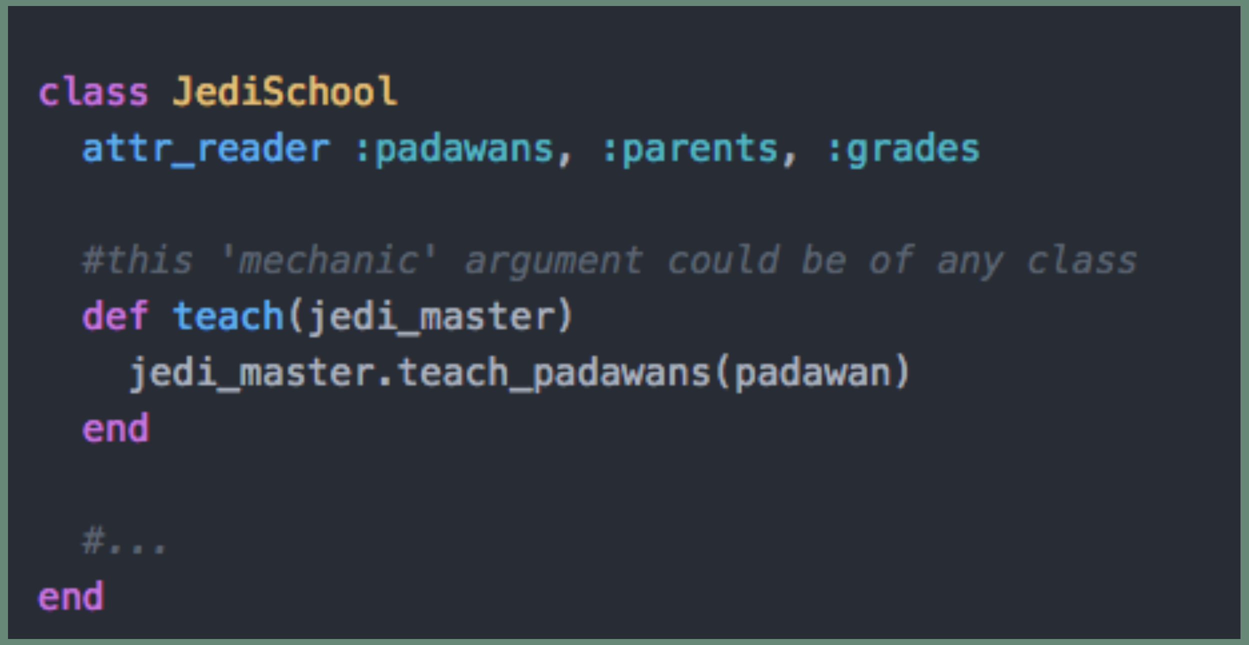
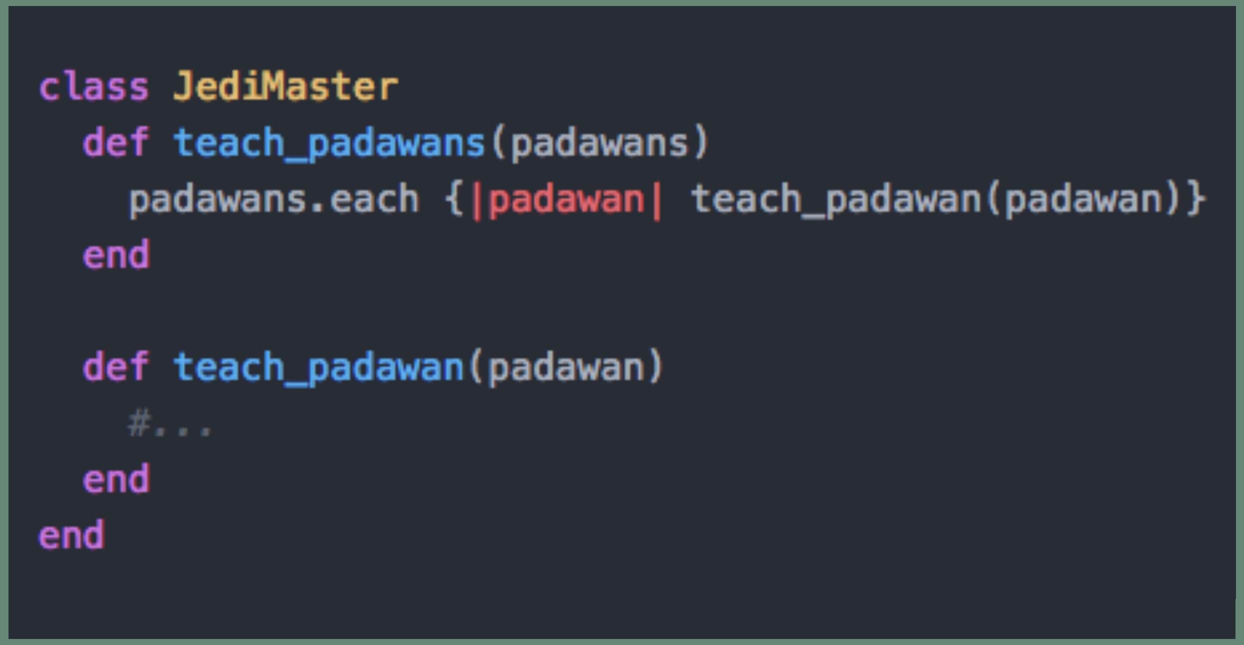
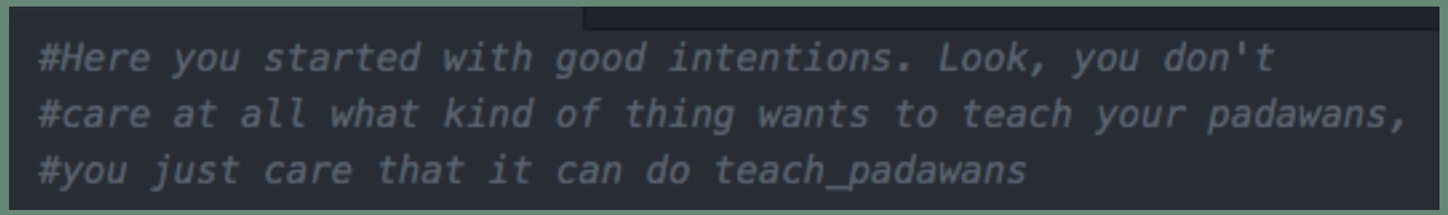


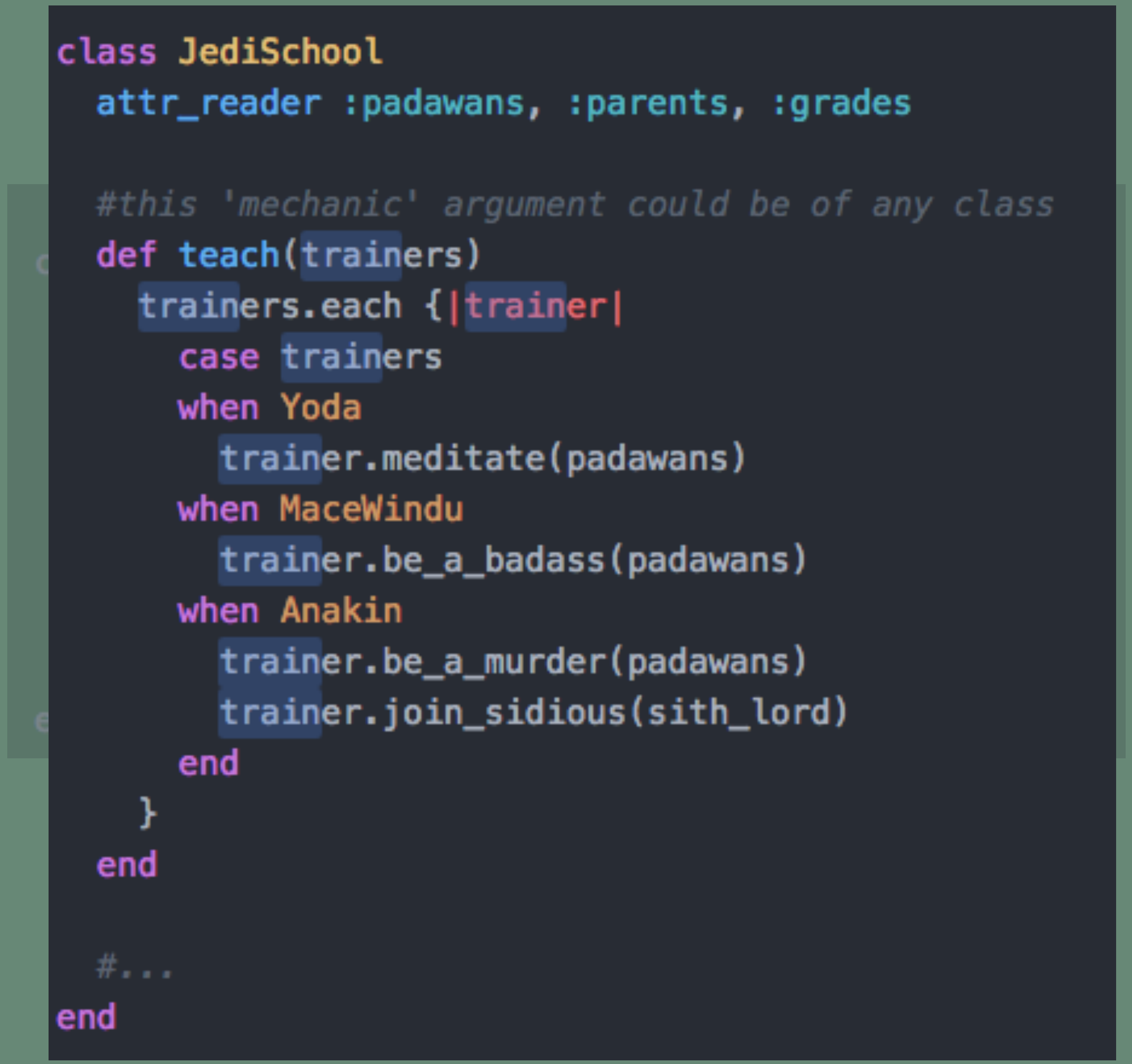
```ruby
class JediSchool
  attr_reader :padawans, :parents, :grades

  #this 'mechanic' argument could be of any class
  def teach(jedi_master)
    jedi_master.teach_padawans(padawan)
  end

  #...
end
```



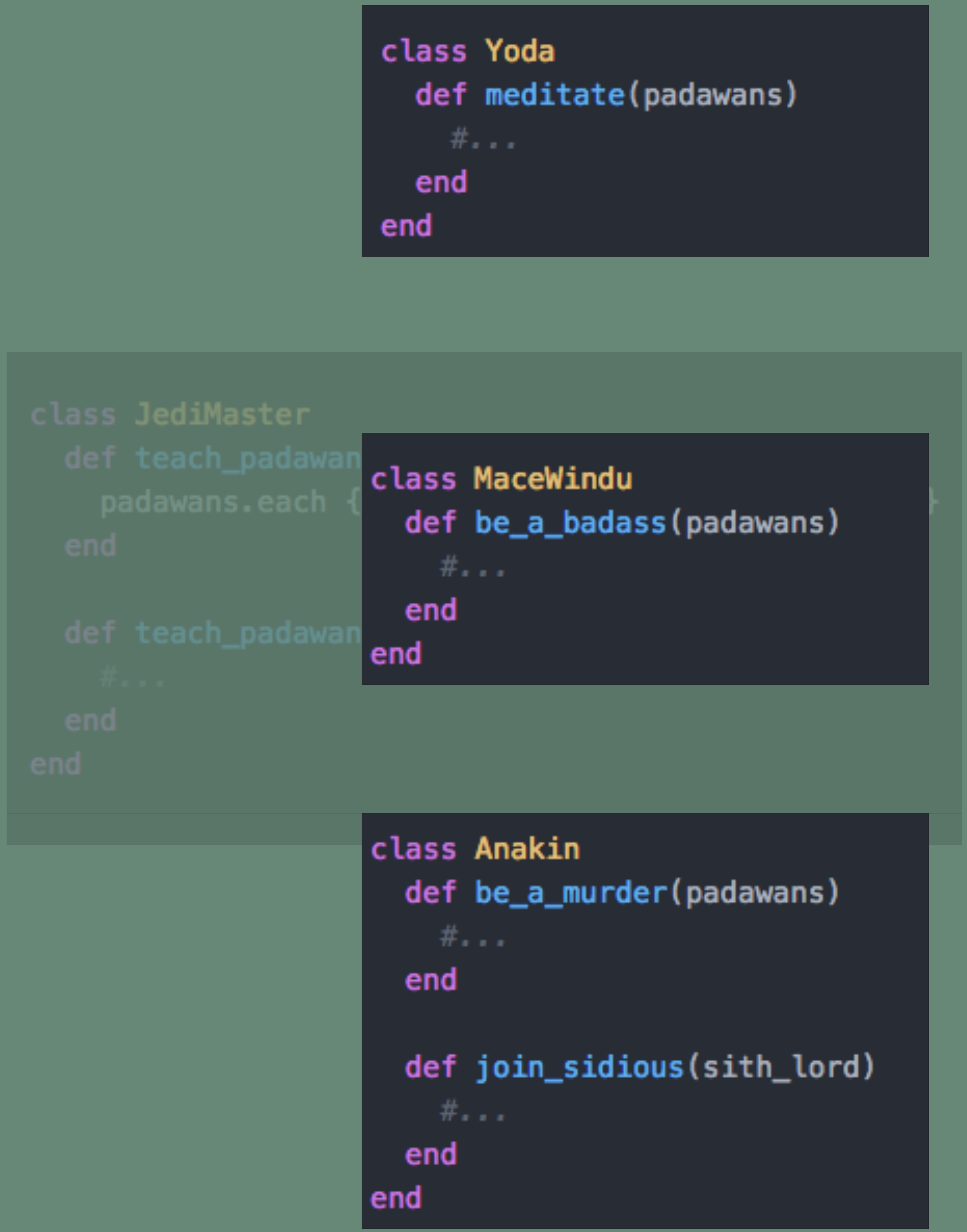
```ruby
class JediMaster
  def teach_padawans(padawans)
    padawans.each {|padawan| teach_padawan(padawan)}
  end

  def teach_padawan(padawan)
    #...
  end
end
```



```ruby
#Here you started with good intentions. Look, you don't
#care at all what kind of thing wants to teach your padawans,
#you just care that it can do teach_padawans
```



```ruby
class JediSchool
  attr_reader :padawans, :parents, :grades

  #this 'mechanic' argument could be of any class
  def teach(trainers)
    trainers.each {|trainer|
      case trainers
      when Yoda
        trainer.meditate(padawans)
      when MaceWindu
        trainer.be_a_badass(padawans)
      when Anakin
        trainer.be_a_murder(padawans)
        trainer.join_sidious(sith_lord)
      end
    }
  end

  #...
end
```

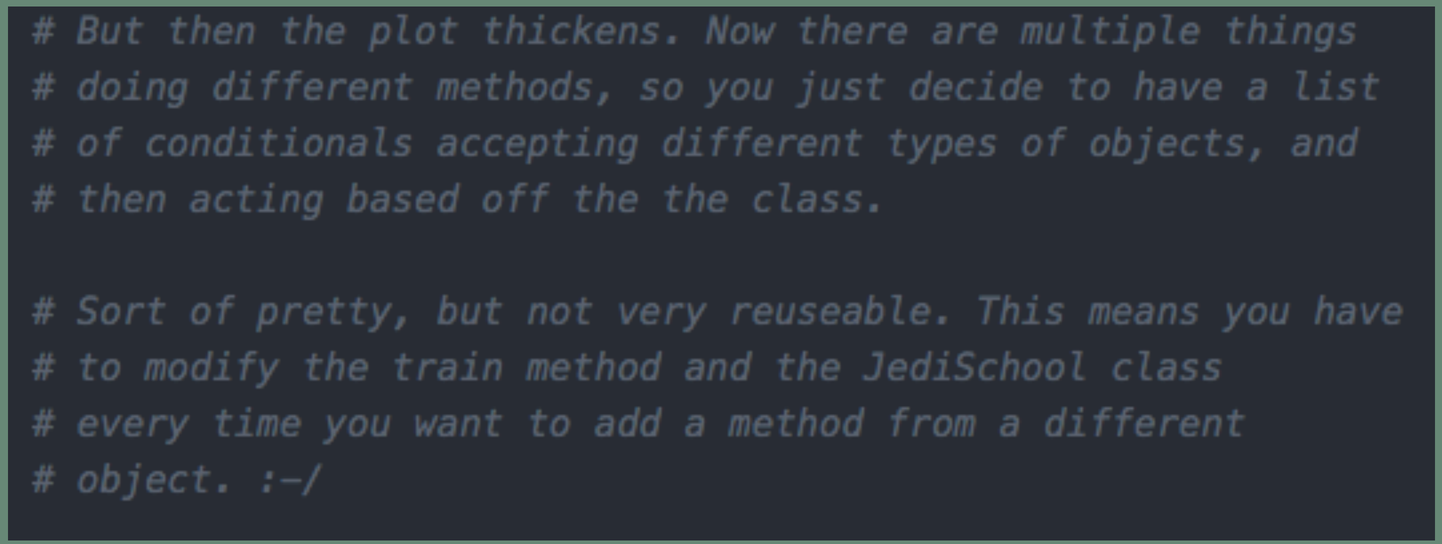


```ruby
class Yoda
  def meditate(padawans)
    #...
  end
end
```

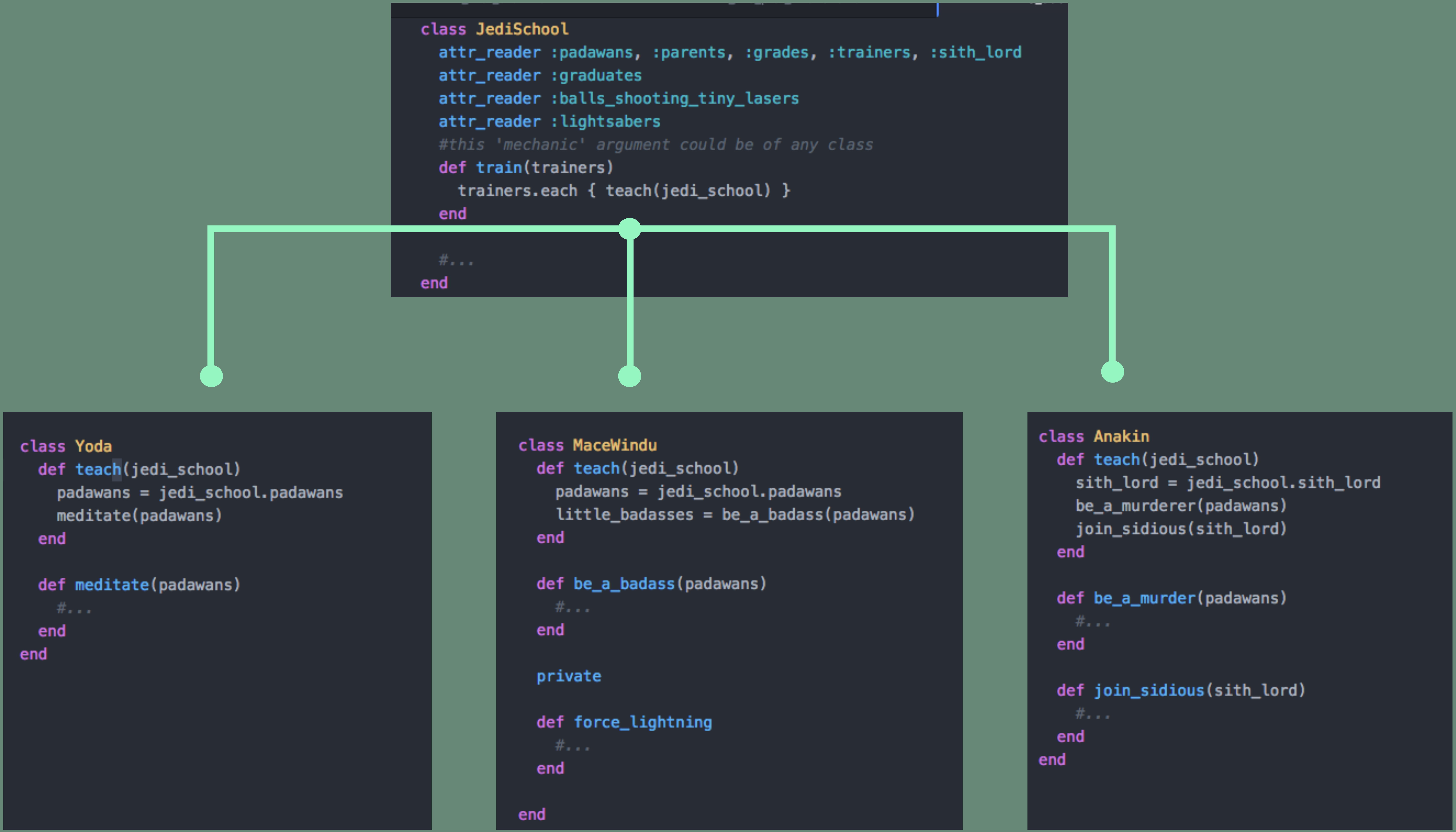
```ruby
class JediMaster
  def teach_padawan
    padawans.each {
  end

  def teach_padawan
    #...
  end
end
```

```ruby
class MaceWindu
  def be_a_badass(padawans)
    #...
  end
end
```

```ruby
class Anakin
  def be_a_murder(padawans)
    #...
  end

  def join_sidious(sith_lord)
    #...
  end
end
```



```ruby
# But then the plot thickens. Now there are multiple things
# doing different methods, so you just decide to have a list
# of conditionals accepting different types of objects, and
# then acting based off the the class.

# Sort of pretty, but not very reuseable. This means you have
# to modify the train method and the JediSchool class
# every time you want to add a method from a different
# object. :-/
```



```ruby
class JediSchool
  attr_reader :padawans, :parents, :grades, :trainers, :sith_lord
  attr_reader :graduates
  attr_reader :balls_shooting_tiny_lasers
  attr_reader :lightsabers
  #this 'mechanic' argument could be of any class
  def train(trainers)
    trainers.each { teach(jedi_school) }
  end

  #...
end
```

```ruby
class Yoda
  def teach(jedi_school)
    padawans = jedi_school.padawans
    meditate(padawans)
  end

  def meditate(padawans)
    #...
  end
end
```

```ruby
class MaceWindu
  def teach(jedi_school)
    padawans = jedi_school.padawans
    little_badasses = be_a_badass(padawans)
  end

  def be_a_badass(padawans)
    #...
  end

  private

  def force_lightning
    #...
  end

end
```

```ruby
class Anakin
  def teach(jedi_school)
    sith_lord = jedi_school.sith_lord
    be_a_murderer(padawans)
    join_sidious(sith_lord)
  end

  def be_a_murder(padawans)
    #...
  end

  def join_sidious(sith_lord)
    #...
  end
end
```