Getting Lean

Designing and Maintaining Software (DAMS)

Louis Rose

Lean software…

Has no extra parts

Solves the problem at hand and no more

Is often easier to change (i.e., is more habitable)

The Advice I Want to Give

~~Your classes should have between no less than X and no more than Y attributes / methods.~~

The Advice I Can Give

Aim for relatively small things. (And then make them a little smaller than that).

Start with a larger thing, if you like or must. Then refactor.

Extract Method

Identify extraction

```
class Menu
  def display
    puts "****************"
    puts "**    Pizzas    **"
    puts "****************"

    pizzas.each do |pizza|
     ...
    end
  end
end
```

Create new method

```
class Menu
  def display
    puts "****************"
    puts "**   Pizzas   **"
    puts "****************"

    pizzas.each do |pizza|
     ...
    end
  end

  def display_banner
  end
end
```

Copy method body

```ruby
class Menu
  def display
    puts "****************"
    puts "**   Pizzas   **"
    puts "****************"

    pizzas.each do |pizza|
     ...
    end
  end

  def display_banner
    puts "****************"
    puts "**   Pizzas   **"
    puts "****************"
  end
end
```

Use the new method

```
class Menu
  def display
    display_banner

    pizzas.each do |pizza|
     ...
    end
  end

  def display_banner
    puts "****************"
    puts "**   Pizzas   **"
    puts "****************"
  end
end
```

Extract Method
(with local variable)

Identify extraction

```
class Menu
  def display
    sections.each do |section|
      puts "****************"
      puts "** #{section} **"
      puts "****************"

      pizzas.each do |pizza|
        ...
      end
    end
  end
end
```

Create new method

```
class Menu
  def display
    sections.each do |section|
      puts "****************"
      puts "**  #{section} **"
      puts "****************"

      pizzas.each do |pizza|
        ...
      end
    end
  end

  def display_banner(section)
  end
end
```

Copy the method body

```ruby
class Menu
  def display
    sections.each do |section|
      puts "****************"
      puts "**  #{section} **"
      puts "****************"

      pizzas.each do |pizza|
        ...
      end
    end
  end

  def display_banner(section)
    puts "****************"
    puts "**  #{section} **"
    puts "****************"
  end
end
```

Use the new method

```
class Menu
  def display
    sections.each do |section|
      display_banner(section)

      pizzas.each do |pizza|
        ...
      end
    end
  end

  def display_banner(section)
    puts "****************"
    puts "** #{section} **"
    puts "****************"
  end
end
```

Extract Method
(with a local assignment)

Identify extraction

```
class Menu
  def display
    most_popular = pizzas.first
    pizzas.each do |pizza|
      most_popular = pizza if pizza.likes > most_popular.likes
    end
    puts "Customer favourite: #{most_popular.name}"
  end
end
```

Create new method

```
class Menu
  def display
    most_popular = pizzas.first
    pizzas.each do |pizza|
      most_popular = pizza if pizza.likes > most_popular.likes
    end
    puts "Customer favourite: #{most_popular.name}"
  end

  def most_popular_from(pizzas)
  end
end
```

Copy method body

```
class Menu
  def display
    most_popular = pizzas.first
    pizzas.each do |pizza|
      most_popular = pizza if pizza.likes > most_popular.likes
    end
    puts "Customer favourite: #{most_popular.name}"
  end

  def most_popular_from(pizzas)
    most_popular = pizzas.first
    pizzas.each do |pizza|
      most_popular = pizza if pizza.likes > most_popular.likes
    end
  end
end
```

Return the assigned value

```
class Menu
  def display
    most_popular = pizzas.first
    pizzas.each do |pizza|
      most_popular = pizza if pizza.likes > most_popular.likes
    end
    puts "Customer favourite: #{most_popular.name}"
  end

  def most_popular_from(pizzas)
    most_popular = pizzas.first
    pizzas.each do |pizza|
      most_popular = pizza if pizza.likes > most_popular.likes
    end
    most_popular
  end
end
```

Use the new method

```ruby
class Menu
  def display
    most_popular = most_popular_from(pizzas)
    puts "Customer favourite: #{most_popular.name}"
  end

  def most_popular_from(pizzas)
    most_popular = pizzas.first
    pizzas.each do |pizza|
      most_popular = pizza if pizza.likes > most_popular.likes
    end
    most_popular
  end
end
```

Extract Class

Identify extraction

```
class Menu
  def display
  end

  def display_banner
  end

  def display_pizza
  end

  def display_pasta
  end

  def display_favourite_dishes
  end

  def load_from_xml(path)
  end

  def add_pizza(pizza)
  end

  def add_pasta(pasta)
  end
end
```

Create new class

```
class Menu
  def display
  end

  def display_banner
  end

  def display_pizza
  end

  def display_pasta
  end

  def display_favourite_dishes
  end

  def load_from_xml(path)
  end

  def add_pizza(pizza)
  end

  def add_pasta(pasta)
  end
end
```

```
class MenuDisplayer
end
```

Create link to new class

```
class Menu
  def display
  end

  def display_banner
  end

  def display_pizza
  end

  def display_pasta
  end

  def display_favourite_dishes
  end

  private

  def displayer
    @d ||= MenuDisplayer.new
  end
  …
end
```

```
class MenuDisplayer
end
```

For each method…

```
class Menu
  def display
  end

  def display_banner
  end

  def display_pizza
  end

  def display_pasta
  end

  def display_favourite_dishes
  end

  private

  def displayer
    @d ||= MenuDisplayer.new
  end
  …
end
```

```
class MenuDisplayer
end
```

Copy method to new class

```
class Menu
  def display
  end

  def display_banner
  end

  def display_pizza
  end

  def display_pasta
  end

  def display_favourite_dishes
  end

  private

  def displayer
    @d ||= MenuDisplayer.new
  end
  …
end
```

```
class MenuDisplayer
  def display
  end
end
```

Forward original to new

```ruby
class Menu
  def display
    displayer.display
  end

  def display_banner
  end

  def display_pizza
  end

  def display_pasta
  end

  def display_favourite_dishes
  end

  private

  def displayer
    @d ||= MenuDisplayer.new
  end
  …
end
```

```ruby
class MenuDisplayer
  def display
  end
end
```

Repeat for all methods

```
class Menu
  def display
    displayer.display
  end

  def display_banner
    displayer.display_banner
  end

  def display_pizza
    displayer.display_pizza
  end

  def display_pasta
    displayer.pasta
  end

  def display_favourite_dishes
    displayer.favourite_dishes
  end

  private

  def displayer
    @d ||= MenuDisplayer.new
  end
  …
end
```

```
class MenuDisplayer
  def display
  end

  def display_banner
  end

  def display_pizza
  end

  def display_pasta
  end

  def display_favourite_dishes
  end
end
```

Refactor original class

```
class Menu
  def display
    displayer.display
  end

  def display_banner
    displayer.display_banner
  end

  def display_pizza
    displayer.display_pizza
  end

  def display_pasta
    displayer.pasta
  end

  def display_favourite_dishes
    displayer.favourite_dishes
  end

  private

  def displayer
    @d ||= MenuDisplayer.new
  end
  …
end
```

```
class MenuDisplayer
  def display
  end

  def display_banner
  end

  def display_pizza
  end

  def display_pasta
  end

  def display_favourite_dishes
  end
end
```

Private methods?

```
class Menu
  def display
    displayer.display
  end

  private

  def display_banner
    displayer.display_banner
  end

  def display_pizza
    displayer.display_pizza
  end

  def display_pasta
    displayer.pasta
  end

  def display_favourite_dishes
    displayer.favourite_dishes
  end

  def displayer
    @d ||= MenuDisplayer.new
  end
  …
end
```

```
class MenuDisplayer
  def display
  end

  private

  def display_banner
  end

  def display_pizza
  end

  def display_pasta
  end

  def display_favourite_dishes
  end
end
```

Private methods?

```
class Menu
  def display
    displayer.display
  end

  private

  def displayer
    @d ||= MenuDisplayer.new
  end
  …
end
```

```
class MenuDisplayer
  def display
  end

  private

  def display_banner
  end

  def display_pizza
  end

  def display_pasta
  end

  def display_favourite_dishes
  end
end
```

Must be public?

```ruby
class Menu
  def display
    displayer.display
  end

  def display_banner
    displayer.display_banner
  end

  def display_pizza
    displayer.display_pizza
  end

  def display_pasta
    displayer.pasta
  end

  def display_favourite_dishes
    displayer.favourite_dishes
  end

  private

  def displayer
    @d ||= MenuDisplayer.new
  end
  …
end
```

```ruby
class MenuDisplayer
  def display
  end

  private

  def display_banner
  end

  def display_pizza
  end

  def display_pasta
  end

  def display_favourite_dishes
  end
end
```

Use Forwardable

```ruby
class Menu
  extend Forwardable

  def_delegators :displayer,
   :display, :display_banner,
   :display_pizza, :display_pasta
   :display_favourite_dishes

  private

  def displayer
    @d ||= MenuDisplayer.new
  end
  …
end
```

```ruby
class MenuDisplayer
  def display
  end

  private

  def display_banner
  end

  def display_pizza
  end

  def display_pasta
  end

  def display_favourite_dishes
  end
end
```

(Almost) never do this

```
class Menu
  # Now part of public API
  def displayer
    @d ||= MenuDisplayer.new
  end
  …
end
```

```
class MenuDisplayer
  def display
  end

  private

  def display_banner
  end

  def display_pizza
  end

  def display_pasta
  end

  def display_favourite_dishes
  end
end
```

Summary

There’s no “correct” size for
a class (method / project).

Aim for small things. Experiment
with REALLY small things to see
where the boundaries lie.

Design or refactor towards smaller things
by extracting methods & classes.

Learn Enumerable!

```ruby
class Menu
  def display
    most_popular = most_popular_from(pizzas)
    puts "Customer favourite: #{most_popular.name}"
  end

  def most_popular_from(pizzas)
    most_popular = pizzas.first
    pizzas.each do |pizza|
      most_popular = pizza if pizza.likes > most_popular.likes
    end
    most_popular
  end
end
```

Learn Enumerable!

```
class Menu
  def display
    most_popular = most_popular_from(pizzas)
    puts "Customer favourite: #{most_popular.name}"
  end

  def most_popular_from(pizzas)
    pizzas.max_by(&:likes)
  end
end
```