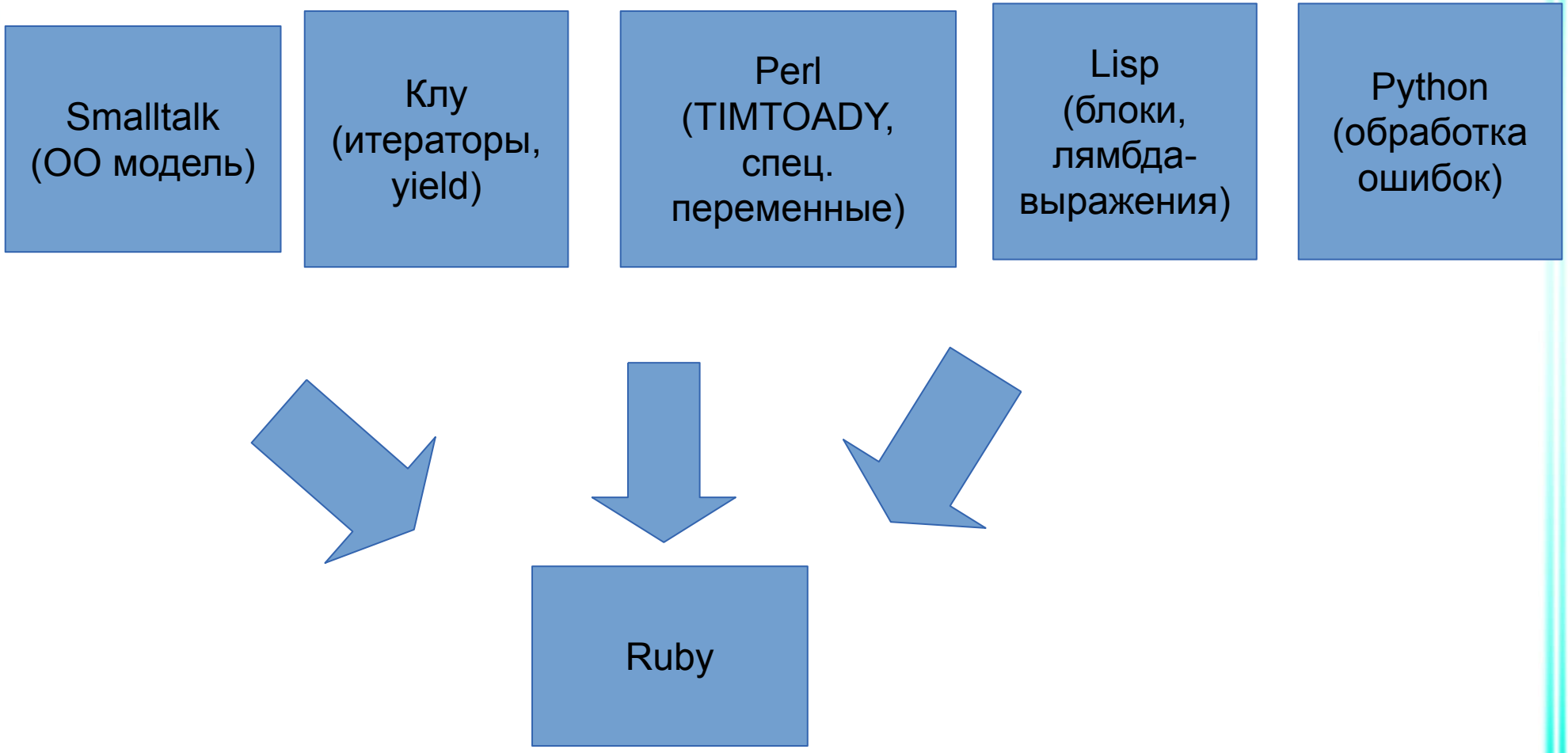
Объектно-ориентированное программирование в Ruby


bomz.org
ЗНАНИЕ - СИЛА
незнание - рабочая сила
bomz.org

Знания

- Какие бывают Ruby
- Объектно-ориентированный подход в Ruby, особенности
- Абстрагирование
- Инкапсуляция
- Наследование
- Полиморфизм
- 3 вида методов
- Методы одиночки
- Методы класса
- Миксины, модули
- Руби-стиль

Происхождение Ruby

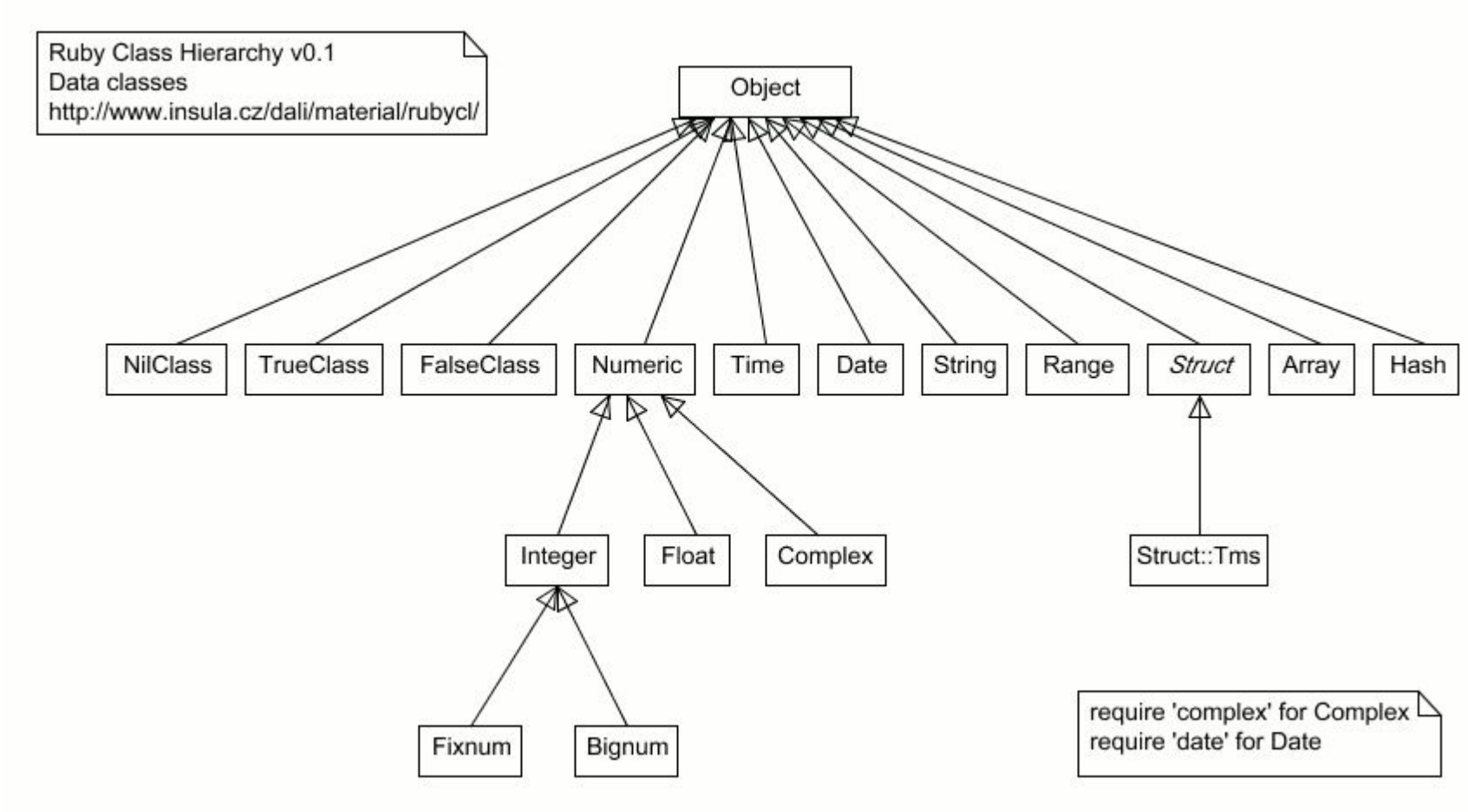


Какие бывают Ruby?

- MRI Ruby
- JRuby (JVM)
- ErRuby (Erlang Ruby, OTP)
- Rubinius (Ruby, написанный на Ruby, LLVM)
- …

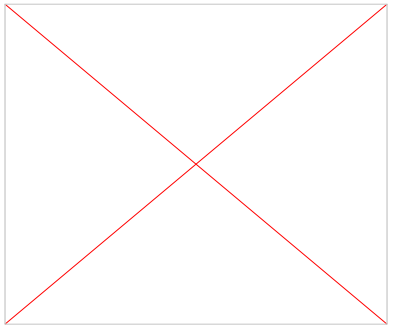
Ruby — 100% ООП

Иерархия типов в Ruby



Создание своей ООП-Вселенной

Концепция ООП Ruby (SmallTalk)



- Отправитель сообщения — объект, который вызывает метод
- Сообщение — это метод
- Получатель сообщения — это объект, метод которого вызван

Получатели и отправители

1

У меня зазвонил телефон.
— Кто говорит?
— Слон.
— Откуда?
— От верблюда.
— Что вам надо?
— Шоколада.
— Для кого?
— Для сына моего.
— А много ли прислать?
— Да пудов этак пять
Или шесть:
Больше ему не съесть,
Он у меня ещё маленький!

Получатель ← Сообщение — Отправитель

Концепция ООП Ruby

```ruby
class Receiver
  def callee
    puts "Меня вызывают"
  end
end
class Sender
  def initialize
    @receiver = Receiver.new
  end
  def call
    @receiver.callee
  end
end
sender = Sender.new
sender.call
```

Объектно-ориентированный подход

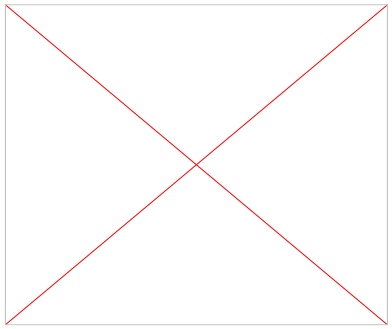
Инкапсулирование

Наследование

Полиморфизм

Абстрагирование

Абстрагирование

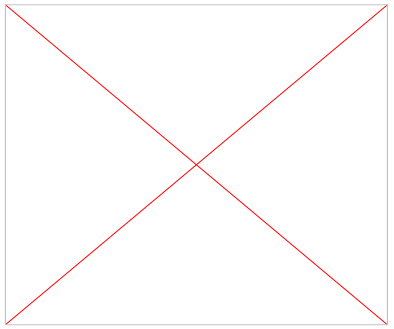


- class
- initialize
- new
- self (текущий объект)

Абстрагирование

```
class Person
  def initialize(name)
    @name = name
  end
  def to_s
    "#{@name} - обыкновенный
Человек"
  end
end
person = Person.new("Миша")
puts person # => Миша - обыкновенный
Человек
```

Инкапсуляция

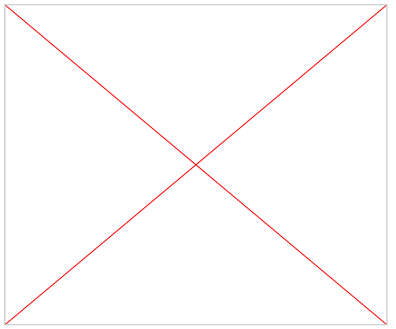


- Методы установщики и получатели
  - val(), val=(value)
  - @val
  - attr_reader, attr_writer
- Модификаторы доступа
  - private
  - public
  - protected

Инкапсуляция

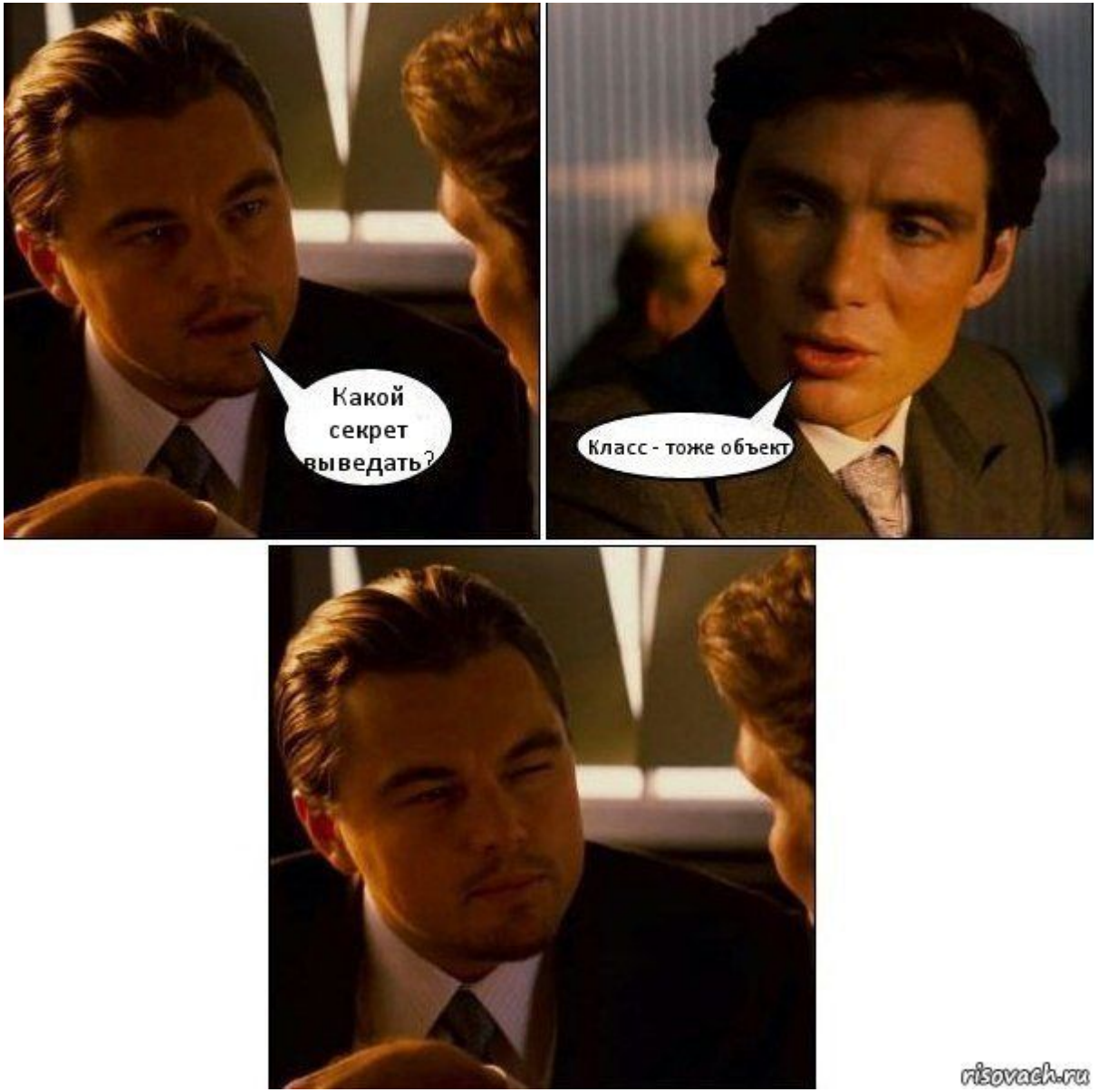
```ruby
class Person
  def name
    @name
  end
  def name=(value)
    @name = value
  end
end
person = Person.new
person.name = "Миша"
puts person.name
```

Особенности private



- Метод м.б. вызван только неявным получателем
- Явным получателем вызывать нельзя (в т.ч. self)
- Позволяет вызывать закрытый (private) метод только внутри других методов объекта класса и его наследников
- …

Особенности private



Особенности private

```
class Corporation
  def press_release
    puts self.top_secret # Что произойдёт здесь?
  end

  private
  def top_secret
    "Класс - это тоже объект"
  end
end

class Sleeper
  def incept(corporation)
    corporation.top_secret
  end
end

corp = Corporation.new
corp.press_release
```

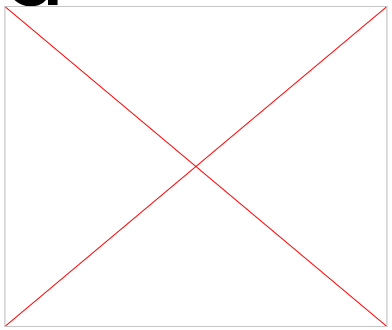
Особенности private

```
class Corporation
  def press_release
    puts self.top_secret # NoMethodError
  end

  private
  def top_secret
    "Класс - это тоже объект"
  end
end

class Sleeper
  def incept(corporation)
    corporation.top_secret
  end
end

corp = Corporation.new
corp.press_release

kobb = Sleeper.new
kobb.incept(corp) # NoMethodError
```

Особенности protected

- Защищённый метод m м.б. вызван явным получателем
- Отправитель o и Получатель p — объекты класса либо же подкласса, где объявлен защищённый метод m
- Получение доступа объектам класса к состоянию друг друга, при этом ограничение доступа снаружи
- …

Особенности protected



```
class Man
  def talk
    self.topless_secret
  end
  protected
    def topless_secret
      puts "По секрету всему свету"
    end
end
misha = Man.new
misha.talk
misha.topless_secret # Ошибка
```

Особенности protected

```
class Man
  def talk(person)
    person.topless_secret
  end
  protected
    def topless_secret
      puts "По секрету всему свету"
    end
end
misha = Man.new
masha = Man.new
misha.talk masha
```

Особенности protected

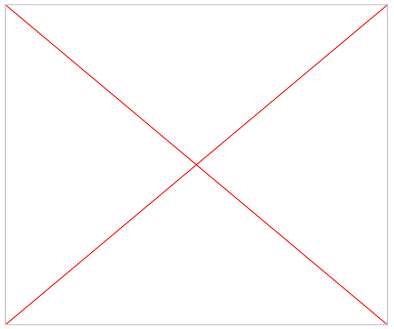
```ruby
class Man
  def initialize
    @favorite_book = "Зелёная"
  end
  def talk(person)
    puts person.topless_secret
  end
  protected
    def favorite_book
      @favorite_book
    end
    def topless_secret
      self.favorite_book
    end
end
```

```ruby
class Designer < Man
  def initialize
    @favorite_book = "Мастер и Маргарита"
  end
end
class Programmer < Man
end
misha = Programmer.new
masha = Designer.new
misha.talk(masha)
```

Инкапсуляция

```
class Person
  attr_accessor :name

end
person = Person.new
person.name = "Миша"
puts person.name
```

Наследование



- Класс-родитель и класс-наследник (class < class)
- Получение всех методов и полей класса-родителя
- Super
- Утиная типизация

Наследование

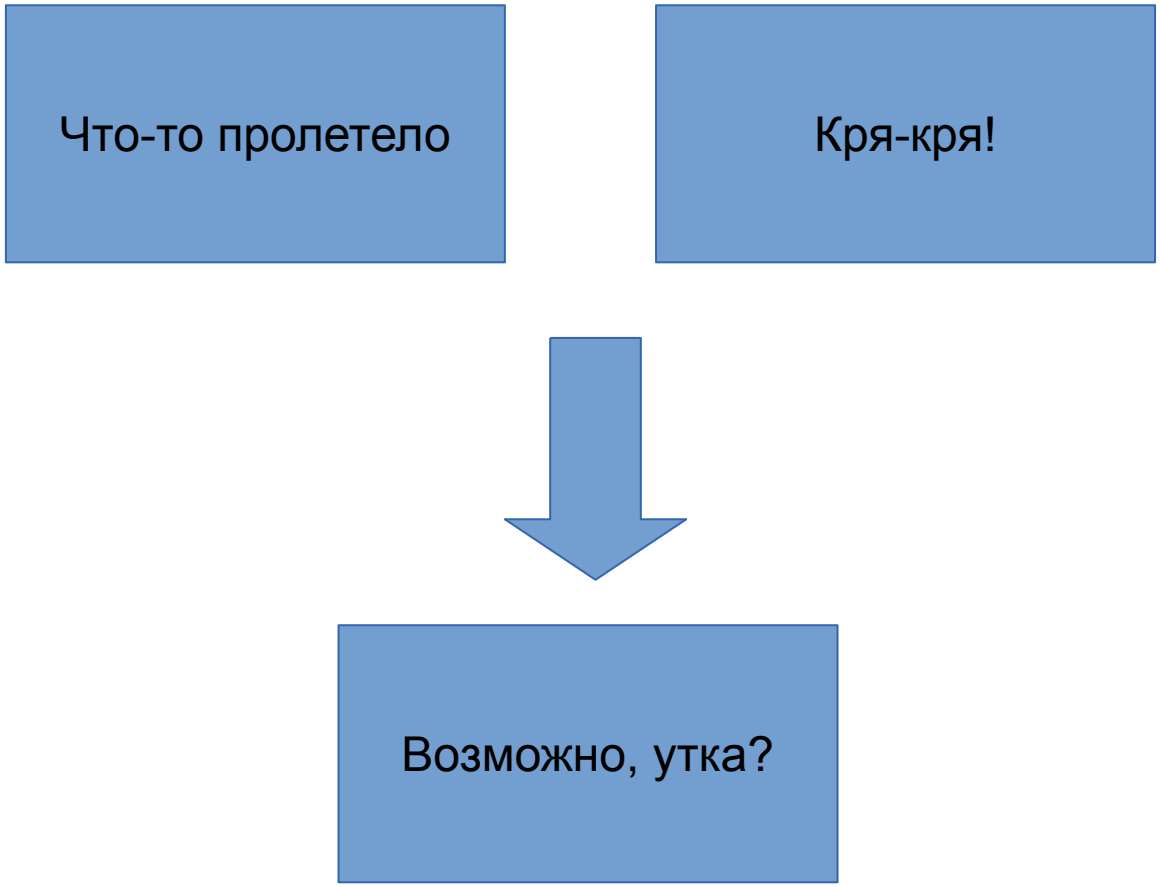
```
class ArticlesController < ApplicationController
end
```

Кот — это Жидкость

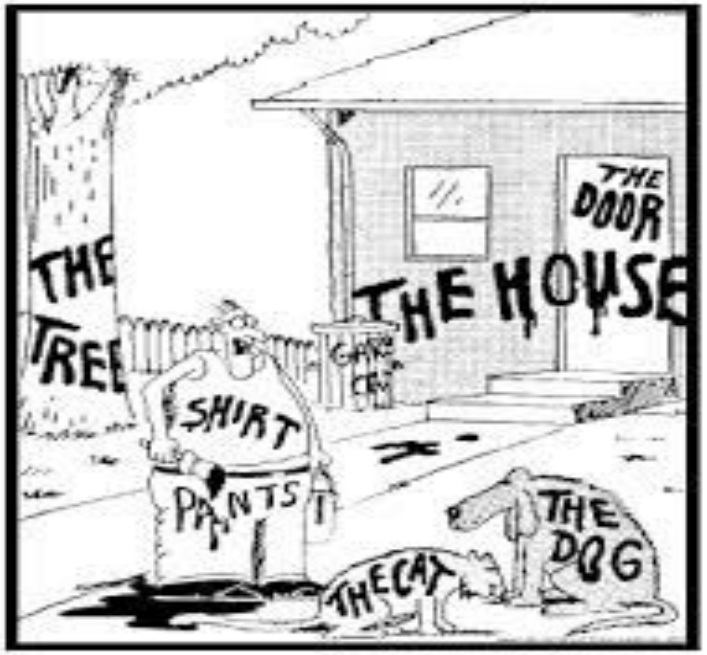
Неявная (утиная) типизация

Вид динамической типизации, когда возможность использования объекта определяется наличием определённых методов и свойств.
В отличие от наследования.

Неявная (утиная) типизация



Статическая типизация



"Now! *That* should clear up
a few things around here!"

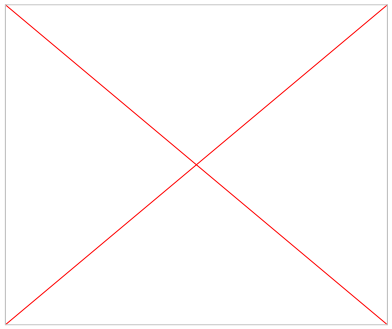
Неявная (утиная) типизация

```
class Silent
  def think
    puts "Думаю"
  end
end
class Chatterbox
  def talk
    puts "Меля Емеля, твоя неделя"
  end
end
class Babbler
  def talk
    puts "И теперь я могу сказать вам, что-то
интересное"
  end
end
```

Неявная (утиная) типизация

```
class TalkShow
  def run(guest)
    if guest.respond_to? :talk
      guest.talk
    else
      raise StandardError, "Кого вы нам прислали!?"
    end
  end
end
TalkShow.new.run(Babbler.new)
TalkShow.new.run(Chatterbox.new)
TalkShow.new.run(Silent.new)
```

Полиморфизм



- Переопределение методов
- Переопределение операторов
- Примеси

Полиморфизм

```
class Person
  def work
    puts "Я тружусь"
  end
end
class Developer < Person
  def work
    puts "Я программирую"
  end
end
class Designer < Person
  def work
    puts "Я проектирую и рисую"
  end
end
```

Полиморфизм

```ruby
class Point
  attr_accessor :x, :y

  def initialize(x,y)
    @x,@y = x,y
  end

  def +(point)
    Point.new(@x+point.x, @y+point.y)
  end
end

point = Point.new(1,2)
other_point = Point.new(2,5)

super_point = point + other_point
puts super_point.x
```

Примеси (mixins), модули

Примеси (mixin) — механизм повторного использования кода в различных классах

Модули — реализация примесей в Ruby

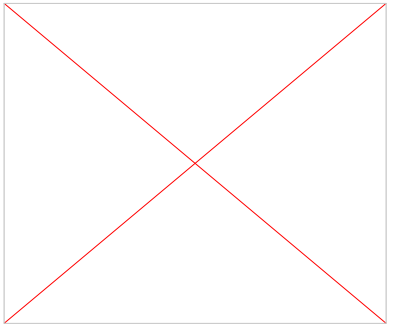
- include Module — включение методов модуля для экземпляров класса
- extend Module — включение методов модуля для класса

Примеси (mixins), модули

```
class Developer
  def write_code
    puts "Я пишу код на Ruby"
  end
end
class Administrator
  def config
    puts "Я настраиваю сервер на Windows"
  end
end
class DevOpsEngineer
  def write_code
    puts "Я пишу код на Ruby"
  end
  def config
    puts "Я настраиваю сервер на Windows"
  end
end
```

Примеси (mixins), модули

```ruby
module CodeWritable
  def write_code
    puts "Я пишу код на Ruby"
  end
end
class Developer
  include CodeWritable
end
module Configurable
  def config
    puts "Я настраиваю сервер на Windows"
  end
end
class Administrator
  include Configurable
end
class DevOpsEngineer
  include CodeWritable
  include Configurable
end
```



3 вида методов

- Методы экземпляра класса
- Методы класса
- Методы одиночки (singletone methods)

Методы одиночки

```
class Competence; end
compy = Competence.new
def compy.level
  10
end
puts compy.level
```

Методы класса

```
class Competence
  def self.about
    puts "Я - метод класса Competence"
  end
end
Competence.about
```

Методы класса

```
class Competence
  class « self
    def about
      puts "Я - метод класса Competence"
    end
  end
end
Competence.about
```

Что ещё нужно знать об ООП в Ruby

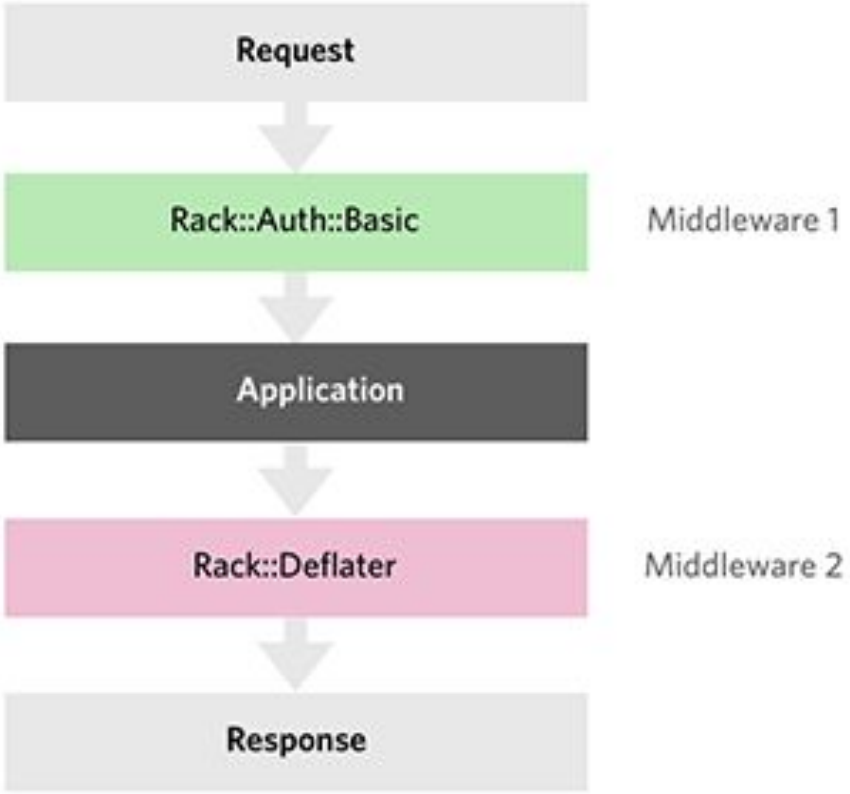
- Классы в Ruby открыты
- Почти всё в Ruby — объект, даже… класс!
  - (самостоятельно) Существуют метаклассы
- Желательно переопределять методы Object в своём классе: to_s, ==, eql?, hash:

```
def hash
  res = 17
  res = 37*hash + @value.hash
  res
end
```

Почти всё в Ruby — объект

2.+(3
)

Rack middleware



Ruby-стиль разработки

Не используйте фигурные скобки для ограничения хешей, передаваемых методу, и скобки вокруг параметров для методов, являющихся частью DSL

```
validates(:name, { presence: true, length: { within: 1..10 }
```

```
validates :name, presence: true, length: { within: 1..10 }
```

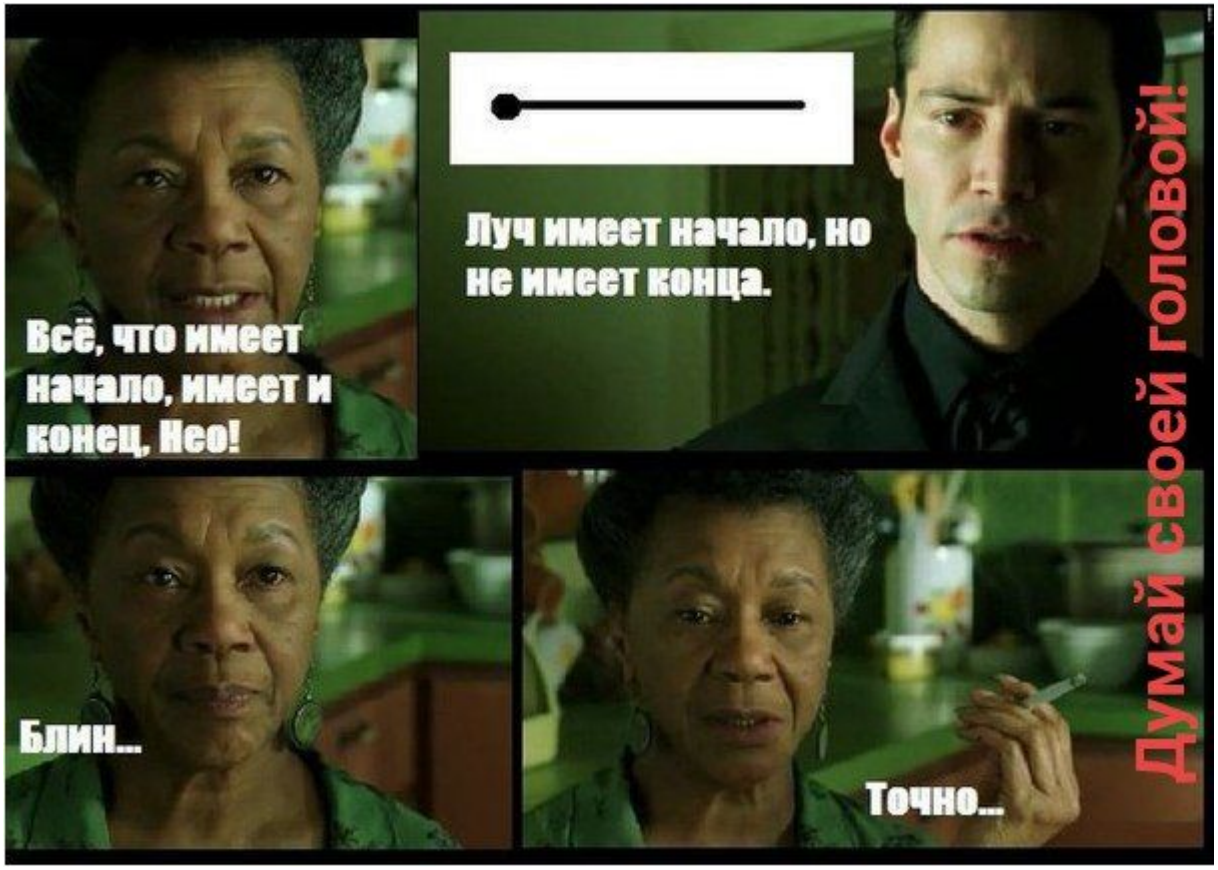
Не используйте фигурные скобки для ограничения хешей, передаваемых методу

```
user.set({ name: 'John', age: 45, permissions: { read: true
```

```
user.set(name: 'John', age: 45, permissions: { read: true }
```

HO

Думайте своей головой!




Ты молод, креативен, талантлив? Амбициозен, уверен в себе, полон свежих идей? А делать хоть что-нибудь умеешь?!
Atkritka.com

Умения

- Создавать классы и объекты
- Скрывать методы
- Создавать иерархии классов
- Разделять функциональность с помощью модулей
- Использовать примеси
- Автозагружать классы с разных папок
- использовать хеши в JS-стиле

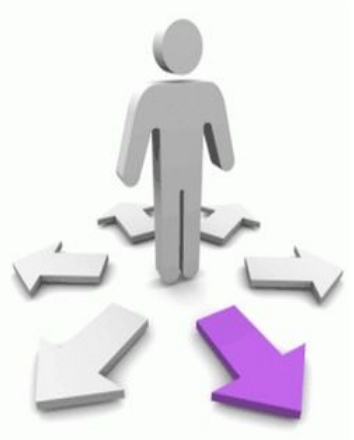
Организуем экспедицию: ООП-Вселенная Ивана Ефремова

Использовать хеши в JS-стиле


```
# Хак - использовать хеши в JS-стиле
class Hash
  def method_missing(id)
    self[id]
  end
end
person = { age: 15, name: 'Вася' }
p person.age
```

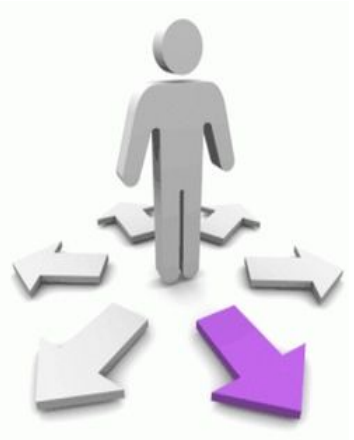

ТУДА
100м
СЮДА
200м
ВЕЧНАЯ НЕОПРЕДЕЛЕННОСТЬ!
КАК ЖИТЬ?
DEMOTIVATORS.RU

Неопределённости



- Почему в Ruby нет абстрактных классов и интерфейсов?
  - Утиная типизация
  - Позднее связывание
- Почему использовать переменные класса считается плохим стилем?
- Почему не создаются методы-одиночки для чисел?
  - Числа и символы являются непосредственными значениями, не ссылками (стр. 100 Мацумото)
  - Обрезанные объекты. Для них не существует методов мутаторов и нельзя создавать синглтон-методы

Неопределённости



```
class Competence
  @@min_level = 3
  def self.min_level
    @@min_level
  end
end

class PrivateCompetence < Competence
  @@min_level = 5
end

p PrivateCompetence.min_level # 5
p Competence.min_level # 5 %-(
```

Результат


Это же так просто!
Я это сделаю!
Я могу это сделать!
Я попытаюсь...
Как мне это сделать?
Я хочу это сделать!
Я не могу это сделать (не умею, не получится)
Я не буду этого делать.

Результат

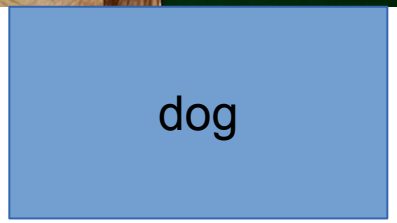
- Научились использовать ООП на языке Ruby
  - Понимать особенности private и protected
  - Использовать модули
  - Проектировать свою ООП-Вселенную
  - …

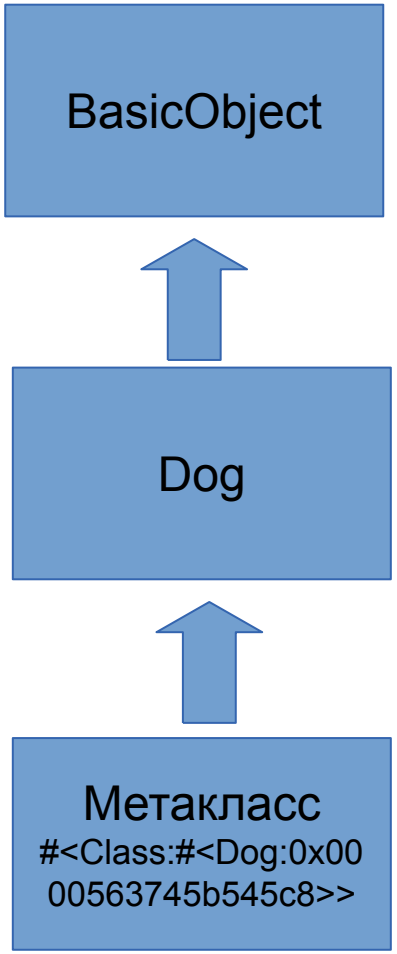
Самостоятельно

- Исключения
- Сравнение (eql?, ==, ===, equal?)
- Создание собственных итераторов, yield
- coerse
- Создание собственных методов-мутаторов
- Автозагрузка констант
- * extend self
- * метакласс (обособленный класс)
- …

Метакласс

```
class Object
  def metaclass
    class << self
      self
    end
  end
end

class Dog; end

barbos = Dog.new
puts barbos.metaclass
```





Автозагрузка классс

- require_relative «filename» — немедленное подключение, поиск файла от текущей папки
- require «filename» - сразу подключаем, поиск файла в путях из $LOAD_PATH
- autoload :Class, «filename» - как require, отложенное подключение, в момент использования

Автозагрузка классов в Rails

```
module MyLib
  extend ActiveSupport::Autoload

  autoload :Model

  eager_autoload do
    autoload :Cache
  end
end

MyLib.eager_load!
```

Автозагрузка классов в Rails

```
require 'active_support'

ActiveSupport::Dependencies.autoload_paths = [
'lib/'
]
```

```
# config/application.rb
config.autoload_paths += %W( events/ )
```

Домашнее задание

- Реализуйте копирование заданного файла в другой файл.
- Имя нового файла должно начинаться с префикса, равного текущей дате-времени.
- Например, имя файла для копирования arrays.rb, он начал копироваться 16 февраля 2015 года в 14 часов 12 минут 10 секунд.
- Тогда имя нового файла 20150216141210arrays.rb.

Домашнее задание

- + к предыдущему заданию:
- перед каждой строчкой добавьте отдельную строку в виде комментария, которая содержит фразу: длина строки - N символов.
- И в конце - комментарий, содержащий фразу: общее количество символов M.

Домашнее задание

- Например:

```
# Длина строки 12 символов
class Person
# Длина строки 3 символов
end
# Длина строки 18 символов
misha = Person.new
# Общее количество символов - 33
```

Домашнее задание

- Постройте частотный словарь по словам из текстового файла stations.txt

Домашнее задание

- Создайте класс профессиональная компетенция ProfCom, с полями title.
- Создайте класс профессионал (Professional). С атрибутами name и competences (массив ProfCom). И методом learn(compy), который добавляет compy в competences.
- Переопределите набор методов (to_s, <=>). to_s выводит визитную карточку — информацию о профессионале, как его зовут и какими компетенциями он обладает. Метод <=> позволяет корректно сравнить уровень профессионалов, по количеству компетенций.
- Создайте классы Разработчик (Developer) и инженер (Engineer) — наследуют от Professional.
- Создайте 10 компетенций.
- Создайте команду из 3 разработчиков и 2 инженеров.
- Каждый из них должен изучить случайное кол-во компетенций

Домашнее задание

- При вводе URL developers выводить всех разработчиков

Домашнее задание

- Создать документ *.ods. В ячейке A1 — слово ФИО.
- В ячейках столбца A — ФИО сотрудников. Но вот незадача!
- В некоторых из них имена сотрудников имеют по ошибке латинскую букву. Например, c (цэ) вместо с (эс).
- Найдите, в каких именно. Подсказка: использовать gem roo.

Список источников

- Основное
- Флэнаган, Мацумото. «Язык программирования Ruby»
- Ruby docs
- Ruby стиль
- Дополнительное
- Ruby koans
- «Путь Ruby» (более новая версия Ruby)
- Abstract class in Ruby
- Composition over inheritance (Joshua Bloch, Effective Java)