Getting Cohesion

Designing and Maintaining Software (DAMS)

Louis Rose

Single Responsibility Principle

A class should have only one reason to change.

- Martin and Martin

Chapter 8, Agile Principles, Patterns and Practices in C#, Prentice Hall, 2009

Finding responsibilities

```ruby
class Pizza
  attr_reader :toppings
  attr_reader :likes

  def initialize(toppings)
    @toppings = toppings
    @likes = 0
  end

  def title
    toppings.title
  end

  def cost
    toppings.cost + 4
  end

  def like!
    @likes += 1
  end

  def rating
    if likes > 1000 then "A"
    elsif likes > 500 then "B"
    elsif likes > 250 then "C"
    elsif likes > 100 then "D"
    else "E" end
  end

  def worse_rating
    if rating == "E"
      nil
    else
      rating.succ
    end
  end
end
```

Based on: https://www.youtube.com/watch?v=5yX6ADjyqyE

Finding responsibilities

```
class Pizza
  attr_reader :toppings
  attr_reader :likes

  def initialize(toppings)
    @toppings = toppings
    @likes = 0
  end

  def title
    toppings.title
  end

  def cost
    toppings.cost + 4
  end

  def like!
    @likes += 1
  end

  def rating
    if likes > 1000 then "A"
    elsif likes > 500 then "B"
    elsif likes > 250 then "C"
    elsif likes > 100 then "D"
    else "E" end
  end

  def worse_rating
    if rating == "E"
      nil
    else
      rating.succ
    end
  end
end
```

Finding responsibilities

```ruby
class Pizza
  attr_reader :toppings
  attr_reader :likes

  def initialize(toppings)
    @toppings = toppings
    @likes = 0
  end

  def title
    toppings.title
  end

  def cost
    toppings.cost + 4
  end

  def like!
    @likes += 1
  end

  def rating
    if likes > 1000 then "A"
    elsif likes > 500 then "B"
    elsif likes > 250 then "C"
    elsif likes > 100 then "D"
    else "E" end
  end

  def worse_rating
    if rating == "E"
      nil
    else
      rating.succ
    end
  end
end
```

Finding responsibilities

```ruby
class Pizza
  attr_reader :toppings
  attr_reader :likes

  def initialize(toppings)
    @toppings = toppings
    @likes = 0
  end

  def title
    toppings.title
  end

  def cost
    toppings.cost + 4
  end

  def like!
    @likes += 1
  end

  def rating
    if likes > 1000 then "A"
    elsif likes > 500 then "B"
    elsif likes > 250 then "C"
    elsif likes > 100 then "D"
    else "E" end
  end

  def worse_rating
    if rating == "E"
      nil
    else
      rating.succ
    end
  end
end
```

Finding responsibilities

```
class Pizza
  attr_reader :toppings
  attr_reader :likes

  def initialize(toppin
    @toppings = toppi
    @likes = 0
  end

  def title
    toppings.title
  end

  def cost
    toppings.cost + 4
  end

  def like!
    @likes += 1
  end
```

```
def rating
                en "A"
                hen "B"
                hen "C"
                hen "D"
```

Finding responsibilities

```
class Pizza
  attr_reader :toppings
  attr_reader :likes

  def initialize(toppin
    @toppings = toppi
    @likes = 0
  end

  def title
    toppings.title
  end

  def cost
    toppings.cost + 4
  end

  def like!
    @likes += 1
  end
```

```
def rating
    en "A"
    hen "B"
    hen "C"
    hen "D"
```

Finding responsibilities

```ruby
class Pizza
  attr_reader :toppings
  attr_reader :likes

  def initialize(toppings)
    @toppings = toppings
    @likes = 0
  end

  def title
    toppings.title
  end

  def cost
    toppings.cost + 4
  end

  def like!
    @likes += 1
  end

  def rating
    if likes > 1000 then "A"
    elsif likes > 500 then "B"
    elsif likes > 250 then "C"
    elsif likes > 100 then "D"
    else "E" end
  end

  def worse_rating
    if rating == "E"
      nil
    else
      rating.succ
    end
  end
end
```

Tactics

Extract objects from primitives and data clumps

Move methods to avoid feature envy

Extract wrappers for ancillary responsibilities

Primitive Obsession

```
class Pizza
  attr_reader :toppings
  attr_reader :likes

  def initialize(toppings)
    @toppings = toppings
    @likes = 0
  end

  def title
    toppings.title
  end

  def cost
    toppings.cost + 4
  end

  def like!
    @likes += 1
  end

  def rating
    if likes > 1000 then "A"
    elsif likes > 500 then "B"
    elsif likes > 250 then "C"
    elsif likes > 100 then "D"
    else "E" end
  end

  def worse_rating
    if rating == "E"
      nil
    else
      rating.succ
    end
  end
end
```

Primitive Obsession

```
class Pizza
  def like!
    @likes += 1
  end

  def rating
    if likes > 1000 then "A"
    elsif likes > 500 then "B"
    elsif likes > 250 then "C"
    elsif likes > 100 then "D"
    else "E" end
  end

  def worse_rating
    if rating == "E"
      nil
    else
      rating.succ
    end
  end
```

```
end
```

Extract Primitive

```ruby
class Pizza
  def like!
    @likes += 1
  end

  def rating
   Rating.from_likes(likes)
  end

  def worse_rating
    if rating.letter == "E"
      nil
    else
      rating.letter.succ
    end
  end
end
```

```ruby
class Rating
  def self.from_likes(likes)
    if likes > 1000 then new("A")
    elsif likes > 500 then new("B")
    elsif likes > 250 then new("C")
    elsif likes > 100 then new("D")
    else new("E") end
  end

  attr_reader :letter

  def initialize(letter)
    @letter = letter
  end
end
```

Extract Primitive

```ruby
class Pizza
  def like!
    @likes += 1
  end

  def rating
    Rating.from_likes(likes)
  end

  def worse_rating
    if rating.letter == "E"
      nil
    else
      rating.letter.succ
    end
  end
end
```

```ruby
class Rating
  def self.from_likes(likes)
    if likes > 1000 then new("A")
    elsif likes > 500 then new("B")
    elsif likes > 250 then new("C")
    elsif likes > 100 then new("D")
    else new("E") end
  end

  attr_reader :letter

  def initialize(letter)
    @letter = letter
  end
end
```

Feature Envy

```ruby
class Pizza
  def like!
    @likes += 1
  end

  def rating
    Rating.from_likes(likes)
  end

  def worse_rating
    if rating.letter == "E"
      nil
    else
      rating.letter.succ
    end
  end
end
```

```ruby
class Rating
  def self.from_likes(likes)
    if likes > 1000 then new("A")
    elsif likes > 500 then new("B")
    elsif likes > 250 then new("C")
    elsif likes > 100 then new("D")
    else new("E") end
  end

  attr_reader :letter

  def initialize(letter)
    @letter = letter
  end
end
```

Move Method

```ruby
class Pizza
  def like!
    @likes += 1
  end

  def rating
    Rating.from_likes(likes)
  end

  def worse_rating
    if rating.letter == "E"
      nil
    else
      rating.letter.succ
    end
  end
end
```

```ruby
class Rating
  def self.from_likes(likes)
    if likes > 1000 then new("A")
    elsif likes > 500 then new("B")
    elsif likes > 250 then new("C")
    elsif likes > 100 then new("D")
    else new("E") end
  end

  attr_reader :letter

  def initialize(letter)
    @letter = letter
  end

  def worse_rating
    if letter == "E"
      nil
    else
```

Move Method

```ruby
class Pizza
  def like!
    @likes += 1
  end

  def rating
    Rating.from_likes(likes)
  end

  def worse_rating
    rating.worse_rating
  end
end
```

```ruby
class Rating
  def self.from_likes(likes)
    if likes > 1000 then new("A")
    elsif likes > 500 then new("B")
    elsif likes > 250 then new("C")
    elsif likes > 100 then new("D")
    else new("E") end
  end

  attr_reader :letter

  def initialize(letter)
    @letter = letter
  end

  def worse_rating
    if letter == "E"
      nil
    else
```

Move Method

```
class Pizza
  def like!
    @likes += 1
  end

  def rating
    Rating.from_likes(likes)
  end

  def worse_rating
    rating.worse_rating
  end
end
```

```
class Rating
  def self.from_likes(likes)
    if likes > 1000 then new("A")
    elsif likes > 500 then new("B")
    elsif likes > 250 then new("C")
    elsif likes > 100 then new("D")
    else new("E") end
  end

  def initialize(letter)
    @letter = letter
  end

  def worse_rating
    if @letter == "E"
      nil
    else
      @letter.succ
    end
```

Move Method

```ruby
class Pizza
  def like!
    @likes += 1
  end

  def rating
    Rating.from_likes(likes)
  end

  def worse_rating
    rating.worse
  end
end
```

```ruby
class Rating
  def self.from_likes(likes)
    if likes > 1000 then new("A")
    elsif likes > 500 then new("B")
    elsif likes > 250 then new("C")
    elsif likes > 100 then new("D")
    else new("E") end
  end

  def initialize(letter)
    @letter = letter
  end

  def worse
    if @letter == "E"
      nil
    else
      @letter.succ
    end
```

Move Method

```
class Pizza
  def like!
    @likes += 1
  end

  def rating
    Rating.from_likes(likes)
  end
end
```

```
class Rating
  def self.from_likes(likes)
    if likes > 1000 then new("A")
    elsif likes > 500 then new("B")
    elsif likes > 250 then new("C")
    elsif likes > 100 then new("D")
    else new("E") end
  end

  def initialize(letter)
    @letter = letter
  end

  def worse
    if @letter == "E"
      nil
    else
      @letter.succ
    end
```

Back to responsibilities

```ruby
class Pizza
  attr_reader :toppings
  attr_reader :likes

  def initialize(toppings)
    @toppings = toppings
    @likes = 0
  end

  def title
    toppings.title
  end

  def cost
    toppings.cost + 4
  end

  def like!
    @likes += 1
  end

  def rating
    if likes > 1000 then "A"
    elsif likes > 500 then "B"
    elsif likes > 250 then "C"
    elsif likes > 100 then "D"
    else "E" end
  end

  def worse_rating
    if rating == "E"
      nil
    else
      rating.succ
    end
  end
end
```

Back to responsibilities

```ruby
class Pizza
  attr_reader :toppings
  attr_reader :likes

  def initialize(toppings)
    @toppings = toppings
    @likes = 0
  end

  def title
    toppings.title
  end

  def cost
    toppings.cost + 4
  end

  def like!
    @likes += 1
  end

  def rating
    Rating.from_likes(likes)
  end
end
```

Extract ancillaries…

```ruby
class Pizza
  attr_reader :toppings
  attr_reader :likes

  def initialize(toppings)
    @toppings = toppings
    @likes = 0
  end

  def title
    toppings.title
  end

  def cost
    toppings.cost + 4
  end

  def like!
    @likes += 1
  end

  def rating
    Rating.from_likes(likes)
  end
end
```

… using delegates

```ruby
class Pizza
  attr_reader :toppings

  def initialize(toppings)
    @toppings = toppings
  end

  def title
    toppings.title
  end

  def cost
    toppings.cost + 4
  end
end
```

```ruby
class Likeable < SimpleDelegator
  attr_reader :likes

  def initialize(subject)
    @likes = 0
    super
  end

  def like!
    @likes += 1
  end

  def rating
    Rating.from_likes(likes)
  end
end
```

… using delegates

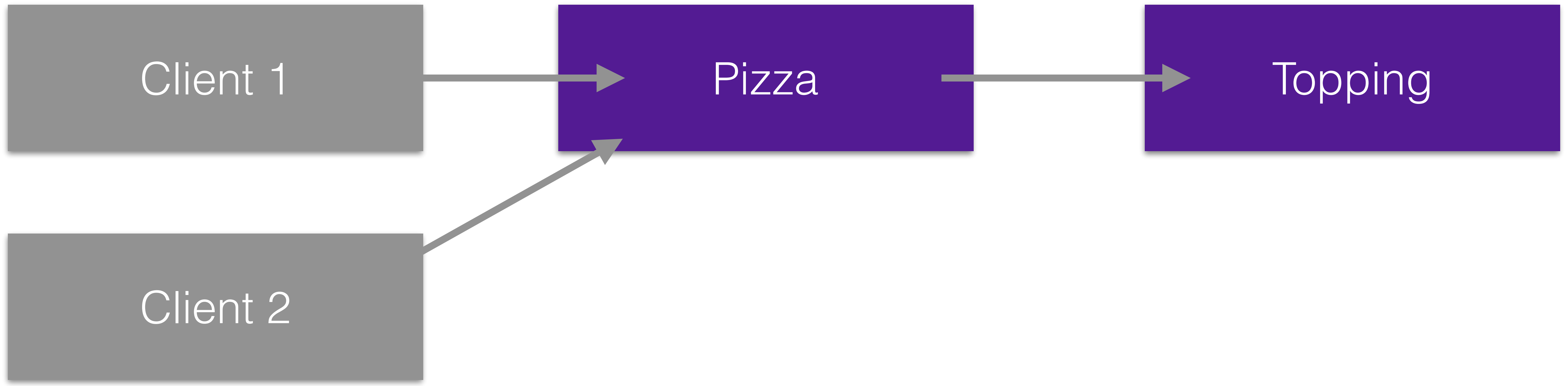
```ruby
class Pizza
  attr_reader :toppings

  def initialize(toppings)
    @toppings = toppings
  end

  def title
    toppings.title
  end

  def cost
    toppings.cost + 4
  end
end
```

```ruby
class Likeable < SimpleDelegator
  attr_reader :likes

  def initialize(subject)
    @likes = 0
    super
  end

  def like!
    @likes += 1
  end

  def rating
    Rating.from_likes(likes)
  end
end
```

```ruby
ham_and_pineapple = Pizza.new(…)
ham_and_pineapple = Likeable.new(ham_and_pineapple)
ham_and_pineapple.like!
ham_and_pineapple.title
```

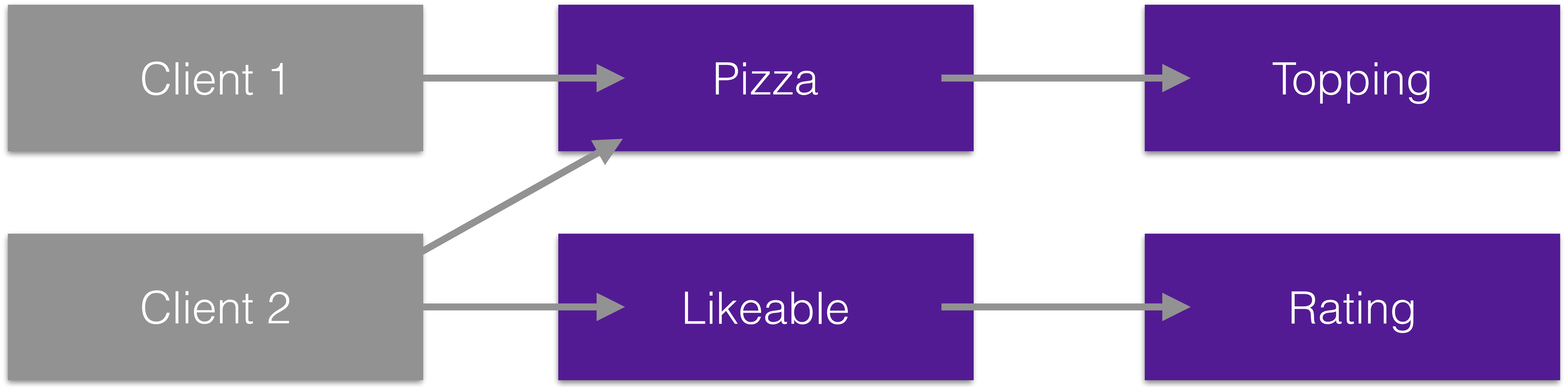
Cohesion & Coupling

Before the refactoring:



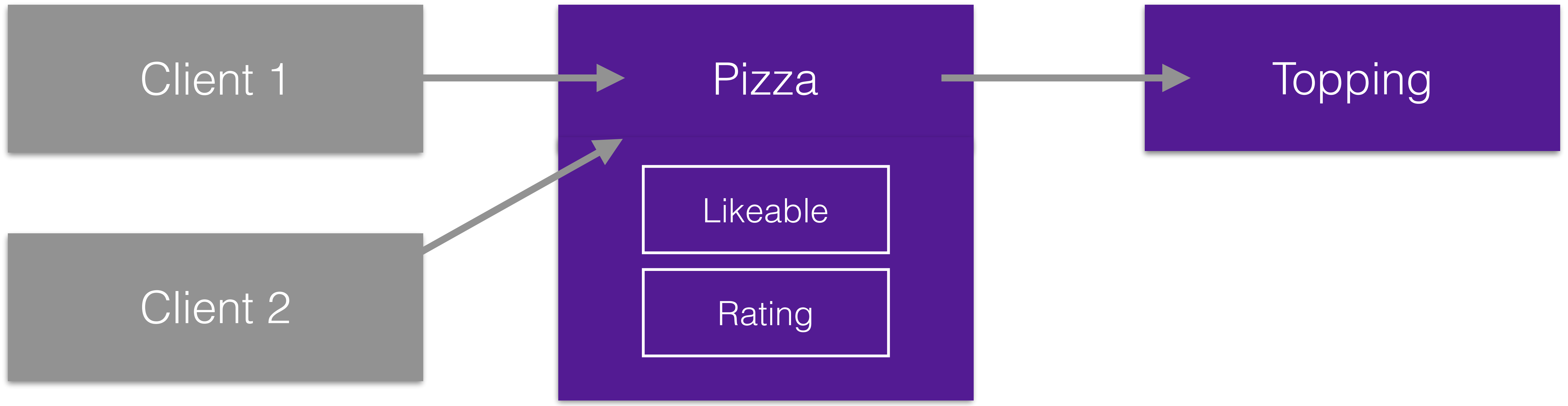
Cohesion & Coupling

After the refactoring:



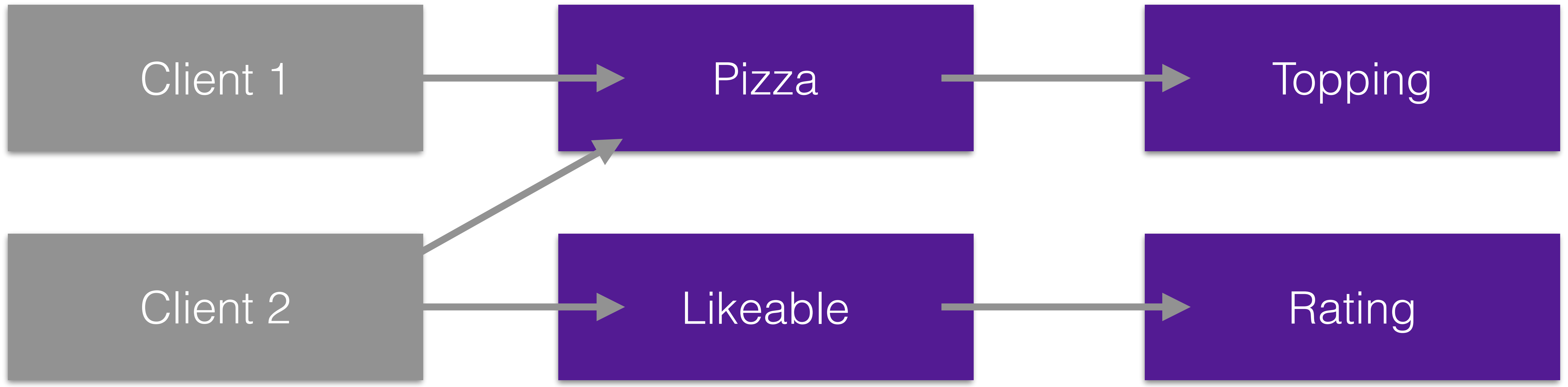
Cohesion & Coupling

All clients were coupled to all of Pizza



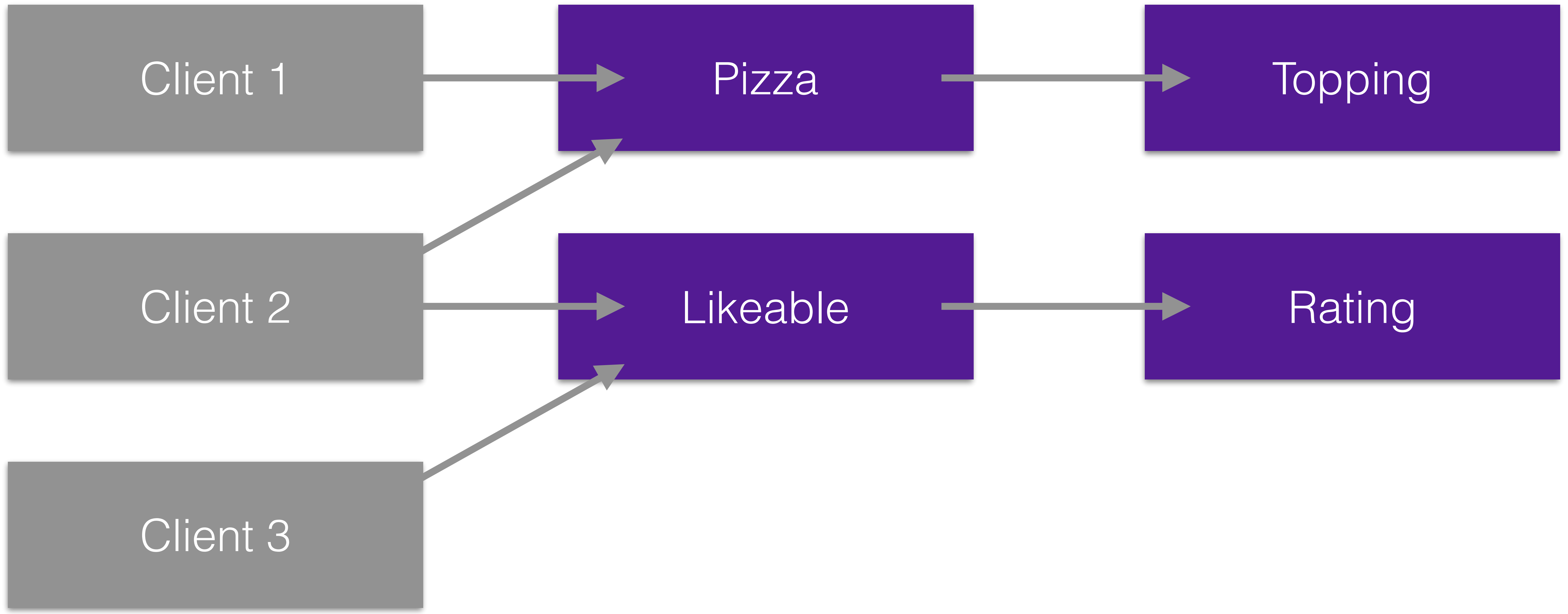
Cohesion & Coupling

Clients that don't need Likeable aren't affected by it



Cohesion & Coupling

Likeable can be re-used in new contexts



Summary

Classes should have a single responsibility:
a single reason to change

Classes with low cohesion normally have
more than one responsibility

Extract classes to redistribute responsibilities