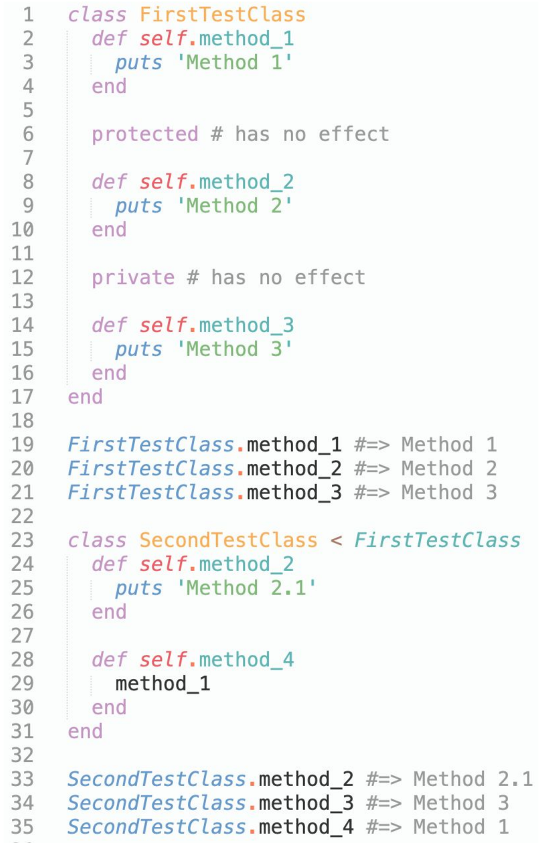
Singleton классы и модули

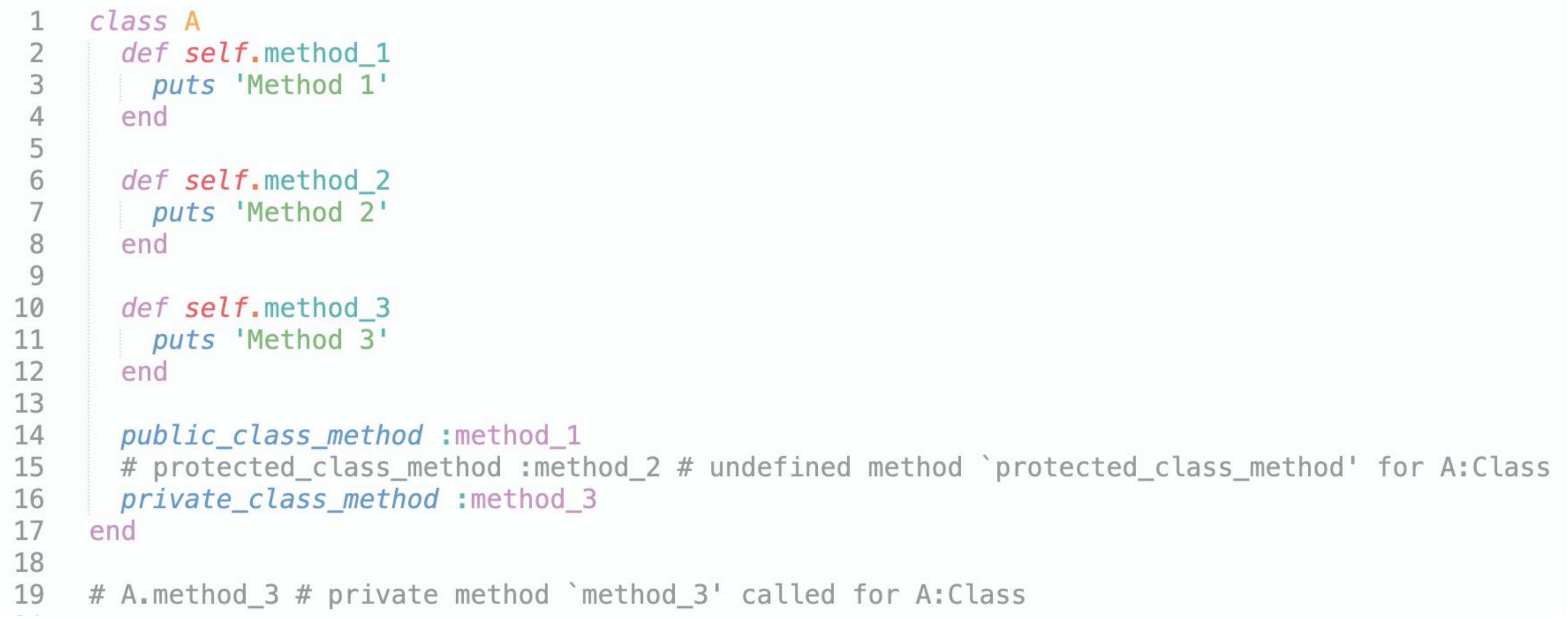
Методы класса, singleton-методы, singleton-классы, модули

Набор методов класса



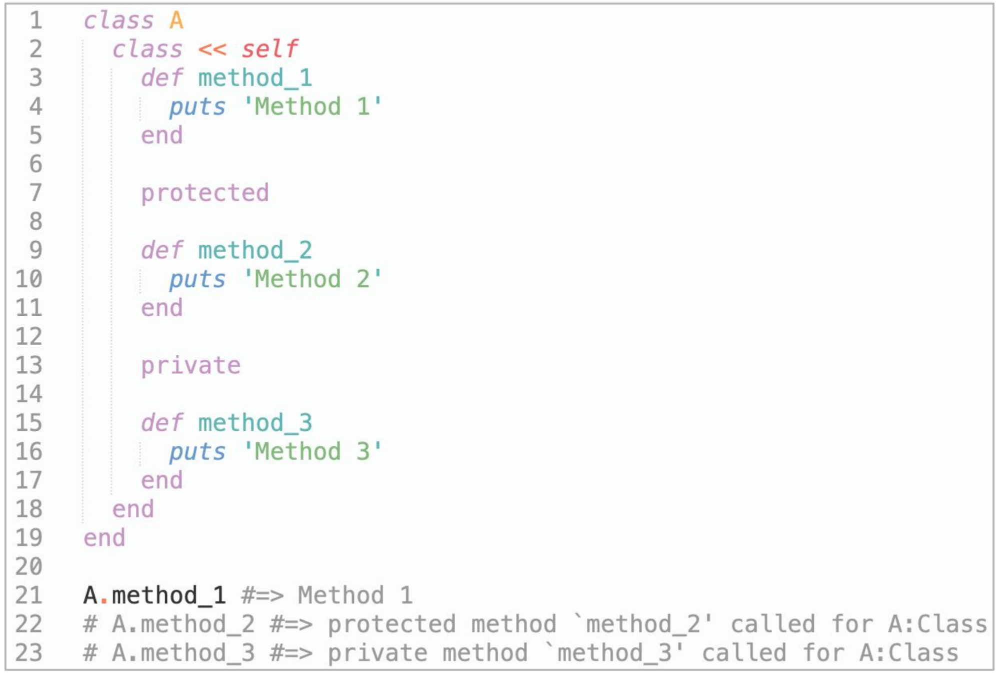
```ruby
class FirstTestClass
  def self.method_1
    puts 'Method 1'
  end

  protected # has no effect

  def self.method_2
    puts 'Method 2'
  end

  private # has no effect

  def self.method_3
    puts 'Method 3'
  end
end

FirstTestClass.method_1 #=> Method 1
FirstTestClass.method_2 #=> Method 2
FirstTestClass.method_3 #=> Method 3

class SecondTestClass < FirstTestClass
  def self.method_2
    puts 'Method 2.1'
  end

  def self.method_4
    method_1
  end
end

SecondTestClass.method_2 #=> Method 2.1
SecondTestClass.method_3 #=> Method 3
SecondTestClass.method_4 #=> Method 1
```

Приватные методы класса



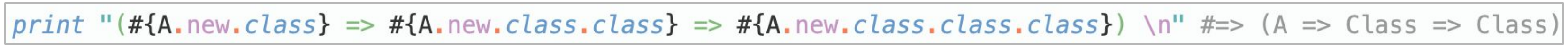
```ruby
class A
  def self.method_1
    puts 'Method 1'
  end

  def self.method_2
    puts 'Method 2'
  end

  def self.method_3
    puts 'Method 3'
  end

  public_class_method :method_1
  # protected_class_method :method_2 # undefined method `protected_class_method' for A:Class
  private_class_method :method_3
end

# A.method_3 # private method `method_3' called for A:Class
```

Класс - тоже объект



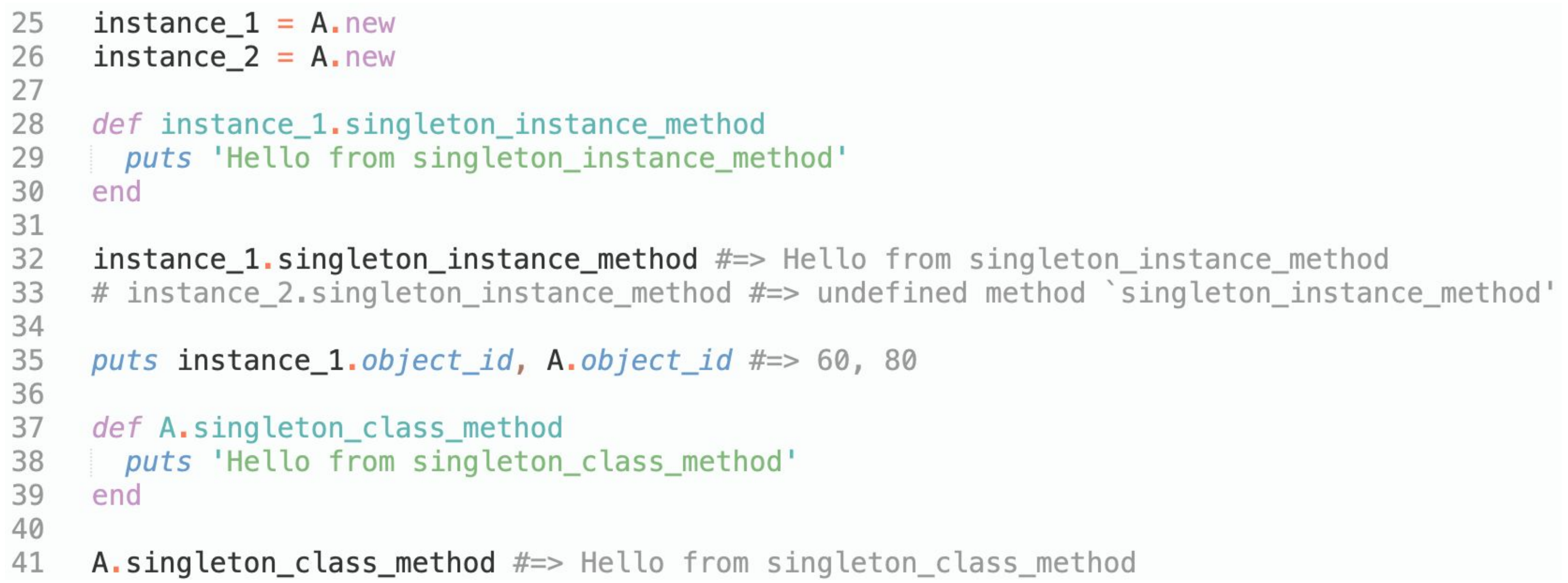
```ruby
class A
  class << self
    def method_1
      puts 'Method 1'
    end

    protected

    def method_2
      puts 'Method 2'
    end

    private

    def method_3
      puts 'Method 3'
    end
  end
end

A.method_1 #=> Method 1
# A.method_2 #=> protected method `method_2' called for A:Class
# A.method_3 #=> private method `method_3' called for A:Class
```

- Класс является экземпляром класса Class;
- Как и любой экземпляр, класс может иметь различные уровни доступа к методам.



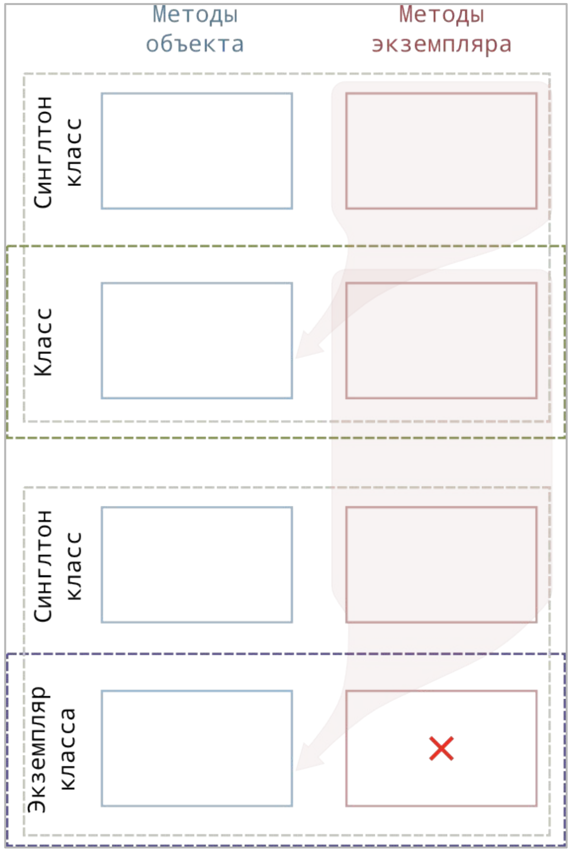
```ruby
print "(#{A.new.class} => #{A.new.class.class} => #{A.new.class.class.class}) \n" #=> (A => Class => Class)
```

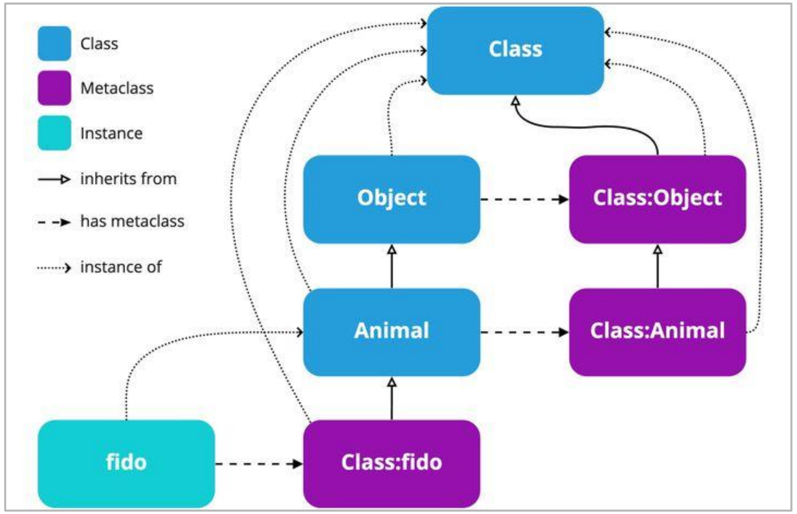
Singleton-методы

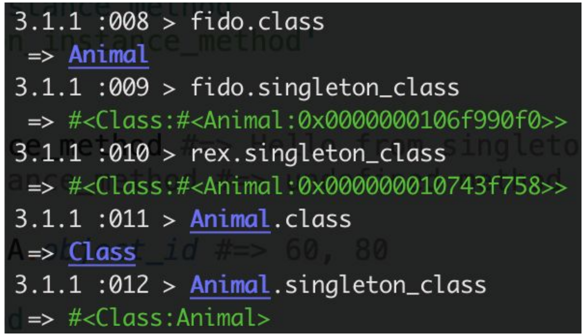


```ruby
instance_1 = A.new
instance_2 = A.new

def instance_1.singleton_instance_method
  puts 'Hello from singleton_instance_method'
end

instance_1.singleton_instance_method #=> Hello from singleton_instance_method
# instance_2.singleton_instance_method #=> undefined method `singleton_instance_method'

puts instance_1.object_id, A.object_id #=> 60, 80

def A.singleton_class_method
  puts 'Hello from singleton_class_method'
end

A.singleton_class_method #=> Hello from singleton_class_method
```

Singleton-классы

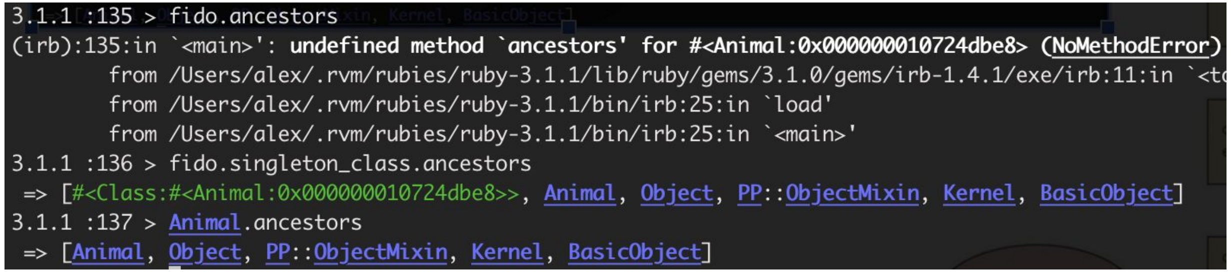




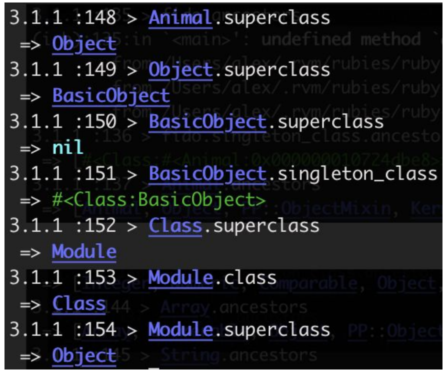


```
3.1.1 :008 > fido.class
 => Animal
3.1.1 :009 > fido.singleton_class
 => #<Class:#<Animal:0x0000000106f990f0>>
3.1.1 :010 > rex.singleton_class
 => #<Class:#<Animal:0x000000010743f758>>
3.1.1 :011 > Animal.class
 => Class
3.1.1 :012 > Animal.singleton_class
 => #<Class:Animal>
```

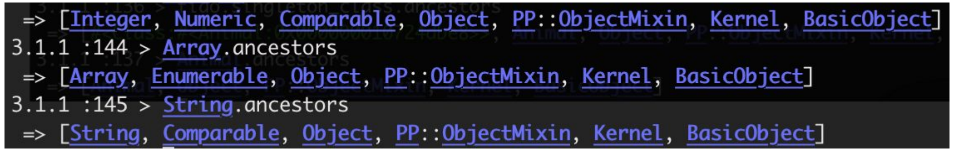
Цепочка наследования классов



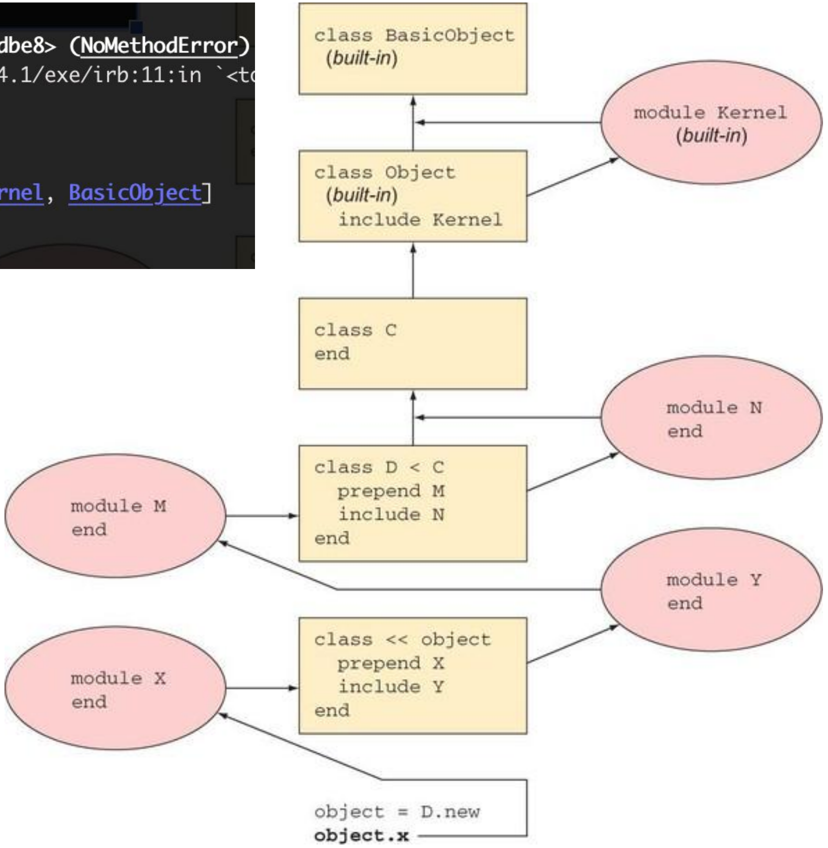
```
3.1.1 :135 > fido.ancestors
(irb):135:in `<main>': undefined method `ancestors' for #<Animal:0x000000010724dbe8> (NoMethodError)
        from /Users/alex/.rvm/rubies/ruby-3.1.1/lib/ruby/gems/3.1.0/gems/irb-1.4.1/exe/irb:11:in `<t
        from /Users/alex/.rvm/rubies/ruby-3.1.1/bin/irb:25:in `load'
        from /Users/alex/.rvm/rubies/ruby-3.1.1/bin/irb:25:in `<main>'
3.1.1 :136 > fido.singleton_class.ancestors
 => [#<Class:#<Animal:0x000000010724dbe8>>, Animal, Object, PP::ObjectMixin, Kernel, BasicObject]
3.1.1 :137 > Animal.ancestors
 => [Animal, Object, PP::ObjectMixin, Kernel, BasicObject]
```



```
3.1.1 :148 > Animal.superclass
 => Object
3.1.1 :149 > Object.superclass
 => BasicObject
3.1.1 :150 > BasicObject.superclass
 => nil
3.1.1 :151 > BasicObject.singleton_class
 => #<Class:BasicObject>
3.1.1 :152 > Class.superclass
 => Module
3.1.1 :153 > Module.class
 => Class
3.1.1 :154 > Module.superclass
 => Object
```



```
 => [Integer, Numeric, Comparable, Object, PP::ObjectMixin, Kernel, BasicObject]
3.1.1 :144 > Array.ancestors
 => [Array, Enumerable, Object, PP::ObjectMixin, Kernel, BasicObject]
3.1.1 :145 > String.ancestors
 => [String, Comparable, Object, PP::ObjectMixin, Kernel, BasicObject]
```

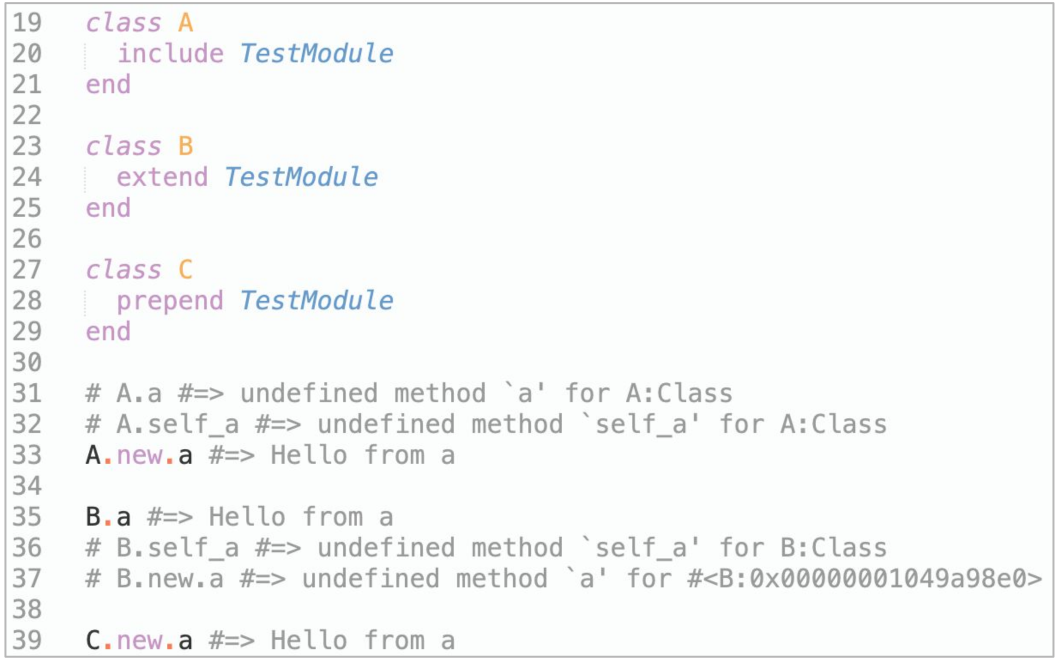


class BasicObject (built-in)
module Kernel (built-in)
class Object (built-in) include Kernel
class C end
module N end
module M end
class D < C prepend M include N end
module Y end
module X end
class << object prepend X include Y end
object = D.new
object.x

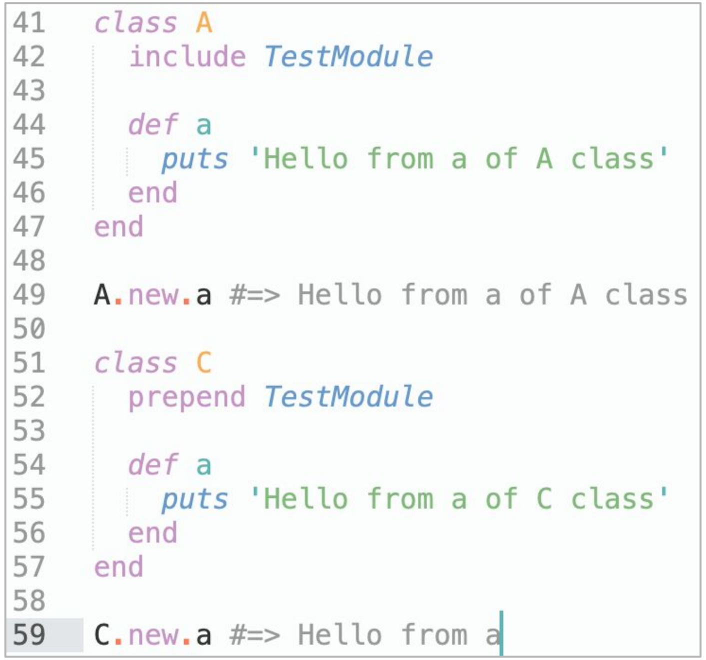
Работа с модулями. Создание

```ruby
module TestModule
  def a
    puts 'Hello from a'
  end

  def self.self_a
    puts 'Hello from self_a'
  end
end

# TestModule.a #=> undefined method `a' for TestModule:Module
# TestModule.new #=> undefined method `new' for TestModule:Module
TestModule.self_a #=> Hello from self_a
puts TestModule.class #=> Module

# class B < TestModule #=> superclass must be an instance of Class
# end
```

Подключения модуля в класс

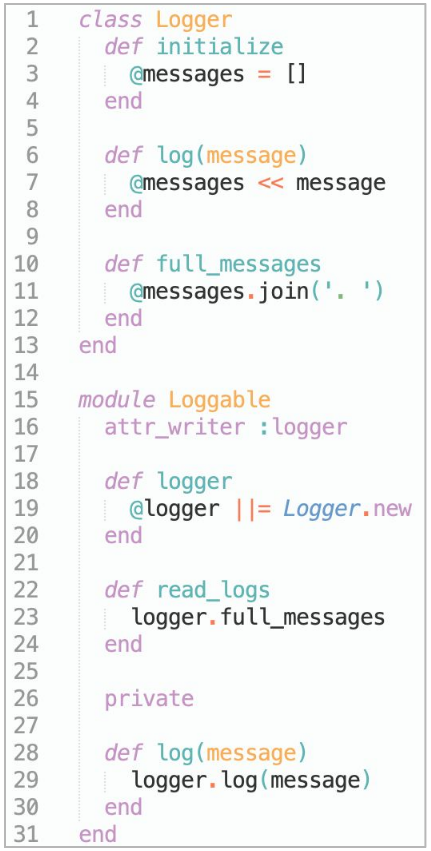


```ruby
class A
  include TestModule
end

class B
  extend TestModule
end

class C
  prepend TestModule
end

# A.a #=> undefined method `a' for A:Class
# A.self_a #=> undefined method `self_a' for A:Class
A.new.a #=> Hello from a

B.a #=> Hello from a
# B.self_a #=> undefined method `self_a' for B:Class
# B.new.a #=> undefined method `a' for #<B:0x00000001049a98e0>

C.new.a #=> Hello from a
```

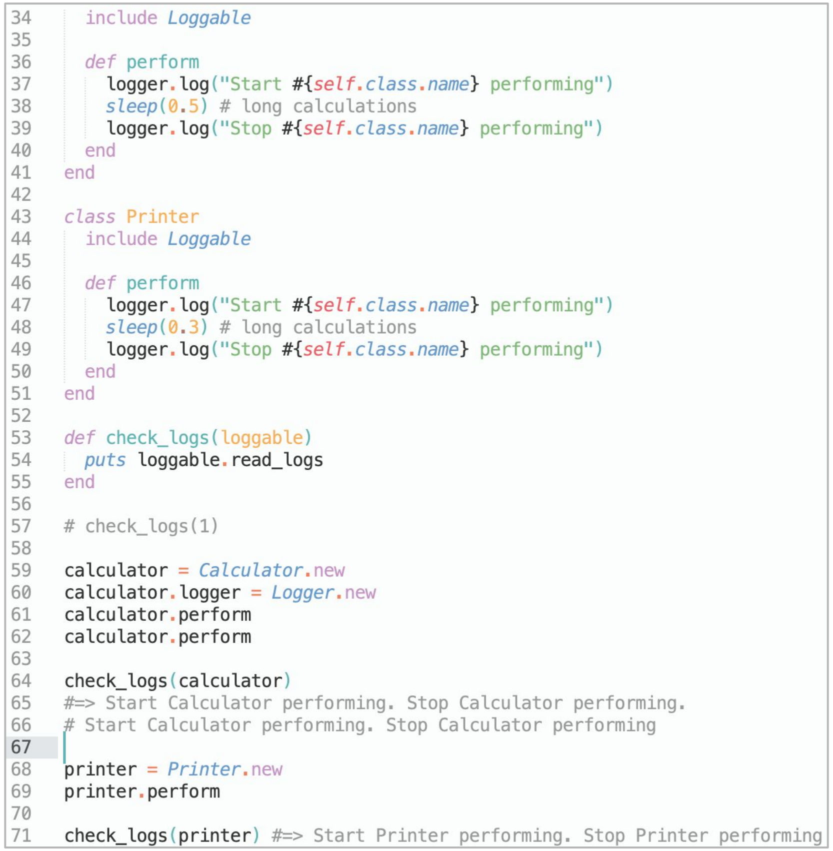


```ruby
class A
  include TestModule

  def a
    puts 'Hello from a of A class'
  end
end

A.new.a #=> Hello from a of A class

class C
  prepend TestModule

  def a
    puts 'Hello from a of C class'
  end
end

C.new.a #=> Hello from a
```

Пример работы Include модуля

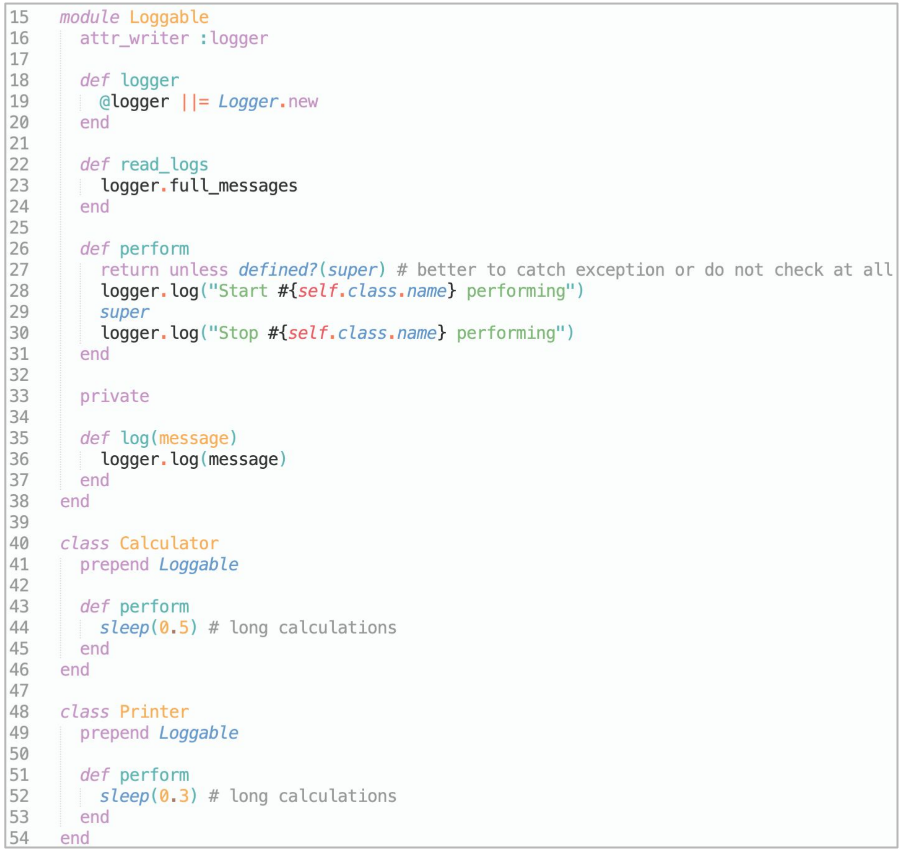


```ruby
class Logger
  def initialize
    @messages = []
  end

  def log(message)
    @messages << message
  end

  def full_messages
    @messages.join('. ')
  end
end

module Loggable
  attr_writer :logger

  def logger
    @logger ||= Logger.new
  end

  def read_logs
    logger.full_messages
  end

  private

  def log(message)
    logger.log(message)
  end
end
```

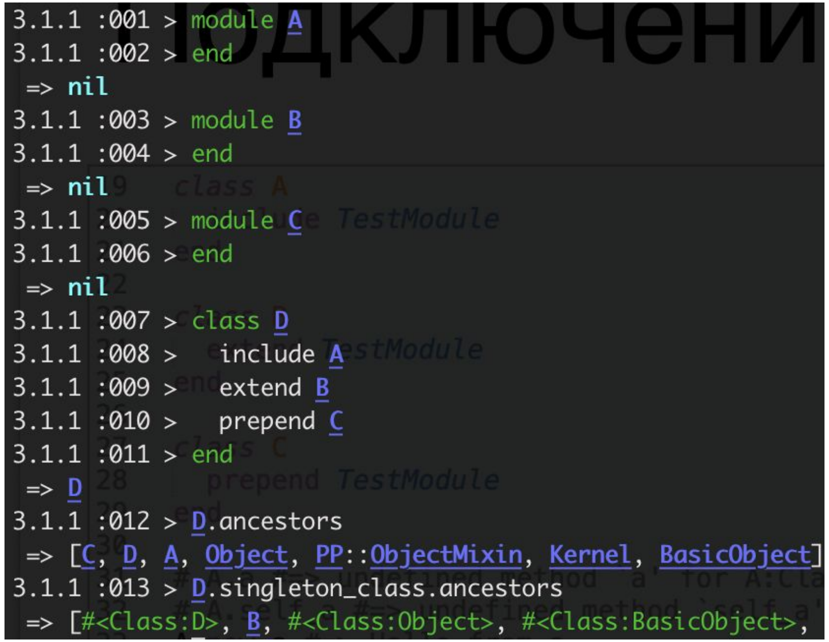


```ruby
  include Loggable

  def perform
    logger.log("Start #{self.class.name} performing")
    sleep(0.5) # long calculations
    logger.log("Stop #{self.class.name} performing")
  end
end

class Printer
  include Loggable

  def perform
    logger.log("Start #{self.class.name} performing")
    sleep(0.3) # long calculations
    logger.log("Stop #{self.class.name} performing")
  end
end

def check_logs(loggable)
  puts loggable.read_logs
end

# check_logs(1)

calculator = Calculator.new
calculator.logger = Logger.new
calculator.perform
calculator.perform

check_logs(calculator)
#=> Start Calculator performing. Stop Calculator performing.
# Start Calculator performing. Stop Calculator performing


printer = Printer.new
printer.perform

check_logs(printer) #=> Start Printer performing. Stop Printer performing
```

Пример работы Prepend модуля



```ruby
module Loggable
  attr_writer :logger

  def logger
    @logger ||= Logger.new
  end

  def read_logs
    logger.full_messages
  end

  def perform
    return unless defined?(super) # better to catch exception or do not check at all
    logger.log("Start #{self.class.name} performing")
    super
    logger.log("Stop #{self.class.name} performing")
  end

  private

  def log(message)
    logger.log(message)
  end
end

class Calculator
  prepend Loggable

  def perform
    sleep(0.5) # long calculations
  end
end

class Printer
  prepend Loggable

  def perform
    sleep(0.3) # long calculations
  end
end
```



```
3.1.1 :001 > module A
3.1.1 :002 > end
 => nil
3.1.1 :003 > module B
3.1.1 :004 > end
 => nil
3.1.1 :005 > module C
3.1.1 :006 > end
 => nil
3.1.1 :007 > class D
3.1.1 :008 >   include A
3.1.1 :009 >   extend B
3.1.1 :010 >   prepend C
3.1.1 :011 > end
 => D
3.1.1 :012 > D.ancestors
 => [C, D, A, Object, PP::ObjectMixin, Kernel, BasicObject]
3.1.1 :013 > D.singleton_class.ancestors
 => [#<Class:D>, B, #<Class:Object>, #<Class:BasicObject>,
```

Полезные ссылки

http://nashbridges.me/introducing-ruby-oop – Подробное описание того, как устроен класс;

https://blog.chumakoff.com/posts/ruby_singlton_class – Еще небольшое описание Singleton-класса;

https://medium.com/podiihq/ruby-modules-77b73c3c1054 – Описание того, как устроены модули;

https://www.rubyguides.com/2018/10/defined-keyword/ – Описание ключевого слова defined?;

https://nithinbekal.com/posts/ruby-decorators/ – Пример декорирования через модуль и через классическую композицию.

Конец! Спасибо!