Yalantis

Лекция 3: ООП

Курс лекций по основами web-разработки на языке программирования Ruby

Парадигма программирования

Совокупность идей и понятий, определяющих стиль написания компьютерных программ (подход к программированию). Это способ формирования команд определяющий организацию вычислений и структурирование работы, выполняемой компьютером.

Основные парадигмы программирования

- **Декларативное программирование**
- **Императивное программирование**
- **Функциональное программирование**
- **Объектно-ориентированное программирование**

В неспециализированных задачах вам скорее случиться работать в рамках одной из них либо в рамках мультипаардигменнвх языков. Список на самом деле намного шире.

Декларативное программирование

HTML

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>This is a Heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

CSS

```
html, body {
      font-family:Verdana;
      font-size: 15px;
      line-height: 1.5;
}
```

SQL

```
SELECT column name(s)
FROM table name
WHERE condition
GROUP BY column name(s)
ORDER BY column_name(s);
```

Императивное программирование

Ruby

```
print "Hello. Please enter a celsius value: "
celsius = gets
fahrenheit = (celsius.to_i * 9 / 5) + 32
print "The fahrenheit equivalent is: "
print fahrenheit
puts "."


# REFACTORED METHOD (denser, but less readable)
# print "Hello. Please enter a celsius value: "
# print "The fahrenheit equivalent is: " gets.to_i *
9/5 + 32, ".\n"
```

Bash

```
echo "*** Converting between the different temperature
scales ***"
echo "1. Convert Celsius temperature into Fahrenheit"
echo "2. Convert Fahrenheit temperatures into Celsius"
echo -n "Select your choice (1-2) : "
read choice
if [ $choice -eq 1 ]
then
echo -n "Enter temperature (C) : "
read tc
# formula Tf=(9/5)*Tc+32
tf=$(echo "scale=2;((9/5) * $tc) + 32" |bc)
echo "$tc C = $tf F"
elif [ $choice -eq 2 ]
then
echo -n "Enter temperature (F) : "
read tf
# formula Tc=(5/9)*(Tf-32)
tc=$(echo "scale=2;(5/9)*($tf-32)"|bc)
echo "$tf = $tc"
else
echo "Please select 1 or 2 only"
exit 1
fi
```

Функциональное программирование

Haskell

```haskell
type Temperature = Double

--Farenheit to Celsius
ftc :: Temperature -> Temperature
ftc = (*(5/9)) . ((+) (-32))


--To Farenheit
--Celsius to Farenheit
ctf :: Temperature -> Temperature
ctf = (+32) . (*(9/5))


--Delisle to Farenheit
dtf :: Temperature -> Temperature
dtf = (212-) . (*(6/5))
```

```haskell
--To Delisle
--Farenheit to Delisle
ftd :: Temperature -> Temperature
ftd = (/6) . (1060-) . (5*)


--Celsius to Delisle
ctd :: Temperature -> Temperature
ctd = (150-) . ((3/2)*)


--To Celsius
--Delisle to Celsius
dtc :: Temperature -> Temperature
dtc = (100-) . (2/3*)
```

Объектно-ориентированное программирование

Java

```
// Java program to convert Celsius
// scale to Fahrenheit scale
class GFG
{
// function to convert Celsius
// scale to Fahrenheit scale
static float Cel_To_Fah(float n)
{
	return ((n * 9.0f / 5.0f) + 32.0f);
}

// Driver code
public static void main(String[] args) {
	float n = 20.0f;
	System.out.println(Cel_To_Fah(n));
}
}
```

C#

```
using System;

class GFG {

// function to convert Celsius
// scale to Fahrenheit scale
static float Cel_To_Fah(float n)
{
	return ((n * 9.0f / 5.0f) + 32.0f);
}

// Driver code
public static void Main()
{
	float n = 20.0f;
	Console.Write(Cel_To_Fah(n));
}
}
```

Монстр сложности

«Программные проекты редко терпят крах по техническим причинам. Чаще всего провал объясняется неадекватной выработкой требований, неудачным планированием или неэффективным управлением. Если же провал обусловлен всё-таки преимущественно технической причиной, очень часто ею оказывается неконтролируемая сложность. Иначе говоря, приложение стало таким сложным, что разработчики перестали по-настоящему понимать, что же оно делает».

Стив Макконнелл, «Совершенный код»

«Существенная черта промышленной программы — уровень сложности: один разработчик практически не в состоянии охватить все аспекты такой системы. Грубо говоря, сложность промышленных программ превышает возможности человеческого интеллекта».

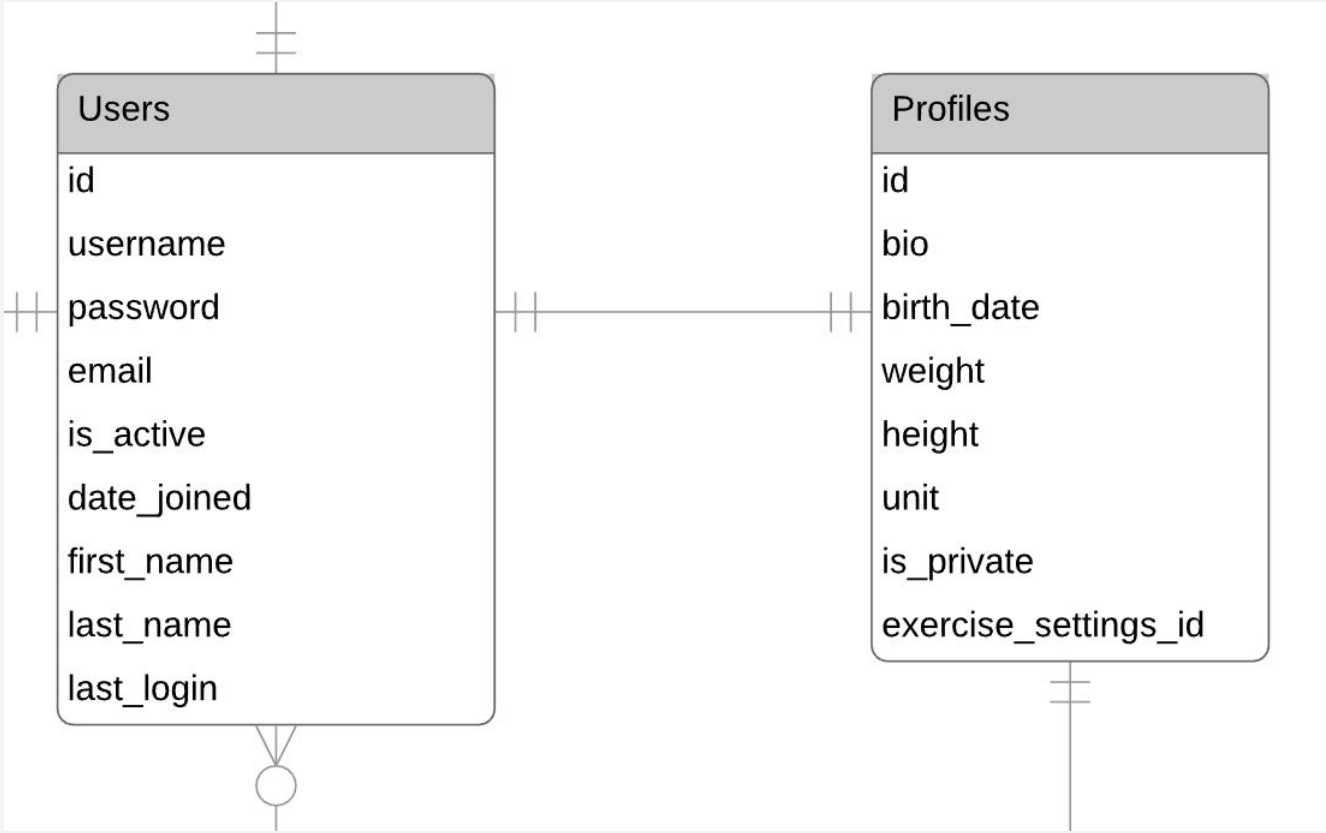
Гради Буч, «Объектно-ориентированный анализ и проектирование»

«Управление сложностью — квинтэссенция программирования»

Брайан Керниган

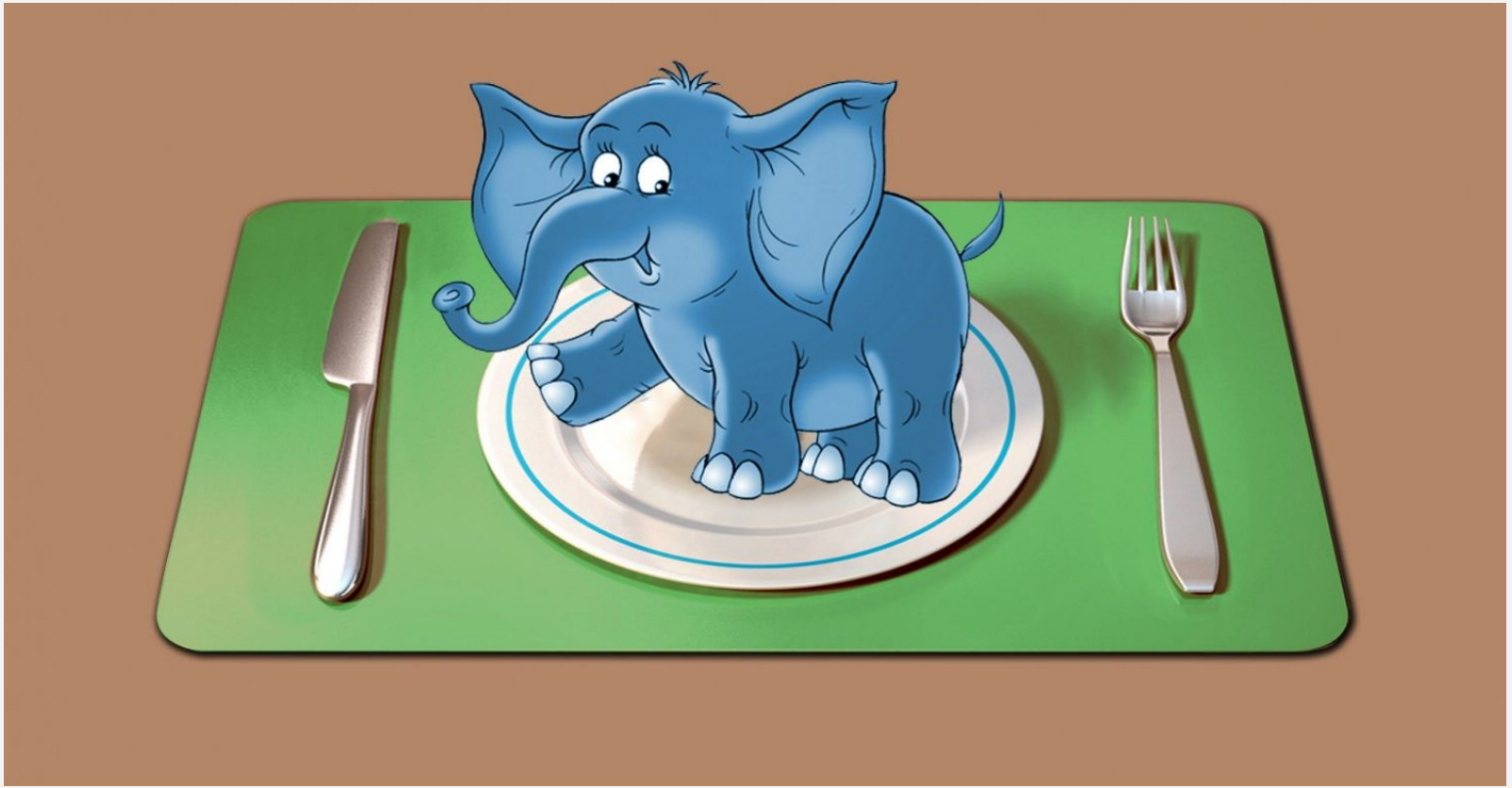
Модели реального мира

Пользователь сайта

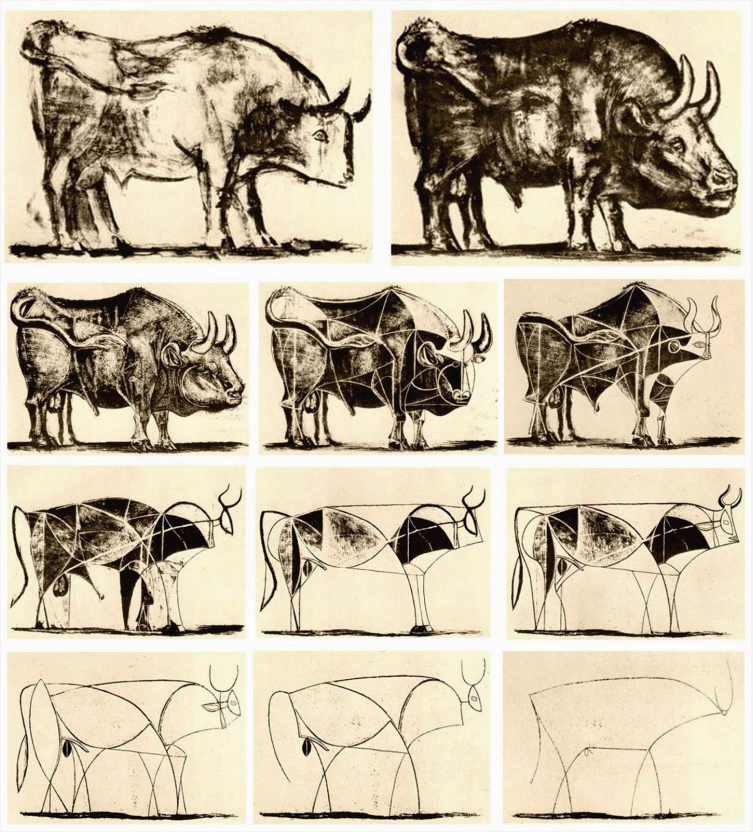


Digital Fingerprint

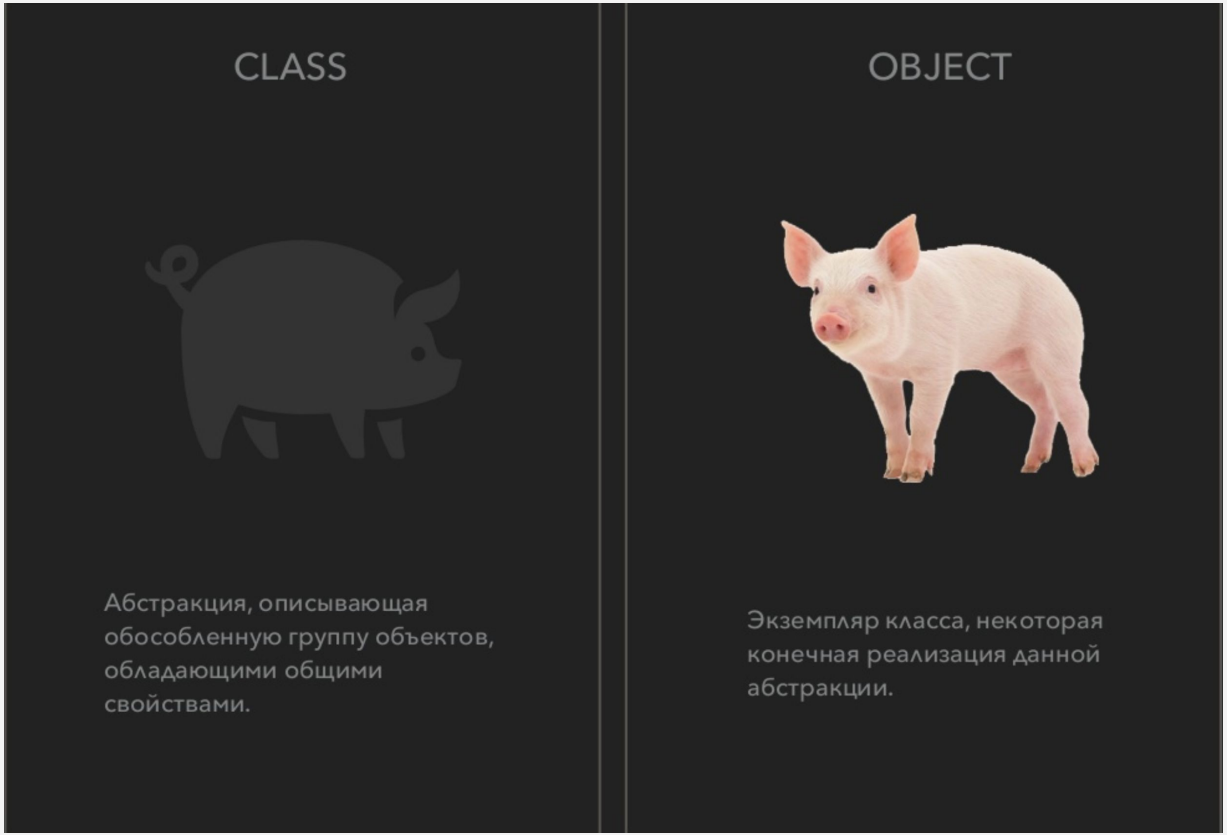
Как съесть слона ?



Абстракция



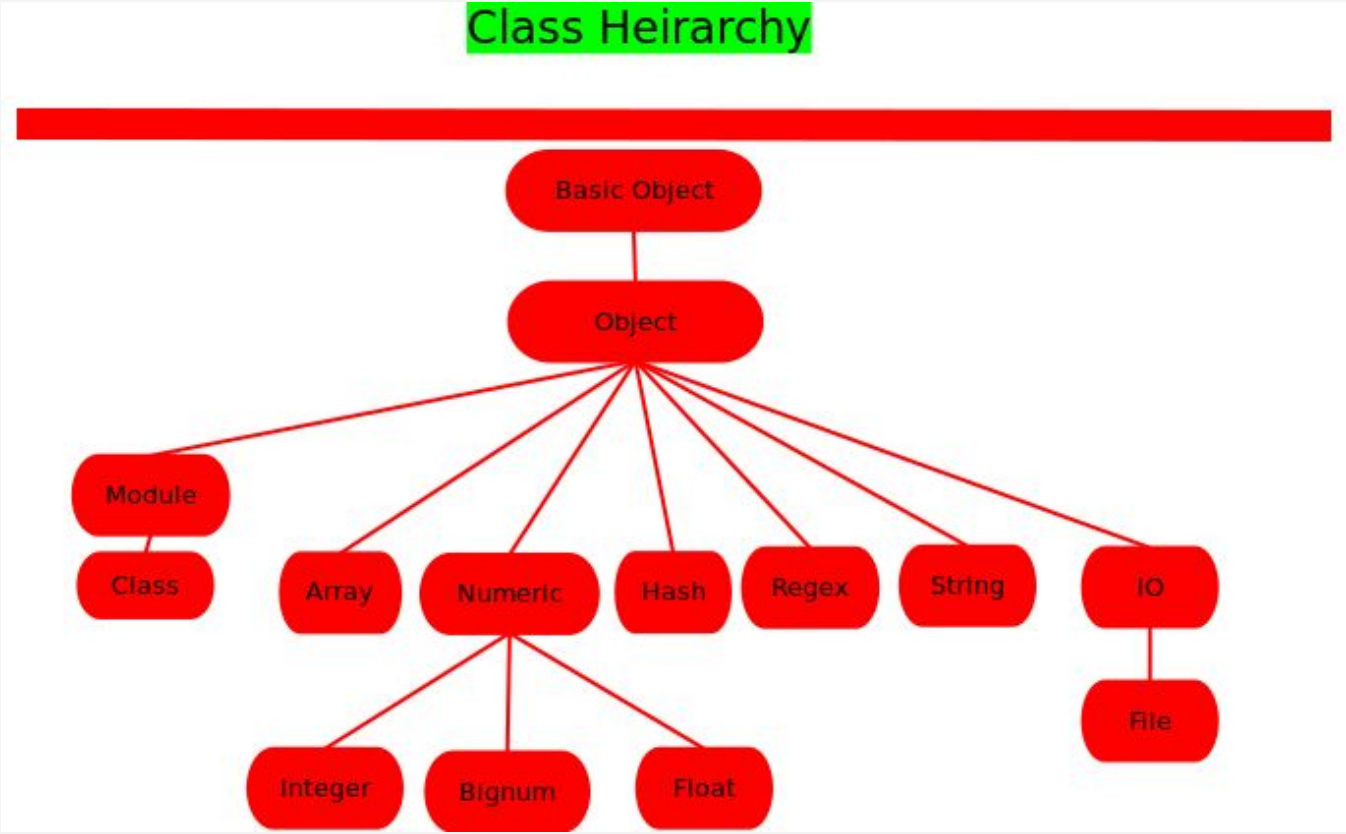
Объект и класс



Объект

```
2.6.0 :033 > object = Object.new
 => #<Object:0x00007fece01e3898>
2.6.0 :034 > object.object_id
 => 70327674543180
2.6.0 :035 > object.class
 => Object
2.6.0 :036 > object.class.ancestors
 => [Object, Kernel, BasicObject]
2.6.0 :037 > object.methods
 => [:instance_variable_defined?, :remove_instance_variable, :instance_of?, :kind_of?, :is_a?, :tap, :instance_variable_set, :protected_
methods, :instance_variables, :instance_variable_get, :public_methods, :private_methods, :method, :public_method, :public_send, :singlet
on_method, :define_singleton_method, :extend, :to_enum, :enum_for, :<=>, :===, :=~, :!~, :eql?, :respond_to?, :freeze, :inspect, :object
_id, :send, :to_s, :display, :nil?, :hash, :class, :singleton_class, :clone, :dup, :itself, :yield_self, :then, :taint, :tainted?, :unta
int, :untrust, :untrusted?, :trust, :frozen?, :methods, :singleton_methods, :equal?, :!, :==, :instance_exec, :!=, :instance_eval, :__id
__, :__send__]
```

Иерархия классов



Класс и объект

```ruby
class Foo
 def bar
   "It's me, bar !"
 end
end

object = Foo.new
object.bar # It's me, bar !
```

Методы класса

```ruby
class Foo
 @@class_variable = "Class variable"
 @class_instance_variable = "Class instance variable"

 def self.class_variable
   @@class_variable
 end

 def self.class_isntance_variable
   @class_instance_variable
 end
end

class Bar < Foo
 @@class_variable = "Bar class variable"
 @class_instance_variable = "Bar class instance variable"
end
```

```ruby
Foo.class_variable # Class variable
Foo.class_instance_variable # Class instance variable
```

```ruby
Bar.class_variable # Bar class variable
Bar.class_instance_variable # Bar class instance variable
```

Разница между переменными класса и инстанса класса

```ruby
class Foo
 @class_instance_variable = "Class instance variable"
 @@class_variable = "Class variable"

 def self.class_method
   puts @class_instance_variable
   puts @@class_variable
 end

 def instance_method
   puts @class_instance_variable
   puts @@class_variable
 end
end
```

```ruby
Foo.class_method
# Class instance variable
# Class variable

obj = Foo.new

obj.instance_method
# nil
# Class variable

class Bar < Foo
end

Bar.class_method
# nil
# Class variable
```

Методы экземпляра

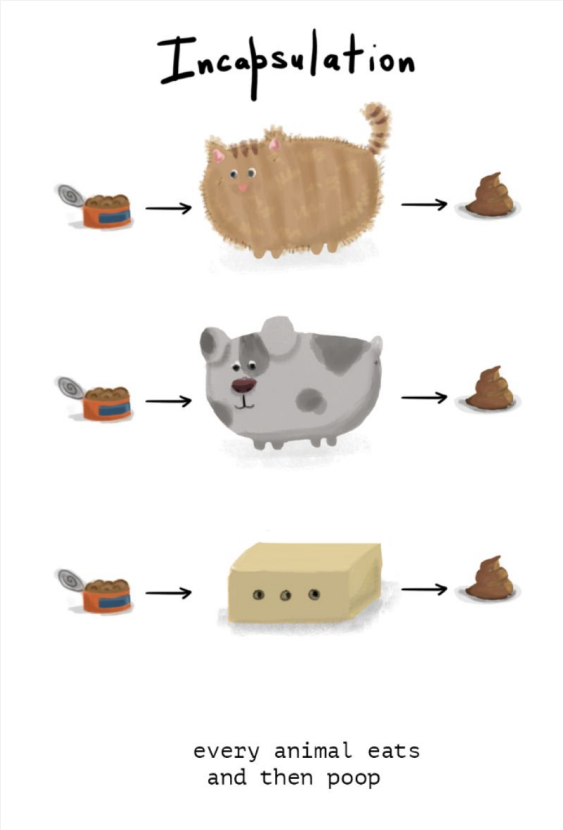
```ruby
class Foo
 attr_accessor :first

 def initialize(first, second)
   @first = first
   @second = second
 end

 def second
   @second
 end

 def self.class_method
   "hello"
 end
end
```

```ruby
foo = Foo.new(1, 2)
foo.first # 1
foo.second # 2
foo.class_method # NameError (undefined local variable or method `foo' for main:Object)
Foo.class_method # hello
```

Синглтон методы

```ruby
class Foo
end

foo_1 = Foo.new
foo_2 = Foo.new

def foo_1.new_singleton_method
 'foo_1 new method'
end

foo_1.new_singleton_method
# 'foo_1 new method'
foo_2.new_singleton_method
# NoMethodError (undefined method `new_singleton_method' for
#<Foo:0x00007fdfd891d290>)
```

Инкапсуляция

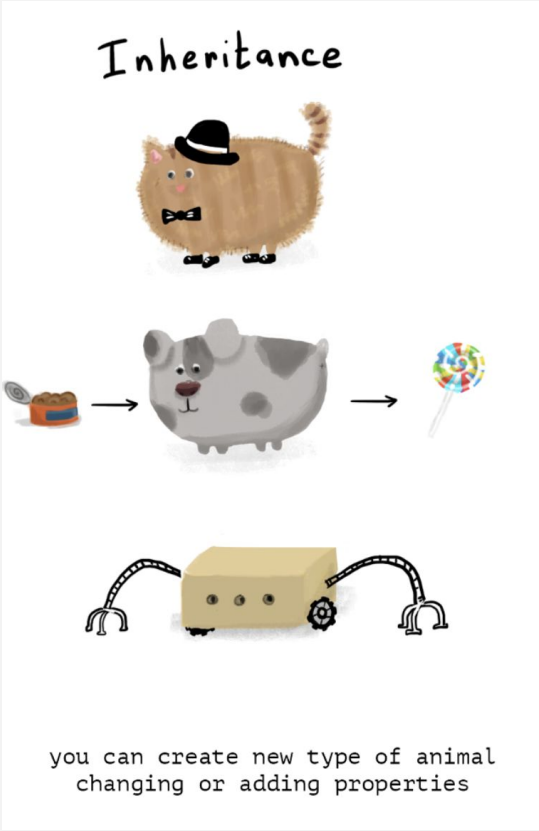


Инапсуляция

```ruby
class Welcome
 attr_accessor :name

 def initialize(name)
   @name = name
 end

 def message
   hello + glade
 end

 private

 def hello
   "Hello #{name} "
 end

 def glade
   "glad to see you !"
 end
end
```

```ruby
welcome = Welcome.new("Bob")
welcome.message # "Hello Bob glad to see you !"
```

Наследование

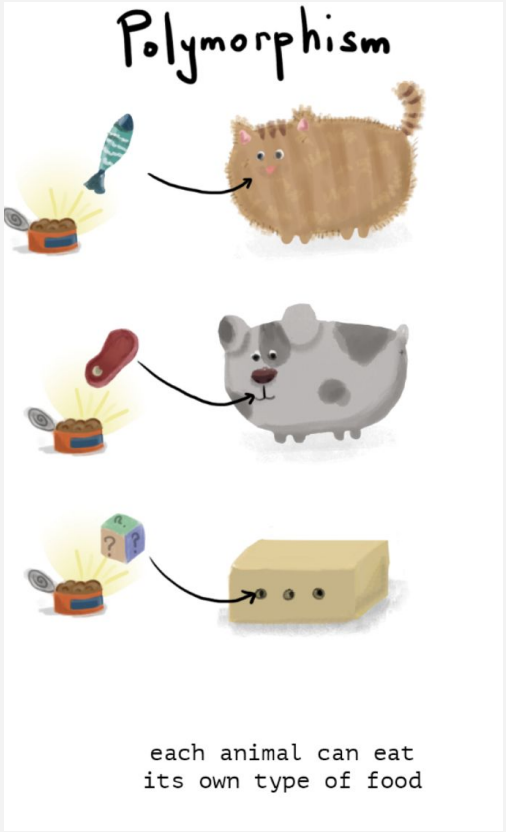


Наследование

```ruby
class Welcome
 attr_accessor :name

 def initialize(name)
   @name = name
 end

 def message
   hello + glade
 end

 private

 def hello
   "Hello #{name} "
 end

 def glade
   "glad to see you !"
 end
end
```

```ruby
class WelcomeRus < Welcome
 private

 def hello
   "Привет #{name} "
 end

 def glade
   "рад тебя видеть !"
 end
end

welcome = WelcomeRus.new("Иван")
welcome.message # "Привет Иван рад тебя видеть !"
```

Полиморфизм



Полиморфизм

```ruby
class GenericParser
 def parse
   raise NotImplementedError, 'You must implement the parse method'
 end
end

class JsonParser < GenericParser
 def parse
   puts 'An instance of the JsonParser class received the parse message'
 end
end

class XmlParser < GenericParser
 def parse
   puts 'An instance of the XmlParser class received the parse message'
 end
end
```

```ruby
puts 'Using the XmlParser'
parser = XmlParser.new
parser.parse

# Using the XmlParser
# An instance of the XmlParser class received the parse message

puts 'Using the JsonParser'
parser = JsonParser.new
parser.parse

# Using the JsonParser
# An instance of the JsonParser class received the parse message
```

Утиная типизация

Утина типизация

```ruby
class XmlParser
 def parse
   puts 'An instance of the XmlParser class received the
parse message'
 end
end

class JsonParser
 def parse
   puts 'An instance of the JsonParser class received the
parse message'
 end
end

class GenericParser
 def parse(parser)
   parser.parse
 end
end
```

```ruby
parser = GenericParser.new

puts 'Using the XmlParser'
parser.parse(XmlParser.new)

# Using the XmlParser
# An instance of the XmlParser class received the parse
message

puts 'Using the JsonParser'
parser.parse(JsonParser.new)

# Using the JsonParser
# An instance of the JsonParser class received the parse
message
```

Что почитать ?

1. http://nashbridges.me/introducing-ruby-oop
2. Главы 7.1 - 7.4 книги “Язык программирования Ruby”
3. http://internetka.in.ua/orm-intro/

Yalantis

Спасибо!

Остались вопросы? Буду рад вам ответить. Не забывайте пользоваться учебным чатом