Metaprogramming

Concepts of Programming Languages

Alexander Schramm

Institut für Softwaretechnik und Programmiersprachen

2. November 2015

Table of Contents

Outline

Motivation

Which issues do we want to tackle?

- Avoid writing boilerplate code
- Write code that shows our intention
- Expand the syntax of languages
- Write type independent code in strongly typed languages

What is Metaprogramming

Definition: Metaprogramming

Metaprograming describes different ways to generate and manipulate code

Differentations:

- Compile time vs. runtime metaprogramming
- Domain language vs. host language

Differentations of Metaprograms

Compile time Metaprogramming

Ways to manipulate or generate code during compilation, e.g: Macros, Templates

Runtime Metaprogramming

Ways to manipulate or generate code, while it is executed, e.g: dynamic methods, Reflections

Differentations of Metaprograms

Domain Language

The Programming Language, in which the metaprogram is written

Host Language

The Programming Language of the generated code

- Can be different (YACC, Compilers)
- Domain language can be a subset of the host language (C++ Templates)
- Domain language can be an integral part of the host language (Ruby)

Outline

Runtime Reflection in Java

What is Reflection?

- Get metadata about an object at runtime
  - What is its class
  - Which methods does it respond to

Example: The Class object

```
Class<Date> c1 = java.util.Date.class;
System.out.println( c1 ); // class java.util.Date

for (Method method : c1.getMethods()){
  System.out.println(method.getName())
}
```

Usage of Reflections

Reflection is used by the JUnit test framework to find test methods.

Example: Test case parser

```
public void parse(Class<?> clazz) {
  Method[] methods = clazz.getMethods();
  for (Method m : methods) {
    if (m.isAnnotationPresent(Test.class)) {
      m.invoke(null);
    }
  }
}
```

Runtime Reflection in Java

The Class object:

- Represents metadata about a class at runtime
- Metadata can be added by annotations (@RunWith(. . .))

Conclusion:

- Not really metaprogramming (no code manipulation happening)
- Example of a runtime object model (more in a second)
- Bad performance!

Outline

Runtime Metaprogramming in Ruby

What is Ruby:

- dynamic, interpreted high level language
- has a rich, accessible runtime object model
- depends on metaprogramming techniques

Usage of an object model:

- In most languages most information about structure is lost after compilation
- The object model represents this structure at runtime
- Rubys object model can be manipulated

The Ruby interpreter

How does the Ruby interpreter work?

- Uses the object model to evaluate code
- Therefore manipulation of the object model manipulates the program

Example: Manipulating code at runtime

```
class Test
  def show; puts "a"; end
  def self.redefine
    define_method(:show){puts "b"}
  end
end

t = Test.new
t.show # => "a"
Test.redefine
t.show # => "b"
```

The Ruby Object Model

How is the object model structured?

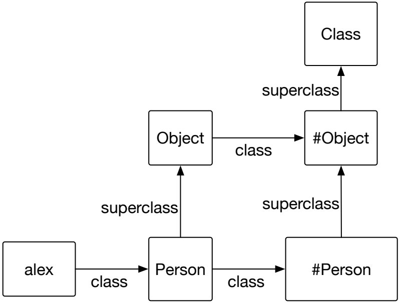
- Every class/module has a corresponding object
- Every instance of a class has an object
- Methods *live* in the class of the object
- Many language constructs have an object
- Every object has a class (and most times a singleton class)
- What is the class of a class object?

The Ruby Object Model

How is the object model structured?

- Every class/module has a corresponding object
- Every instance of a class has an object
- Methods *live* in the class of the object
- Many language constructs have an object
- Every object has a class (and most times a singleton class)
- What is the class of a class object?
  - A class object has the class `Class`
  - Class methods *live* in the singleton/eigenclass of the class object

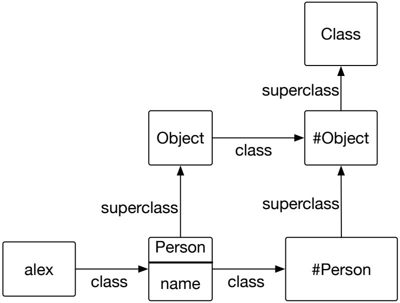
The Ruby Object Model




```ruby
class Person
  def name;
  end
end

alex = Person.new
```

The Ruby Object Model

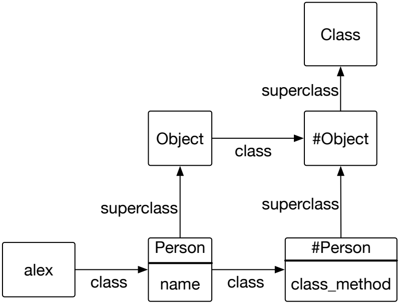



```ruby
class Person
  def name;
  end

  def self.class_method
  end
end

alex = Person.new
```

The Ruby Object Model



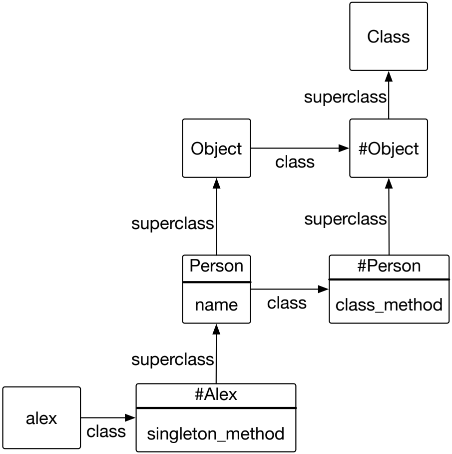

```
class Person
  def name;
  end

  def self.class_method
  end
end

alex = Person.new

class << alex
  def singleton_method
  end
end
```

The Ruby Object Model

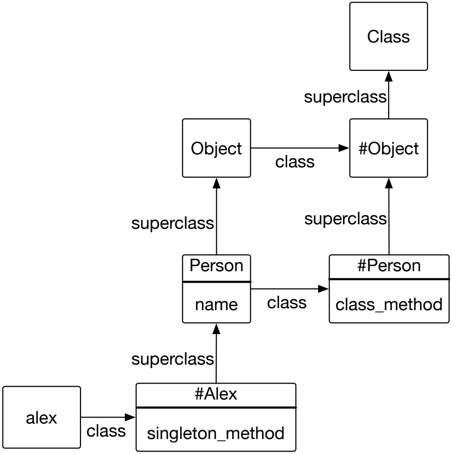



```ruby
class Person
  def name;
  end

  def self.class_method
  end
end

alex = Person.new
class << alex
  def singleton_method
  end
end
```

Method Lookup



1. Call `obj.method`
2. Go one step right
3. Use method if defined, else go one up
4. Repeat step 3 until the method is found
5. Call `obj.method_missing('method')`

Method Missing example

One could use `method_missing('method')` to implement methods.
JBuilder does this for Json generation:

Example: Using JBuilder

```
json.firstName "John"
json.lastName  "Smith"
json.age 25
json.children(@children) do |child|
  json.name child.name
end
```

Further Usages

What else can be done?

- Define classes at runtime: `newClass = Class.new do ... end`
- Alias methods
- Remove methods
- Evaluate strings as code
- Hook into runtime events: `included`, `method_added`, `inherited`, . . .

Outline

C++ Templates

- Templates are a compile time mechanism to define type independent code
- How: Definition of a *Template*, which will generate a method with appropriate types when the template is used

Example: C++ Template

```
template <typename T>
T max(T x, T y)
{
  if (x < y)
    return y;
  else
    return x;
}
```

C++ Templates

```
template <typename T>
T max(T x, T y)
{
  if (x < y)
    return y;
  else
    return x;
}
```

What will be generated?

- `max(1,2)`

C++ Templates

```
template <typename T>
T max(T x, T y)
{
  if (x < y)
    return y;
  else
    return x;
}
```

What will be generated?

- `max(1,2)`
  - `int max(int a, int b)`

C++ Templates

```
template <typename T>
T max(T x, T y)
{
  if (x < y)
    return y;
  else
    return x;
}
```

What will be generated?

- `max(1,2)`
  - `int max(int a, int b)`
- `max("a","b")`

C++ Templates

```
template <typename T>
T max(T x, T y)
{
  if (x < y)
    return y;
  else
    return x;
}
```

What will be generated?

- `max(1,2)`
  - `int max(int a, int b)`
- `max("a","b")`
  - `string max(string a, string b)`

C++ Templates

```
template <typename T>
T max(T x, T y)
{
  if (x < y)
    return y;
  else
    return x;
}
```

What will be generated?

- `max(1,2)`
  - `int max(int a, int b)`
- `max("a","b")`
  - `string max(string a, string b)`
- `max(1,"a")`

C++ Templates

```
template <typename T>
T max(T x, T y)
{
  if (x < y)
    return y;
  else
    return x;
}
```

What will be generated?

- `max(1,2)`
  - `int max(int a, int b)`
- `max("a","b")`
  - `string max(string a, string b)`
- `max(1,"a")`
  - No such function Error

C++ Templates

- Templates are actually a turing complete, functional language
- Everything is immutable
- Therefore no loops
- But: Recursion with template specialization

Example: Template specialization

```
template <unsigned n, bool done = (n < 2)>
struct fibonacci {
  static unsigned const value =
    fibonacci<n-1>::value +
    fibonacci<n-2>::value;
}
template <unsigned n>
struct fibonacci<n, true> {
  static unsigned const value = n;
}
```

Template Specialization

Template specialization can be used for more:

- Generic implementation for most types, but specialised for specific types
  - Vector template as array of the type
  - But not for booleans, because of space (16-32 bit)
- Partial specialization vs. full specialization

Example: Template specialisation for performance

```
template <typename T>
class vector{
  T* vec_data;
}

template <>
class vector <bool>{
  unsigned int* vec_data;
}
```

Outline

Haskell Templates

What is Haskell?

- Statically typed, purely functional language
- Template mechanism similar to C++
- Access to the abstract syntax tree (AST)

How can we use that?

- Write constructs which imitate language level syntax
- Write domain specific languages
- Extend the language
- Adapt the language to a problem domain

Haskell Templates

Example: Typesafe println macro

```
intToString :: Integer -> String

data Format = Int | Str | Lit String

parse :: String -> [Format]
parse "" = []
parse ('%' : 'i' : rest) = Int : parse rest
parse ('%' : 's' : rest) = Str : parse rest
parse (c:str) = Lit c : parse rest

gen :: [Format] -> Exp -> Exp
gen []           acc = acc
gen (Int   : xs) acc = [| \n -> $(gen xs [| $acc ++ intToString n |]) |]
gen (Str   : xs) acc = [| \s -> $(gen xs [| $acc ++ s |]) |]
gen (Lit s : xs) acc = gen xs [| $acc ++ $(stringE s) |]

sprintf :: String -> Exp
sprintf str = gen (parse str) [| "" |]

-- $(sprintf "Error: %s on line %d") msg line generates:
-- (\s_0 -> \n_1 -> "" ++ "Error: " ++ s_0 ++ " on line " ++ intToString
 ↪  n_1) msg line
```

Outline

Lisp

The Lisp programming language

- One of the oldest programming languages still in use
- Many implementations: Clojure, Common Lisp, Scheme ...
- Very simple, straightforward syntax: S-Expressions

An S-Expression is either

- an atom (a identifier) or
- in the form `(a b)` where `a` and `b` are S-Expressions
- The first member of the list is treated as a method call, the rest as its arguments

Example: Lisp Syntax

```
(list 1 2 (list 3 4))
(+ 3 4 5)
(set x (list 3 4))
```

Macros

What are macros

- Construct and manipulate the AST
- They look very similar to normal methods
- They are actually called exactly like normal methods

Code as data

- Lets look at the valid Lisp program `(+ 2 3 4)`
- It's a call to the + method with the argument 2, 3 and 4
- At the same time it's a list of the 4 atoms +, 2, 3 and 4
- Data can be manipulated, code is data, therefore code can be manipulated

Write Macros

Example: How to write a Macro

```
(defmacro unless (condition x y)
  `(if (not ~condition) ~x ~y)
)
```

- Arguments passed to a macro are not evaluated
  - Allow evaluation with `~(+ 2 3)`
- Macros should return valid Lisp code
  - Generate unevaluated lists with `'(a b c)`
  - Unevaluated list except macros `` `(a b c) ``

Use Macros

Example: How to use a macro

```
(unless (> a b) (set x a) (set x b))
```

- Macros are called just as normal functions
- Good for newcommers to Lisp: no knowledge of macros needed

Usecases of Macros

What can macros be used for?

- Extend the language with constructs that look like language level constructs
- Write domain specific languages
- Adapt the language to a specific problem
- Write more readable code
- Write more concise code

Outline

Conclusion

More Metaprogramming:

- Groovy language with runtime and compile time metaprogramming on the JVM
- Macros in Scala
- Macros in Elixir, a Ruby like, functional language

When to use Metaprogramming?

- Depends on the language
- Metaprogramming can lead to bad and good code
- Always evaluate all approaches to solve a problem

Conclusion

Questions?