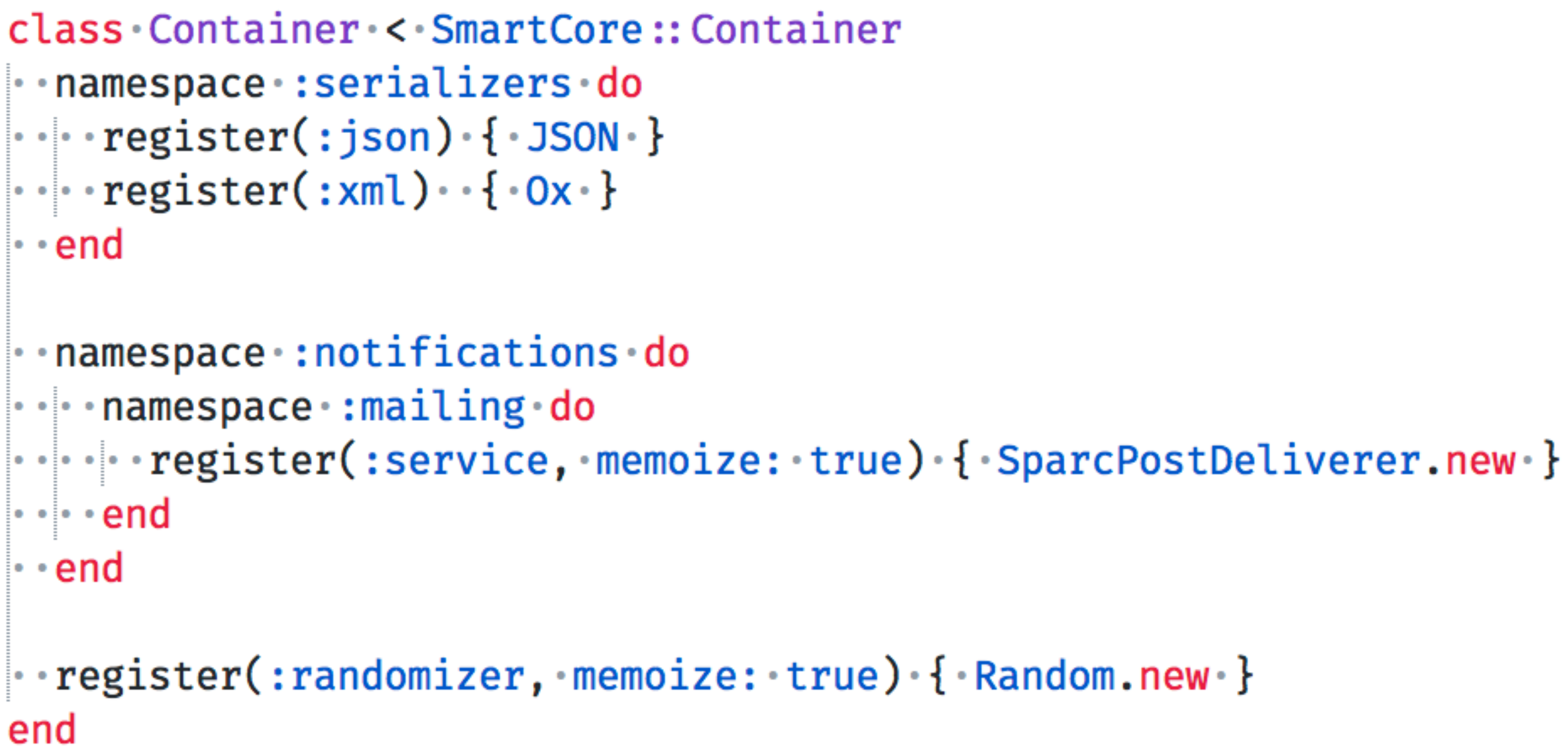
SmartCore::Container

- ✓ - **support for instance behaviour;**
- ✓ - support for mixin behaviour;
- ✓ - memoization;
- ✓ - reloading;
- ✓ - semantic errors;
- ✓ - simple API:
  - ✓ - **.namespace / #namespace**
  - ✓ **- .register / #register**
  - ✓ **- #resolve**
- ✓ - inheritable;
- ✓ - thread-safe;
- ✓ - no dependencies;
- ✓ - maintainable source code;

SmartCore::Container - Execution/Work Flow

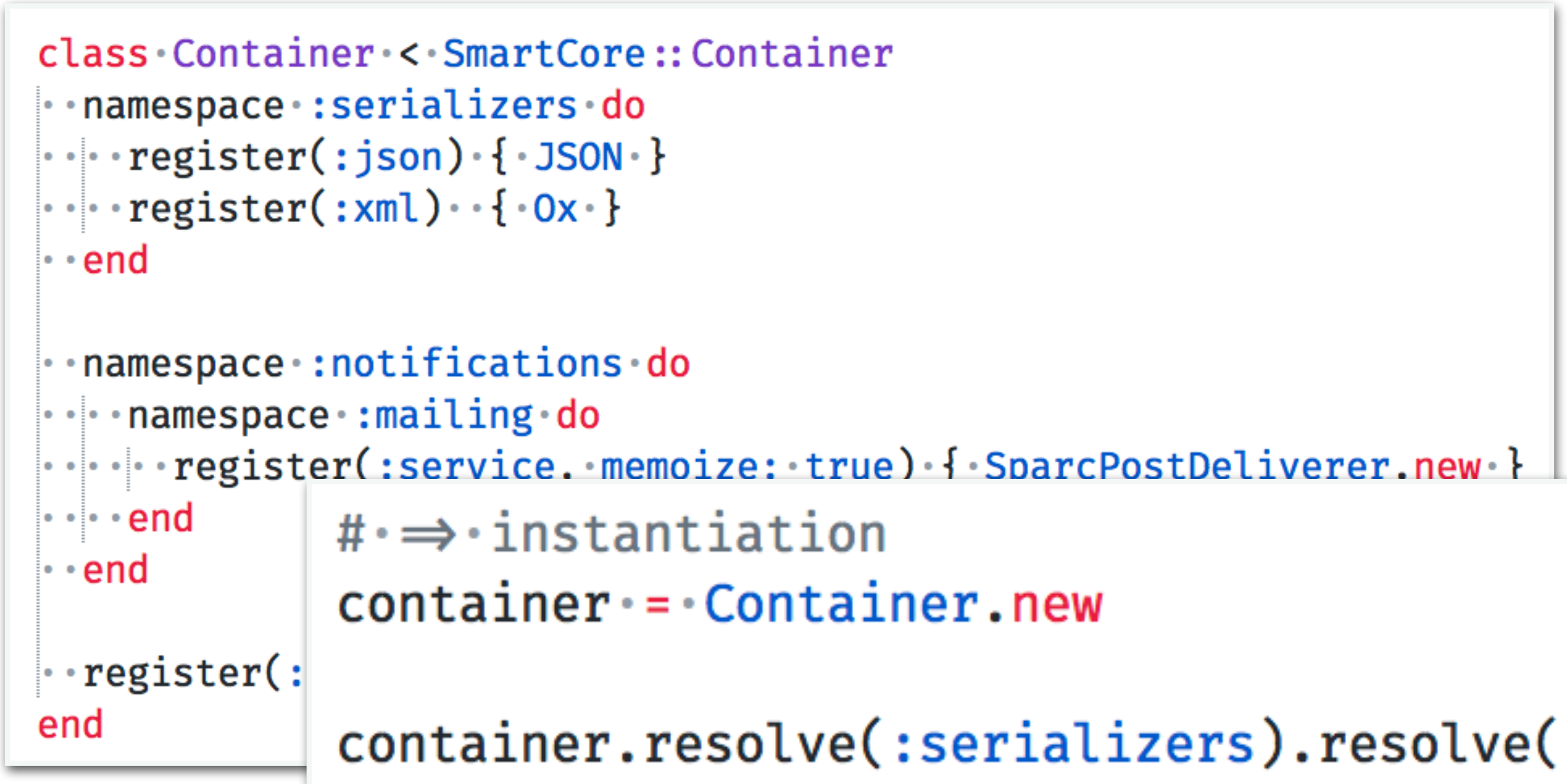


SmartCore::Container - Instance Behaviour (UK)

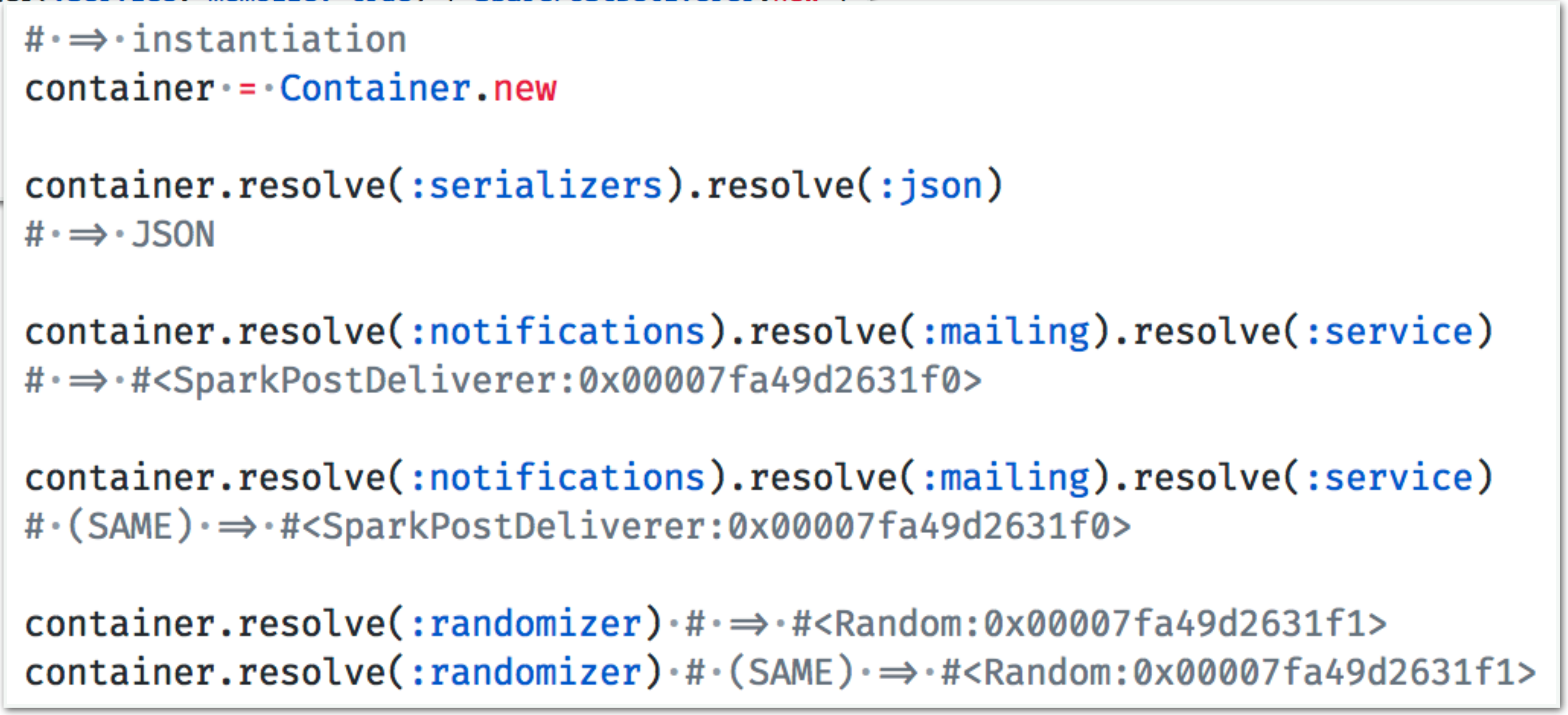


```ruby
class Container < SmartCore::Container
  namespace :serializers do
    register(:json) { JSON }
    register(:xml)  { Ox }
  end

  namespace :notifications do
    namespace :mailing do
      register(:service, memoize: true) { SparcPostDeliverer.new }
    end
  end

  register(:randomizer, memoize: true) { Random.new }
end
```

SmartCore::Container - Instance Behaviour (UK)



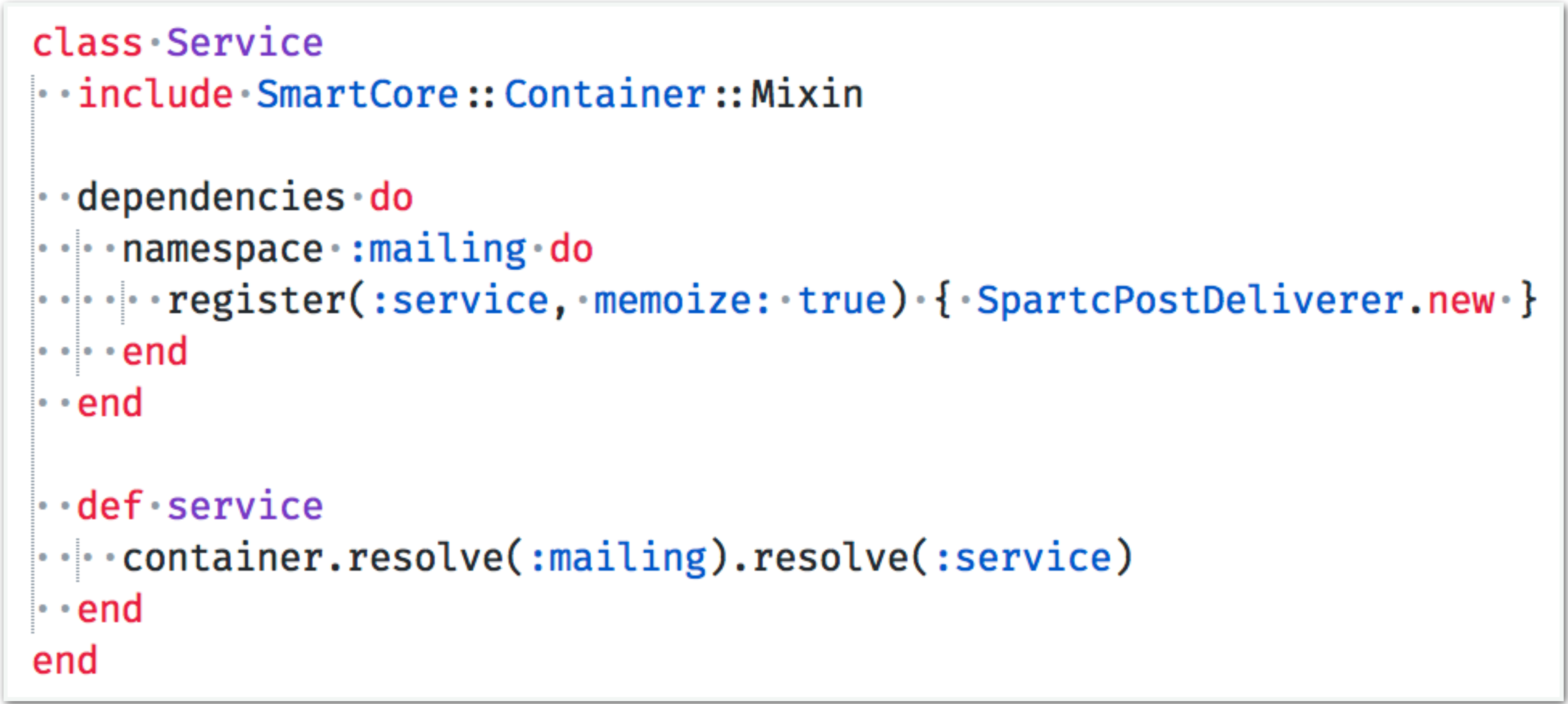
```
class Container < SmartCore::Container
  namespace :serializers do
    register(:json) { JSON }
    register(:xml)  { Ox }
  end

  namespace :notifications do
    namespace :mailing do
      register(:service, memoize: true) { SparcPostDeliverer.new }
    end
  end

  register(:
end
```



```
# => instantiation
container = Container.new

container.resolve(:serializers).resolve(:json)
# => JSON

container.resolve(:notifications).resolve(:mailing).resolve(:service)
# => #<SparkPostDeliverer:0x00007fa49d2631f0>

container.resolve(:notifications).resolve(:mailing).resolve(:service)
# (SAME) => #<SparkPostDeliverer:0x00007fa49d2631f0>

container.resolve(:randomizer) # => #<Random:0x00007fa49d2631f1>
container.resolve(:randomizer) # (SAME) => #<Random:0x00007fa49d2631f1>
```

SmartCore::Container - Mixin Behaviour

- include SmartCore::Container::Mixin
- .dependencies
- .container (глобальный)
- #container (глобальный)



```ruby
class Service
  include SmartCore::Container::Mixin

  dependencies do
    namespace :mailing do
      register(:service, memoize: true) { SpartcPostDeliverer.new }
    end
  end

  def service
    container.resolve(:mailing).resolve(:service)
  end
end
```

Service.container == Service.new.container

SmartCore::<ComingSoon>

- **SmartCore::Container**
  - state freeze;
  - #merge / #merge!;
  - container composition;
  - definition-level exceptions;
- **SmartCore::Initializer**:
  - convertible attributes (GG **tainbox**);
- **SmartCore::Operation:**
  - basic step->step->step abstraction;
  - dependency injection;
- **SmartCore::Injector:**
  - different injection strategies;
- **SmartCore::Validator**:
  - idempotent invocations over the list of attributes;
  - no-instance API;

THX

https://github.com/0exp/smart_core
https://github.com/0exp/