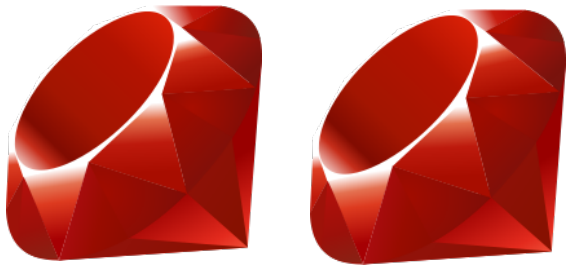

Access Control

Access Control

- When designing your class – important to think about how much of it you will be exposing to the world
- Encapsulation – try to hide the internal representation of the object. This way if you decide to change it later on – you can.
- 3 levels: `public`, `protected` and `private`

Access Control (Continued)

- `public` methods – no access control is enforced. Anybody can call these methods.
  - Default (except `initialize` which is `private`)
- `protected` methods – can be invoked by the objects of the defining class or its subclasses
- `private` methods – cannot be invoked with an explicit receiver
  - Cannot invoke another object's private methods

Specifying access control

- 2 ways to specify access control:
  1. Specify `public`, `protected` or `private`
     - Everything until the next access control keyword will be of that access control level
  2. Define the methods regularly and then specify `public`, `private`, `protected` access levels and list the comma-separated methods under those levels using method symbols

Access Control example

```
class MyAlgorithm
  private
    def test1
      "Private"
    end
  protected
    def test2
      "Protected"
    end
  public
    def public_again
      "Public"
    end
end
class Another
  def test1
    "Private, as declared later on"
  end
  private :test1
end
```

(Unlike Java) *private* access control does NOT allow you to invoke a method from a different instance of the same class