Parsing Ruby

Rathesan Iyadurai

Goal

Extract all classes and their methods without having to define the complete grammar of Ruby.

Previously on “Parsing Ruby”

```ruby
class Dog

  attr_accessor :age

  def initialize(age)
    raise "NOO" if age < 1

    @name = "foo class bar"
    @age  = age
  end

  def bark
    if age > 3
      puts "wuff"
    else
      puts "wiff"
    end
  end

end
```

```ruby
class Dog

  attr_accessor :age

  def initialize(age)
    raise "NOO" if age < 1

    @name = "foo class bar"
    @age  = age
  end

  def bark
    if age > 3
      puts "wuff"
    else
      puts "wiff"
    end
  end

end
```

```ruby
class Dog

  attr_accessor :age

  def initialize(age)
    raise "NOO" if age < 1

    @name = "foo class bar"
    @age  = age
  end

  def bark
    if age > 3
      puts "wuff"
    else
      puts "wiff"
    end
  end

end
```

```ruby
class Dog

  attr_accessor :age

  def initialize(age)
    raise "NOO" if age < 1

    @name = "foo class bar"
    @age  = age
  end

  def bark
    if age > 3
      puts "wuff"
    else
      puts "wiff"
    end
  end

end
```

```ruby
class Dog

  attr_accessor :age

  def initialize(age)
    raise "NOO" if age < 1

    @name = "foo class bar"
    @age  = age
  end

  def bark
    if age > 3
      puts "wuff"
    else
      puts "wiff"
    end
  end

end
```

```
'foo class bar'
"foo class bar"

%?foo class bar?
%{foo class bar}
%<foo class bar>
%(foo class bar)
%[foo class bar]
%q{foo class bar}
%Q{foo class bar}

<<spongebob.strip
foo class bar
spongebob
```

Modifiers

```ruby
class Dog

  attr_accessor :age

  def initialize(age)
    raise "NOO" if age < 1

    @name = "foo class bar"
    @age  = age
  end

  def bark
    if age > 3
      puts "wuff"
    else
      puts "wiff"
    end
  end

end
```

```ruby
class Dog

  attr_accessor :age

  def initialize(age)
    raise "NOO" if age < 1

    @name = "foo class bar"
    @age  = age
  end

  def bark
    if age > 3
      puts "wuff"
    else
      puts "wiff"
    end
  end

end
```

```ruby
class Dog

  attr_accessor :age

  def initialize(age)
    raise "NOO" if age < 1

    @name = "foo class bar"
    @age  = age
  end

  def bark
    str = if age > 3
      "wuff"
    else
      "wiff"
    end
    puts str
  end

end
```

```ruby
class Dog

  attr_accessor :age

  def initialize(age)
    raise "NOO" if age < 1

    @name = "foo class bar"
    @age  = age
  end

  def bark
    str = if age > 3
      "wuff"
    else
      "wiff"
    end
    puts str
  end

end
```

```
modifier :=
  startOfLine ,
  stuffToConsume ,
  newline not ,
  '=' not ,
  'if'
```

```
class Dog

  attr_accessor :age

  def initialize(age)
    raise "NOO" if age < 1

    @name = "foo class bar"
    @age  = age
  end

  def bark
    str = if age > 3
      "wuff"
    else
      "wiff"
    end
    puts str
  end

end
```

```
modifier :=
  startOfLine ,
  stuffToConsume ,
  newline not ,
  '=' not ,
  'if'
```

```
class Dog

  attr_accessor :age

  def initialize(age)
    raise "NOO" if age < 1

    @name = "foo class bar"
    @age  = age
  end

  def bark
    str = if age > 3
      "wuff"
    else
      "wiff"
    end
    puts str
  end

end
```

```
modifier :=
  startOfLine ,
  stuffToConsume ,
  newline not ,
  '=' not ,
  'if'
```

```
class Dog

  attr_accessor :age

  def initialize(age)
    raise "NOO" if age < 1

    @name = "foo class bar"
    @age  = age
  end

  def bark
    str = if age > 3
      "wuff"
    else
      "wiff"
    end
    puts str
  end

end
```

```
modifier :=
  startOfLine ,
  stuffToConsume ,
  newline not ,
  '=' not ,
  'if'
```

```
class Dog

  attr_accessor :age

  def initialize(age)
    raise "NOO" if age < 1

    @name = "foo class bar"
    @age  = age
  end

  def bark
    str = if age > 3
        "wuff"
      else
        "wiff"
      end
      puts str
  end

end
```

~ 50 rules

10 for matching “end”s

13 for keywords

10 for strings and comments

Keywords

```
def if
   endless
   bado
end.class
```

Keywords

```
def if
    endless
    bado
end.class
```

String Interpolation

```
name = "Jan"
"Hi #{name}!" # => "Hi Jan!"

"Hi #{class Dog; end}"

"Hi #{puts ""}"

"Hi #{list.each { |l| foo }}"
```

PetitParser

VS

https://github.com/jruby/jruby-parser

```
class Dog
end if foo?
```

Now what?

```ruby
def foo
  a = "i'm a local variable!"
  b
end

def b
  "i'm a method!"
end
```

```
def foo(arg)
   a = "i'm a local variable!"
   arg.b
end
```

Overview

Goal: Extract all classes and their methods without having to define the complete grammar of Ruby

Problems: modifiers, keywords, string interpolation

Goal achieved?

RubyGrammar Performance

smooth milliseconds per character count

Milliseconds

3000
2500
2000
1500
1000
500
0

0
5000
10000
15000
20000
25000
30000
35000
40000
45000

Number of Characters

RubyGrammar Performance

smooth milliseconds per method count

Milliseconds

3000
2500
2000
1500
1000
500
0

0
20
40
60
80
100
120

Number of Methods