Ruby "MetaClasses"

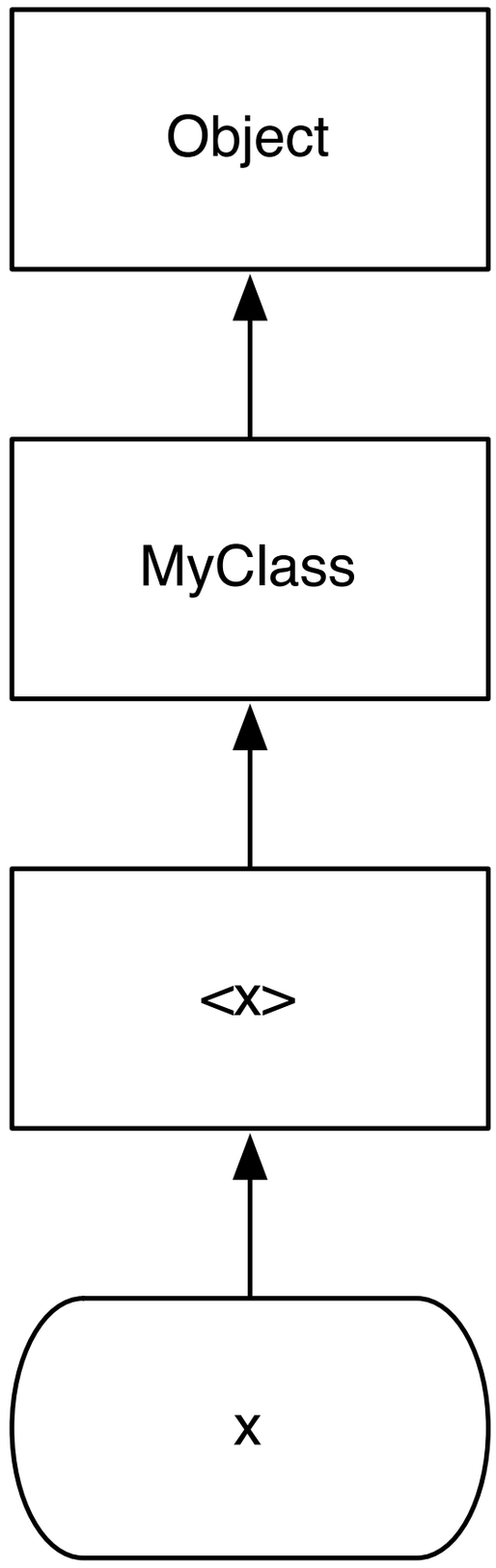
```
class X
  def y
    3.14
  end
end
```

```ruby
x = X.new

p x.y
# => 3.14

p x.class.ancestors
# => [X, Object, Kernel]
```

```ruby
def x.y
  42
end
```


```ruby
p x.y
# => 42
```

```
class Person
  def self.people
    @@people
  end
end
```



```
module Y
  def y
    "happy"
  end
end
```

```ruby
x = X.new
x.extend Y
p x.y
# => "happy"
p x.class.ancestors
# => [X, Object, Kernel]
```

```
class X
  include Y
end
```

```ruby
x = X.new
p x.y
# => 3.14
p x.class.ancestors
# => [X, Y, Object, Kernel]
```

Pros & Cons

+ #including a module is a clean alternative to multiple inheritance or interfaces.

+ #including a module respects the inheritence chain, allowing extension via super:

```
include GBSDebugSession
```

- Not having an open MOP means we can't #uninclude (w/o writing a C extension on Module).