

# Метапрограммирование

- Что такое метапрограммирование?
- Зачем оно нужно?
- Как правильно использовать метапрограммирование в Руби?

# Метапрограммирование. Что это?

- Мета- (с греч. μετά- «между, после, через») - часть слова обозначающая абстрагированность, обобщенность, превращение
- В программировании - изменение программой своего устройства (методы, переменные, интерфейсы классов и объектов) в рантайме

# Setters and getters (Java)

```
public class Cat {  
  
    private String name;  
    private int age;  
    private int weight;  
  
    public Cat(String name, int age, int weight) {  
        this.name = name;  
        this.age = age;  
        this.weight = weight;  
    }  
  
    public Cat() {  
    }  
  
    public String getName() {  
        return name;  
    }  
  
    public void setName(String name) {  
        this.name = name;  
    }  
  
    public int getAge() {  
        return age;  
    }  
  
    public void setAge(int age) {  
        this.age = age;  
    }  
  
    public int getWeight() {  
        return weight;  
    }  
  
    public void setWeight(int weight) {  
        this.weight = weight;  
    }  
}
```

## Setters and getters (Ruby)

```
class Cat

  def initialize name, age, weight
    @name, @age, @weight = name, age, weight
  end

  def name; @name; end
  def age; @age; end
  def weight; @weight; end

  def name= name
    @name = name
  end

  def age= age
    @age = age
  end

  def weight= weight
    @weight = weight
  end

end
```

```
class Cat

  attr_writer :name, :age, :weight

  def initialize name, age, weight
    @name, @age, @weight = name, age, weight
  end

  def name; @name; end
  def age; @age; end
  def weight; @weight; end

end
```

```
class Cat

  attr_accessor :name, :age, :weight

  def initialize name, age, weight
    @name, @age, @weight = name, age, weight
  end

end
```

```
class Cat

  def self.my_attr_accessor *attributes
    attributes.each do |attribute|
      define_method "#{attribute}" do
        self.instance_variable_get "@#{attribute}"
      end
      define_method "#{attribute}=" do |value|
        self.instance_variable_set "@#{attribute}", value
      end
    end
  end

  my_attr_accessor :name, :age, :weight

  def initialize name, age, weight
    @name, @age, @weight = name, age, weight
  end

end
```

## Monkey patching

```
class Array

  def how_many?
    length
  end

end

%w(one two three).how_many?
=> 3
```



# Зачем:

- DRY
- DSL
- Гибкость
- Декомпозиция
- Повышение уровня абстракции (Racc, Ragel)

# Почему Ruby хорош для метапрограммирования?

- Monkey patching
- Лямбды
- Миксины (Include, Prepend, Extend)
- Методы интроспекции классов и объектов
- Методы метапрограммирования классов и объектов

# Интроспекция объектов

- `methods`
- `public_methods`
- `private_methods`
- `singleton_methods`
- `respond_to?`
- `class`
- `instance_of?`
- `instance_variables`
- `instance_variable_get`

# Интроспекция классов

- `methods`
- `instance_methods`
- `public_methods`
- `private_methods`
- `class_variables`
- `ancestors`
- `include?`
- `included_modules`
- `class_variable_defined?`
- `const_defined?`

# Интроспекция классов

- `methods`
- `instance_methods`
- `public_methods`
- `private_methods`
- `class_variables`
- `ancestors`
- `include?`
- `included_modules`
- `class_variable_defined?`
- `const_defined?`

# Метапрограммирование объектов

- send
- method\_missing
- instance\_eval
- remove\_instance\_variable
- extend
- define\_singleton\_method
- instance\_variable\_set

# Метапрограммирование классов

- `define_method`
- `include`
- `extend`
- `const_missing`
- `class_eval`
- `class_variable`
- `class_variable_get`

# Стрельба по ногам

```
class Cat

  attr_accessor :name

  def initialize string
    name = string
  end

end

cat = Cat.new("КиТ")
cat.name # => nil
```



# HOMework:

hexlet program download rails metaprogramming