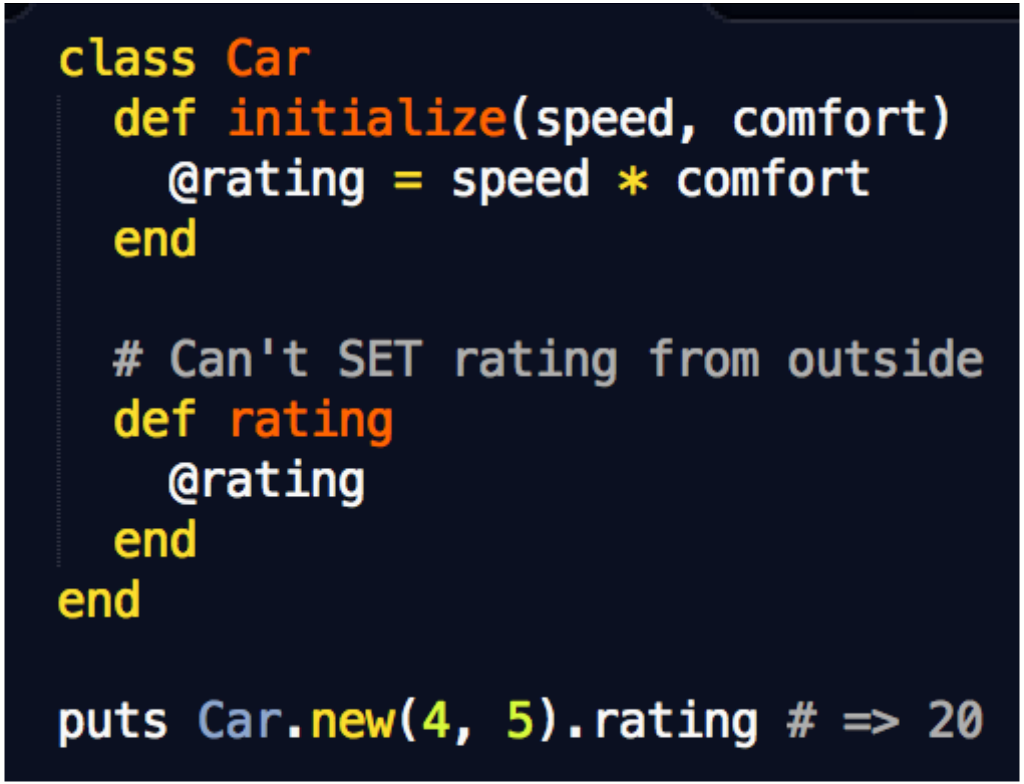
In this lecture, we will discuss…

- ✧ **Three levels** of access control
- ✧ **Controlling** access
- ✧ How **private** is private access?

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 so you **can change it** later
- **Three** levels: `public`, `protected` and `private`

Encapsulation



```
class Car
  def initialize(speed, comfort)
    @rating = speed * comfort
  end

  # Can't SET rating from outside
  def rating
    @rating
  end
end

puts Car.new(4, 5).rating # => 20
```

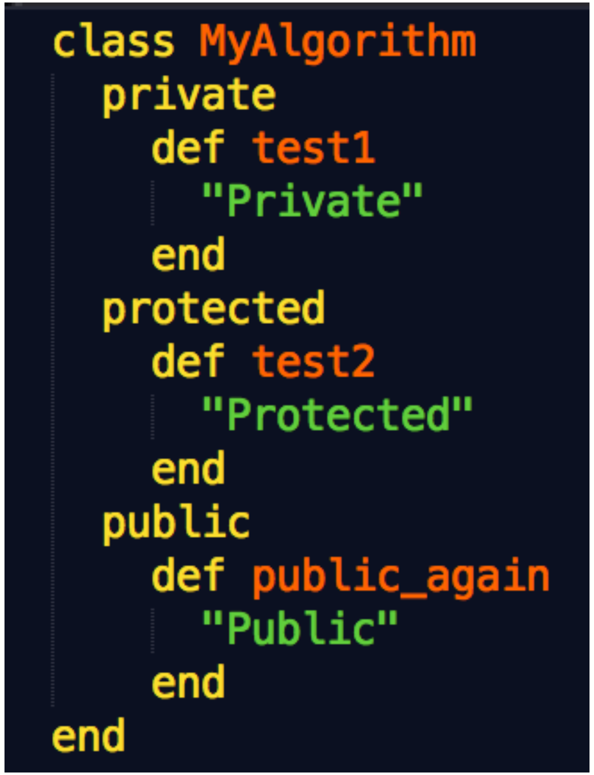
Details of how rating is calculated are kept inside the class

Specifying Access Control

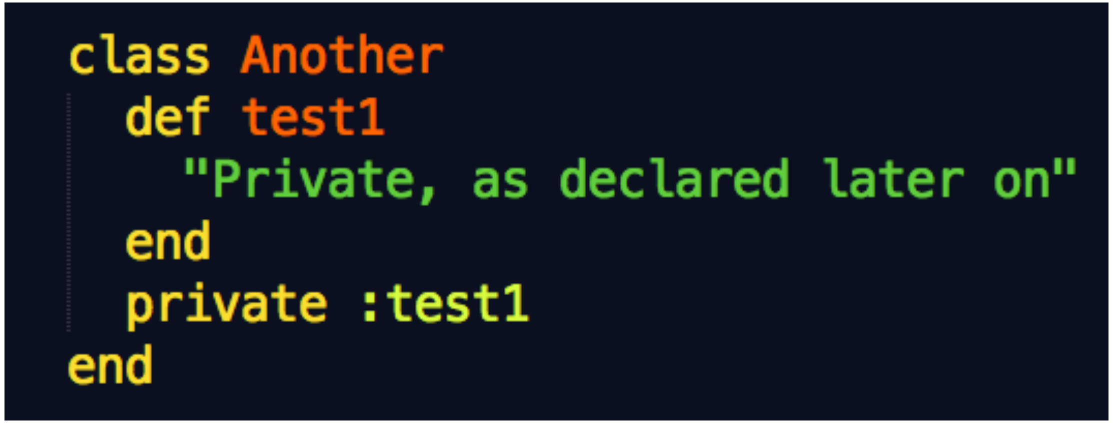
- ✧ **Two** ways to **specify** access control:

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**

Specifying Access Control



```
class MyAlgorithm
  private
    def test1
      "Private"
    end
  protected
    def test2
      "Protected"
    end
  public
    def public_again
      "Public"
    end
end
```

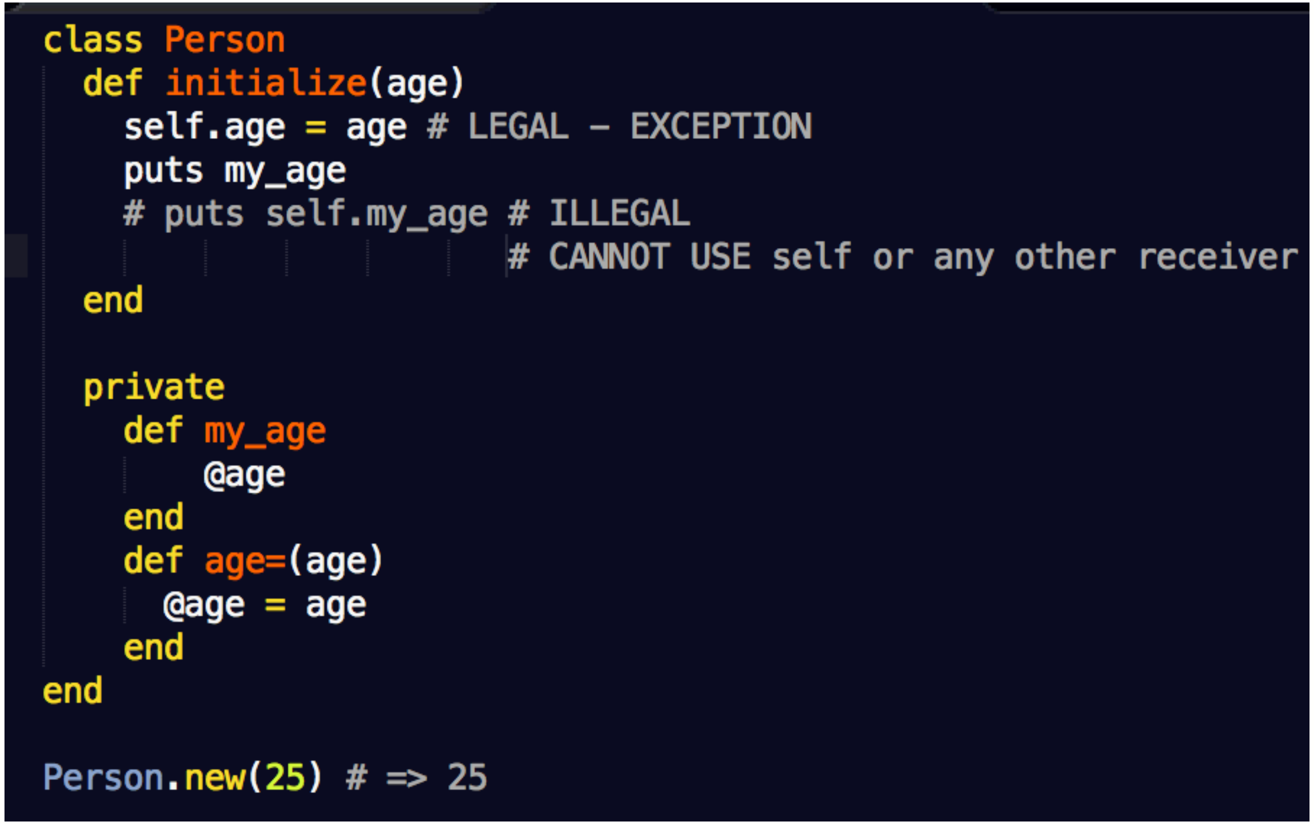


```
class Another
  def test1
    "Private, as declared later on"
  end
  private :test1
end
```

Access Control

- ✧ `public` methods – **no** access control is enforced
  - **Anybody** can call these methods
- ✧ `protected` methods – **can be invoked** by the objects of the defining class or its subclasses
- ✧ `private` methods – **cannot be invoked** with an explicit receiver
  - **Exception**: Setting an attribute can be invoked with an explicit receiver

Private Access



```
class Person
  def initialize(age)
    self.age = age # LEGAL - EXCEPTION
    puts my_age
    # puts self.my_age # ILLEGAL
                       # CANNOT USE self or any other receiver
  end

  private
    def my_age
        @age
    end
    def age=(age)
      @age = age
    end
end

Person.new(25) # => 25
```

Summary

- ✧ `public` and `private` access controls used the **most**
- ✧ `private` methods are **not callable** from outside or inside the class with an explicit receiver

What's next?

- Introduction to Unit Testing