```ruby
# Простой пример наследования

class Parent
  def say_hello
    puts "Hello from #{self}"
  end
end

parent = Parent.new
puts parent.say_hello

class Child < Parent

end

child = Child.new
puts child.say_hello

# Исследуем дерево наследования

puts "Child superclass: #{Child.superclass}"
puts "Parent.superclass: #{Parent.superclass}"
puts "Object.superclass: #{Object.superclass}"
puts "BasicObject.superclass #{BasicObject.superclass.inspect}"

# Пример со стандартным методом to_s

class Person
  def initialize(name)
    @name = name
  end
end

p = Person.new( "Michael" )
puts p # <Person:0x007fc812839550>

class Person
  def initialize(name)
```

```ruby
    @name = name
  end

  def to_s
    "Person named  #{@name} "
  end
end

p = Person.new( "Michael" )
puts p # Person named Michael

# Простой пример использования наследования
# в стандартной библиотеке ruby не был найден
# сейчас активно не применяется (за пределами Rails)

module Trig
  PI = 3.141592654

  def Trig.sin(x)
    # ..
  end

  def Trig.cos(x)
    # ..
  end
end

module Moral
  VERY_BAD = 0

  BAD = 1

  def Moral.sin(badness)
    # ...
  end
end

y = Trig.sin(Trig::PI/4)
```

```ruby
wrongdoing = Moral.sin(Moral::VERY_BAD)

# Пример агрегирования модуля в класс

module Debug
  def who_am_i?
    "#{self.class.name} (id: #{self.object_id}): #{self.name}"
  end
end

class Phonograph
  include Debug
  attr_reader :name

  def initialize(name)
    @name = name
  end
  # ...
end

class EightTrack
  include Debug

  attr_reader :name
    def initialize(name)
    @name = name
  end

  # ...
end

ph = Phonograph.new("West End Blues")
puts ph.who_am_i?
et = EightTrack.new("Surrealistic Pillow")
puts et.who_am_i?
```