Ruby master - Misc #13804

Protected methods cannot be overridden

08/10/2017 09:41 PM - davidarnold (David Arnold)

Status:	Open
Priority:	Normal
Assignee:	

Description

In Ruby, the main reason you would use protected instead of private is because you want different instances of the same class lineage to be able to access the method.

However, the rules around protected (callable only where self of the context is the same as the [...] method definition) means that protected method effectively cannot be overridden by subclasses. The redefinition of the method resets the protected access check to the subclass, so that instances of the parent class (or other subclasses) cannot call the method anymore.

Is the recommendation that using protected is just bad practice and should be avoided? Or is there a way to make the protected behavior aware of the parent method that is being overridden and keep the access check at the same level in the class hierarchy?

Example:

```
class Person
  def initialize(ssn)
    @ssn = ssn
  end

  def same?(other)
    tax_id == other.tax_id
  end

  protected
  def tax_id
    @ssn
  end
end

class HappyPerson < Person
  def shout
    'yay!'
  end
end

# corporations are people now
class Corporation < Person
  def initialize(ein)
    @ein = ein
  end

  protected
  def tax_id
    @ein
  end
end

bob = Person.new('000-00-0001')
sally = HappyPerson.new('000-00-0002')
acme = Corporation.new('00-0000001')

puts bob.same? bob   # true

puts bob.same? sally # false
puts sally.same? bob # false

puts acme.same? bob  # false
```

```
puts bob.same? acme  #=> protected method `tax_id' called ... (NoMethodError)
```