Parsing Ruby

Rathesan Iyadurai

Goal

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

Ruby

a dynamic, reflective, object-oriented, general-purpose programming language.

https://www.ruby-lang.org/en/

```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
```

PetitParser

```
identifier := #letter asParser , #word asParser star.
identifier parse: 'hello2'.   -> (h (e l l o 2))
identifier parse: '123'.      -> letter expected 0
```

```
body      := ' blabla ' asParser.
classDef := 'class' asParser , body , 'end' asParser.
classDef parse: 'class blabla end' -> ('class' ' blabla ' 'end')
```

```
method :=
  'def' ,
  identifier ,
  program ,
  'end'

class :=
  'class' ,
  identifier ,
  program ,
  'end'

program :=
  anything ,
  ((class / method) ,
     anything) star
```

```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
```

```
anything :=
  (class / method / 'end')
  negate star
```

```
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
```

The “end”

```
method :=
  'def' ,
  identifier ,
  program ,
  'end'

class :=
  'class' ,
  identifier ,
  program ,
  'end'

program :=
  anything ,
  ((class / method) ,
    anything) star

anything :=
  (class / method / 'end')
  negate star
```

```
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
```

```
method := …

class := …

if :=
 'if' ,
 program ,
 'end'

program :=
 anything ,
 ((class / method / if) ,
  anything) star

anything :=
 (class / method / if / 'end')
 negate star
```

```
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
```

```
program :=
  anything ,
  ((class /
     method /
     module /
     eigenClass /
   forLoop /
   begin /
   modifier /
   conditional /
   doBlock /
   … ) ,
 anything) star
```

String Literals

```
method :=
  'def' ,
  identifier ,
  program ,
  'end'

class :=
  'class' ,
  identifier ,
  program ,
  'end'
```

```
if :=
  'if' ,
  program ,
  'end'
```

```
program := …

anything := …
```

```
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

```
method :=
  'def' ,
  identifier ,
  program ,
  'end'

class :=
  'class' ,
  identifier ,
  program ,
  'end'

if :=
  'if' ,
  program ,
  'end'


program := …

anything := …
```

```
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
```

```
method :=
  'def' ,
  identifier ,
  program ,
  'end'

class :=
  'class' ,
  identifier ,
  program ,
  'end'

if :=
  'if' ,
  program ,
  'end'


program := …

anything := …
```

```
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
```

“I believe people want to express themselves when they program. They don't want to fight with the language. Programming languages must feel natural to programmers. I tried to make people enjoy programming and concentrate on the fun and creative part of programming when they use Ruby”

–Yukihiro Matsumoto

Overview

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

Problems: nested “end”s, string literals and modifiers

Next steps: Find solution for modifiers, Case Study