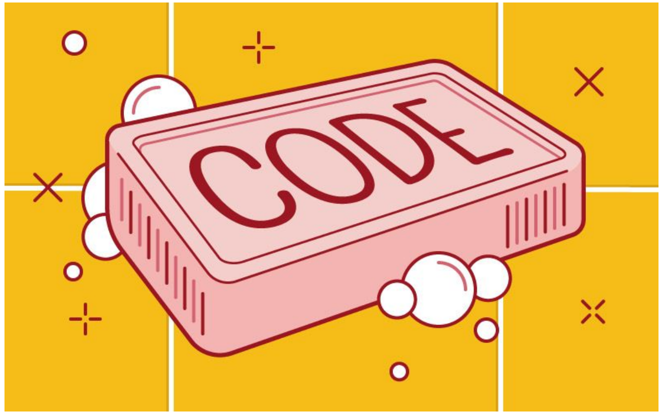
nimble

Clean Code

Terone & Tobias

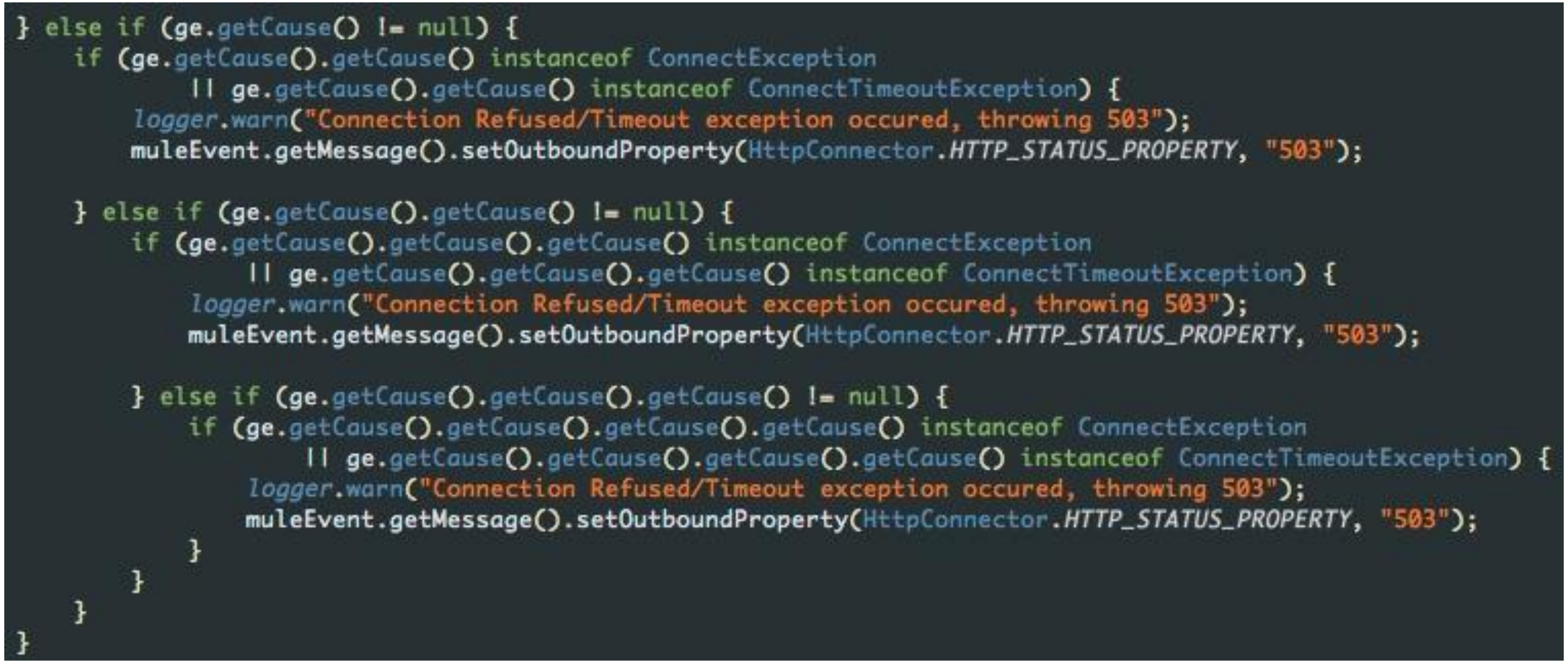
Growth Session #26 - July 26 2019

Introduction - What?



- **Clean code is simple & readable**
  → Keep your code simple in implementation, while taking care of naming conventions, spacing & structure
- **Clean code is considerate & expressive**
  → The code should speak for itself, so future readers will understand
- **Clean code is tested**
  → When changes are made, we can have more confidence our code still works
- **Clean code is SOLID & DRY**
  → Commonly used design principles, such as SRP & OCP

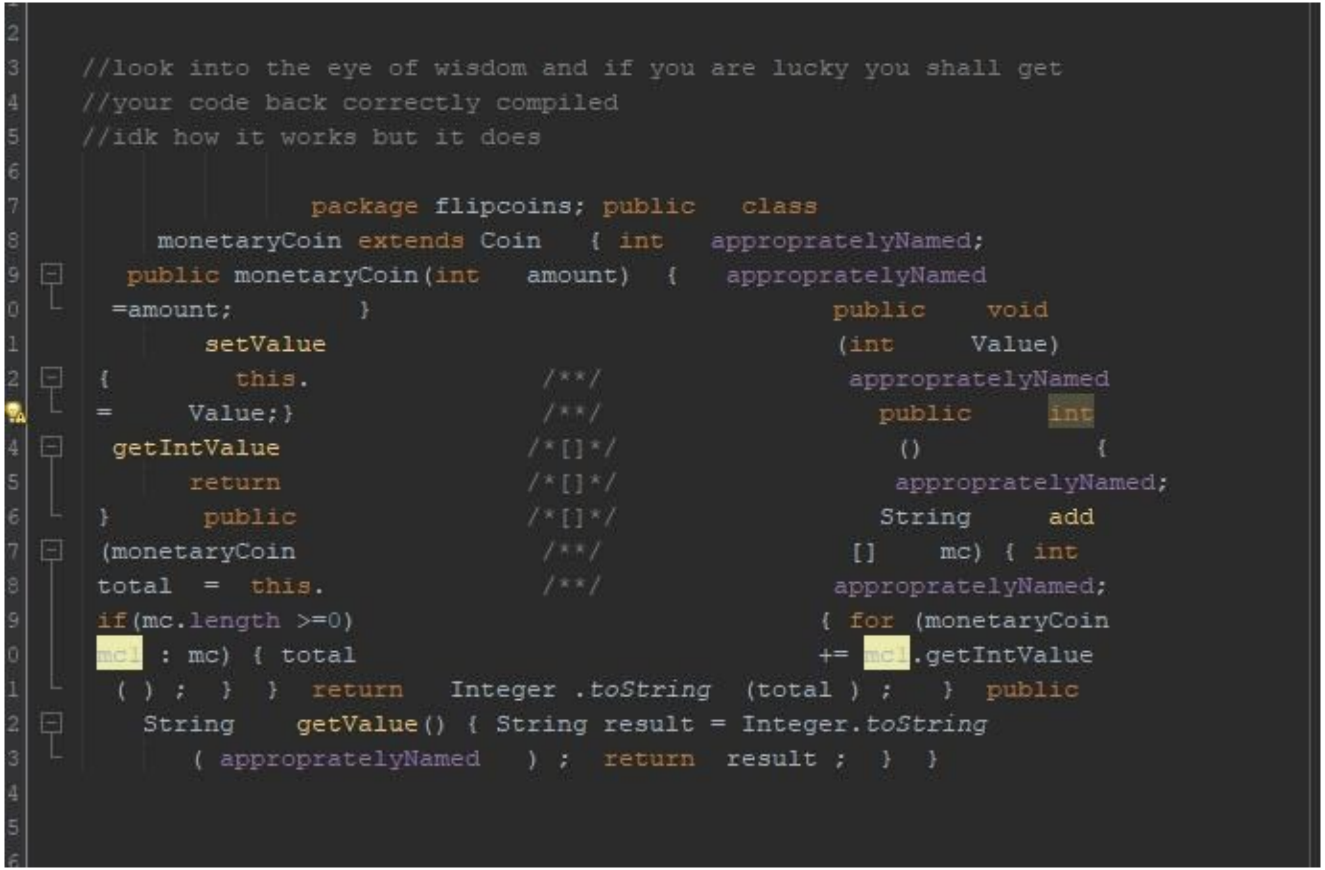
Bad Code



```
} else if (ge.getCause() != null) {
    if (ge.getCause().getCause() instanceof ConnectException
            || ge.getCause().getCause() instanceof ConnectTimeoutException) {
        logger.warn("Connection Refused/Timeout exception occured, throwing 503");
        muleEvent.getMessage().setOutboundProperty(HttpConnector.HTTP_STATUS_PROPERTY, "503");

    } else if (ge.getCause().getCause() != null) {
        if (ge.getCause().getCause().getCause() instanceof ConnectException
                || ge.getCause().getCause().getCause() instanceof ConnectTimeoutException) {
            logger.warn("Connection Refused/Timeout exception occured, throwing 503");
            muleEvent.getMessage().setOutboundProperty(HttpConnector.HTTP_STATUS_PROPERTY, "503");

        } else if (ge.getCause().getCause().getCause() != null) {
            if (ge.getCause().getCause().getCause().getCause() instanceof ConnectException
                    || ge.getCause().getCause().getCause().getCause() instanceof ConnectTimeoutException) {
                logger.warn("Connection Refused/Timeout exception occured, throwing 503");
                muleEvent.getMessage().setOutboundProperty(HttpConnector.HTTP_STATUS_PROPERTY, "503");
            }
        }
    }
}
```

*** Actual production code of a banking software application**

More Bad Code



```
//look into the eye of wisdom and if you are lucky you shall get
//your code back correctly compiled
//idk how it works but it does

               package flipcoins; public   class
     monetaryCoin extends Coin   { int   appropratelyNamed;
   public monetaryCoin(int   amount)  {   appropratelyNamed
  =amount;        }                              public    void
        setValue                                 (int     Value)
 {        this.               /**/                appropratelyNamed
 =     Value;}                /**/                  public     int
  getIntValue                /*[]*/                  ()           {
       return                /*[]*/                  appropratelyNamed;
 }      public               /*[]*/                 String     add
 (monetaryCoin                /**/                [[]    mc) { int
 total  =  this.              /**/               appropratelyNamed;
 if(mc.length >=0)                              { for (monetaryCoin
 mc1 : mc) { total                              += mc1.getIntValue
  ( ) ;  }  }  return   Integer .toString  (total ) ;   }  public
    String    getValue() { String result = Integer.toString
       ( appropratelyNamed   ) ;  return  result ;  }  }
```

Meaningful Names

➔ **Use intention revealing & pronounceable names**

```
int d; // elapsed time in days
```

```
int elapsedTimeInDays;
```

```
String genymdhms
```

```
String generationTimestamp
```

➔ **Beware of using names which are similar**

```
class XYZControllerForEfficientStorageOfStrings
```

```
class XYZControllerForEfficientHandlingOfStrings
```

➔ **Use searchable names**

```
for (int j = 0; j < 34; j++) {
    s += (t[j] * 4) / 5;
}
```

```
int realDaysPerIdealDay = 4;
const int WORK_DAYS_PER_WEEK = 5;
int sum = 0;
for (int = 0; j < NUMBER_OF_TASKS; j++) {
    int realTaskDays = taskEstimate[j] * realDaysPerIdealDay;
    int realTaskWeeks = (realdays / WORK_DAYS_PER_WEEK);
    sum += realTaskWeeks;
}
```

Meaningful Names

➔ **Classes & objects should have noun names**

```
class Customer
```

```
class DateFormatter
```

➔ **Method names should have verb names**

```
fun createPresentation()
```

```
fun explain()
```

➔ **Pick one word per concept, be consistent & stick with it**

```
fun fetch()
```

```
fun retrieve()
```

Functions

- Function should be small!

- Functions should do only one thing, without side effects (SRP)

  TO `functionName()`, it does…

  TO `renderPageWithSetupsAndTeardowns()`, we include the setups and teardowns. Eventually we render the page in HTML.

- Functions should not have more than three arguments

  → **0** : Ideal
  → **1-2** : Good
  → **3** : Should be avoided where possible
  → **4** : Requires very special justification

"The first rule of functions is that they should be small. The second rule of functions is that they should be smaller than that" - Uncle Bob

Comments

Good comments ✓

- Informative comments
- Explanation of intent
- Clarification
- Warning of consequences

Bad comments ✗

- Redundant comments

```
// The booking of a hotel
Booking: booking
```

```
// Render page with setups & teardowns
fun render()

fun renderPageWithSetupsAndTeardowns()
```

- Misleading comments

→ Try to avoid comments, as they're often not updated, when code changes. Which can be misleading!

Classes

- **A class should read like a newspaper**
  → Read it from top to bottom down
  → Starts with a title, but gets more detailed the further you read

- **A class is measured by its responsibilities**
  → A function is measured by its physical lines
  → Class description uses words like “or”, “and”, …? Probably too many responsibilities

- **Cohesion: The indication of the relationship within the class**
  → High cohesion: Its methods & variables are co-dependent
  → Low cohesion: Its methods & variables are not co-dependent

- **Coupling: The indication of the relationships between classes**
  → Tight coupling: It is highly dependent of other classes
  → Loose coupling: It is not highly dependent of other classes

Practical Examples

99 Bottles of Beer

- **99 bottles** of beer on the wall, **99 bottles** of beer.
  Take one down and pass it around, **98 bottles** of beer on the wall.

- **98 bottles** of beer on the wall, **98 bottles** of beer.
  Take one down and pass it around, **97 bottles** of beer on the wall.

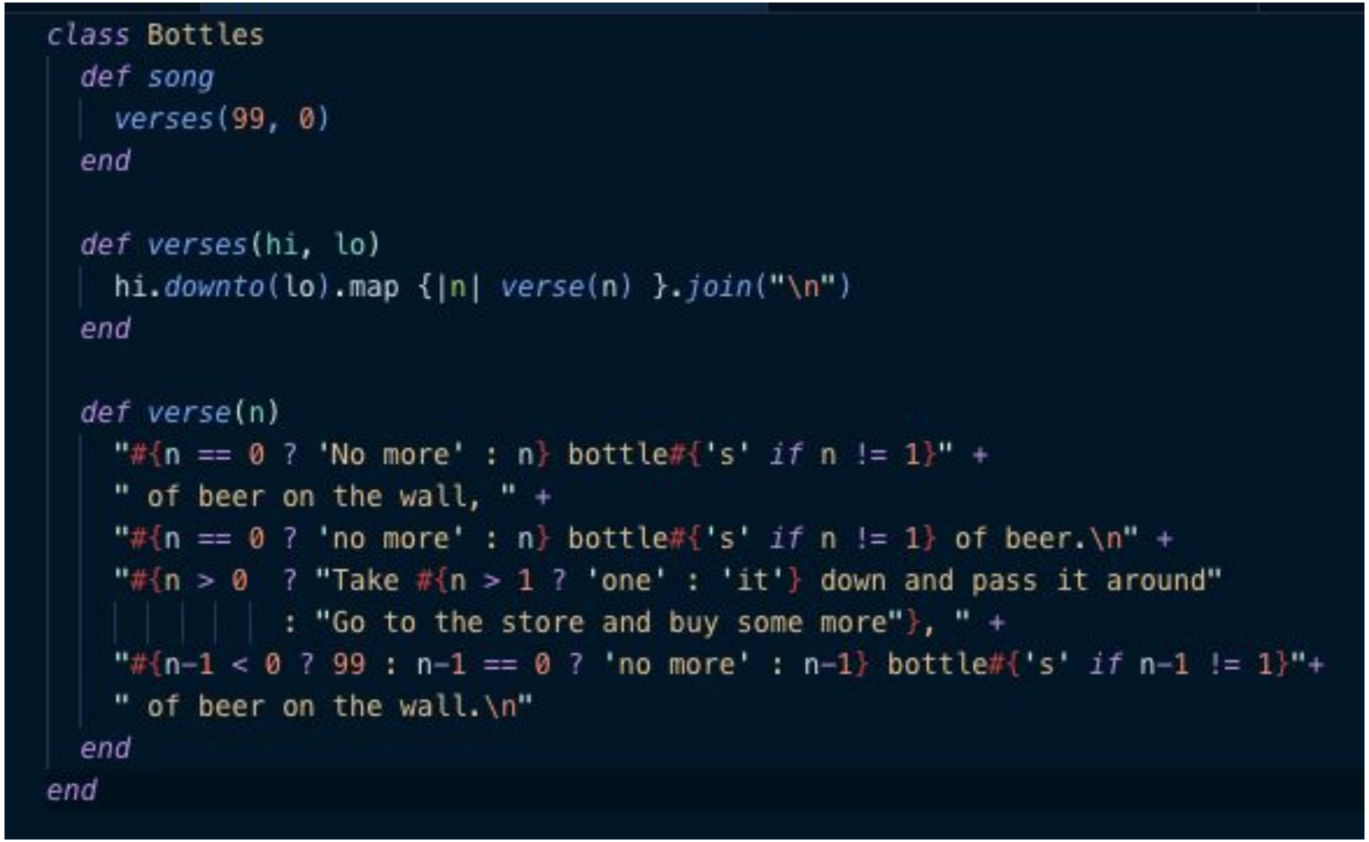
- …

- **2 bottles** of beer on the wall, **2 bottles** of beer.
  Take one down and pass it around, **1 bottle** of beer on the wall.

- **1 bottle** of beer on the wall, **1 bottle** of beer.
  Take one down and pass it around, **no more bottles** of beer on the wall.

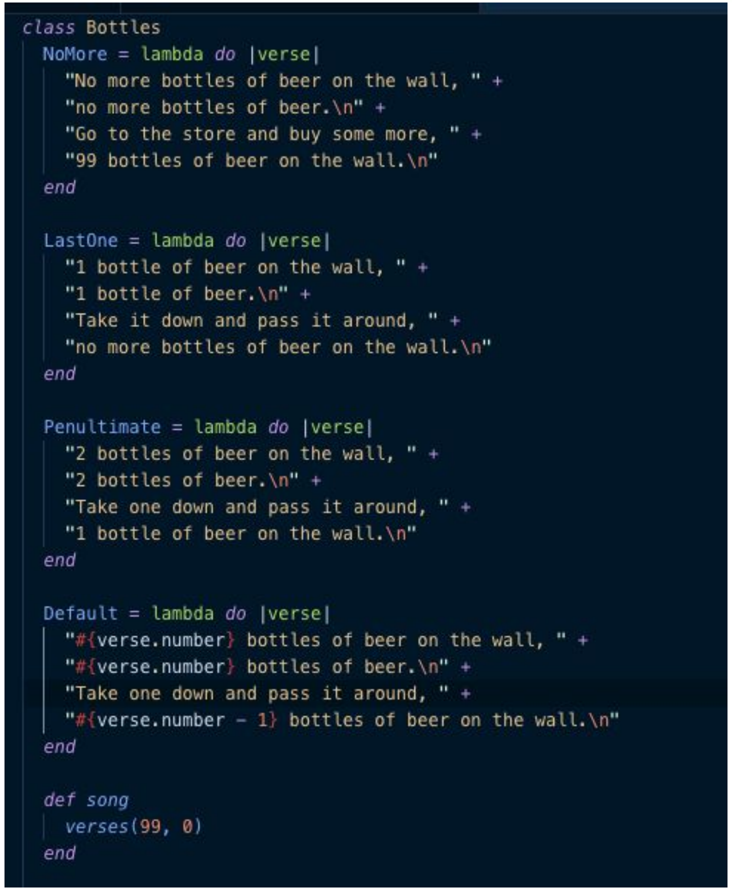
- **No more bottles** of beer on the wall, **no more bottles** of beer.
  Go to the store and buy some more, **99 bottles** of beer on the wall.

99 Bottles of beer - Concise

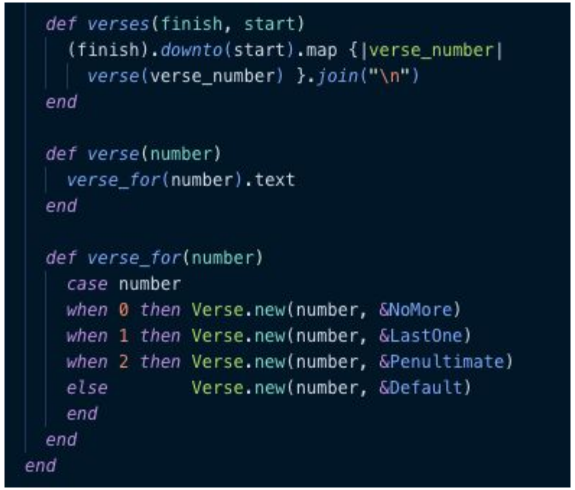


```ruby
class Bottles
  def song
    verses(99, 0)
  end

  def verses(hi, lo)
    hi.downto(lo).map {|n| verse(n) }.join("\n")
  end

  def verse(n)
    "#{n == 0 ? 'No more' : n} bottle#{'s' if n != 1}" +
    " of beer on the wall, " +
    "#{n == 0 ? 'no more' : n} bottle#{'s' if n != 1} of beer.\n" +
    "#{n > 0  ? "Take #{n > 1 ? 'one' : 'it'} down and pass it around"
              : "Go to the store and buy some more"}, " +
    "#{n-1 < 0 ? 99 : n-1 == 0 ? 'no more' : n-1} bottle#{'s' if n-1 != 1}"+
    " of beer on the wall.\n"
  end
end
```

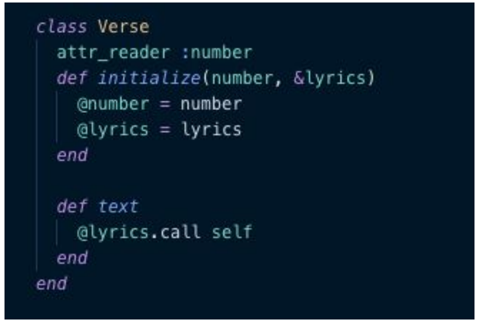
99 Bottles of Beer - Speculative / General



```ruby
class Bottles
  NoMore = lambda do |verse|
    "No more bottles of beer on the wall, " +
    "no more bottles of beer.\n" +
    "Go to the store and buy some more, " +
    "99 bottles of beer on the wall.\n"
  end

  LastOne = lambda do |verse|
    "1 bottle of beer on the wall, " +
    "1 bottle of beer.\n" +
    "Take it down and pass it around, " +
    "no more bottles of beer on the wall.\n"
  end

  Penultimate = lambda do |verse|
    "2 bottles of beer on the wall, " +
    "2 bottles of beer.\n" +
    "Take one down and pass it around, " +
    "1 bottle of beer on the wall.\n"
  end

  Default = lambda do |verse|
    "#{verse.number} bottles of beer on the wall, " +
    "#{verse.number} bottles of beer.\n" +
    "Take one down and pass it around, " +
    "#{verse.number - 1} bottles of beer on the wall.\n"
  end

  def song
    verses(99, 0)
  end
```

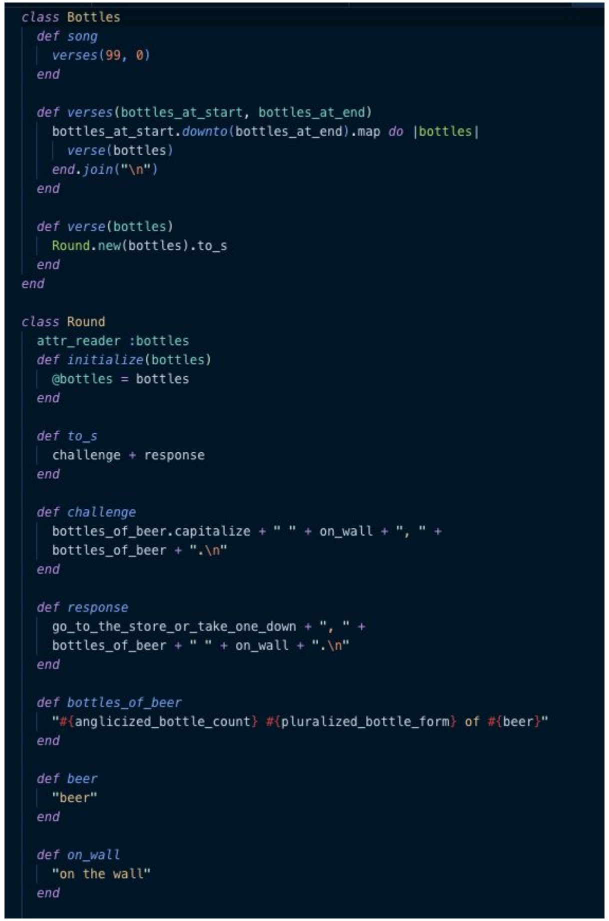


```ruby
  def verses(finish, start)
    (finish).downto(start).map {|verse_number|
      verse(verse_number) }.join("\n")
  end

  def verse(number)
    verse_for(number).text
  end

  def verse_for(number)
    case number
    when 0 then Verse.new(number, &NoMore)
    when 1 then Verse.new(number, &LastOne)
    when 2 then Verse.new(number, &Penultimate)
    else        Verse.new(number, &Default)
    end
  end
end
```

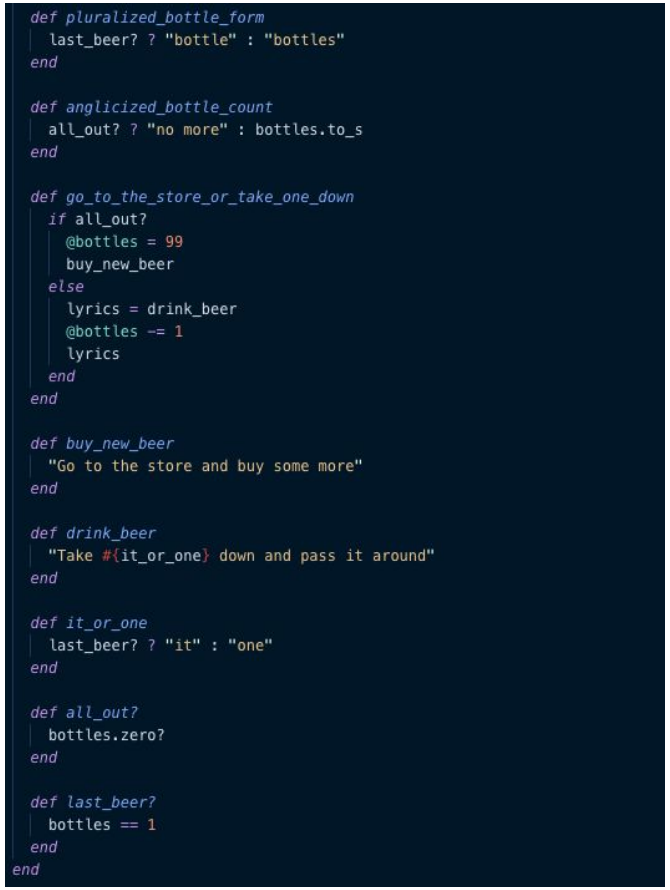


```ruby
class Verse
  attr_reader :number
  def initialize(number, &lyrics)
    @number = number
    @lyrics = lyrics
  end

  def text
    @lyrics.call self
  end
end
```

99 Bottles of Beer - Abstract

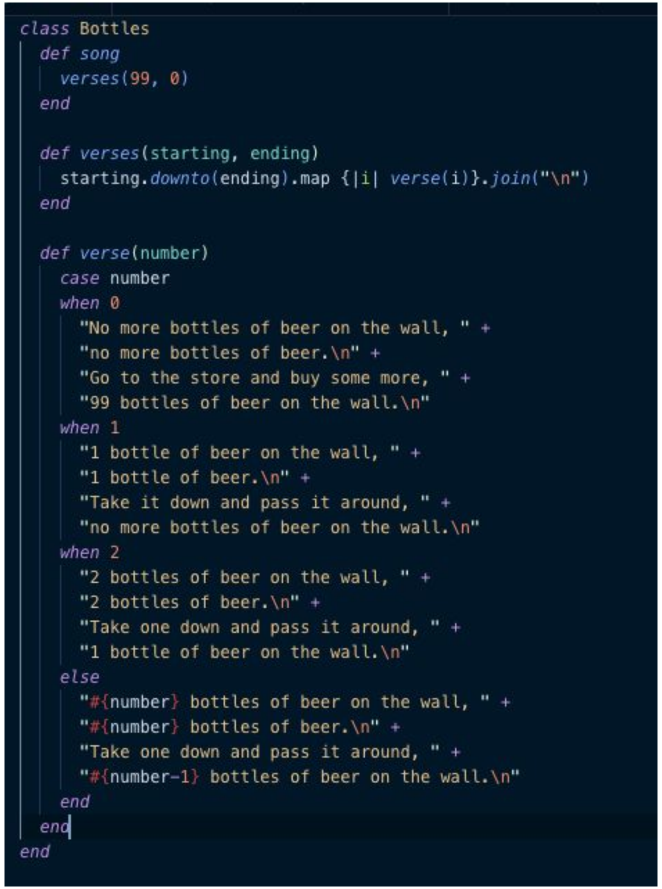


```ruby
class Bottles
  def song
    verses(99, 0)
  end

  def verses(bottles_at_start, bottles_at_end)
    bottles_at_start.downto(bottles_at_end).map do |bottles|
      verse(bottles)
    end.join("\n")
  end

  def verse(bottles)
    Round.new(bottles).to_s
  end
end

class Round
  attr_reader :bottles
  def initialize(bottles)
    @bottles = bottles
  end

  def to_s
    challenge + response
  end

  def challenge
    bottles_of_beer.capitalize + " " + on_wall + ", " +
    bottles_of_beer + ".\n"
  end

  def response
    go_to_the_store_or_take_one_down + ", " +
    bottles_of_beer + " " + on_wall + ".\n"
  end

  def bottles_of_beer
    "#{anglicized_bottle_count} #{pluralized_bottle_form} of #{beer}"
  end

  def beer
    "beer"
  end

  def on_wall
    "on the wall"
  end
```



```ruby
  def pluralized_bottle_form
    last_beer? ? "bottle" : "bottles"
  end

  def anglicized_bottle_count
    all_out? ? "no more" : bottles.to_s
  end

  def go_to_the_store_or_take_one_down
    if all_out?
      @bottles = 99
      buy_new_beer
    else
      lyrics = drink_beer
      @bottles -= 1
      lyrics
    end
  end

  def buy_new_beer
    "Go to the store and buy some more"
  end

  def drink_beer
    "Take #{it_or_one} down and pass it around"
  end

  def it_or_one
    last_beer? ? "it" : "one"
  end

  def all_out?
    bottles.zero?
  end

  def last_beer?
    bottles == 1
  end
end
```

99 Bottles of Beer - Shameless



```ruby
class Bottles
  def song
    verses(99, 0)
  end

  def verses(starting, ending)
    starting.downto(ending).map {|i| verse(i)}.join("\n")
  end

  def verse(number)
    case number
    when 0
      "No more bottles of beer on the wall, " +
      "no more bottles of beer.\n" +
      "Go to the store and buy some more, " +
      "99 bottles of beer on the wall.\n"
    when 1
      "1 bottle of beer on the wall, " +
      "1 bottle of beer.\n" +
      "Take it down and pass it around, " +
      "no more bottles of beer on the wall.\n"
    when 2
      "2 bottles of beer on the wall, " +
      "2 bottles of beer.\n" +
      "Take one down and pass it around, " +
      "1 bottle of beer on the wall.\n"
    else
      "#{number} bottles of beer on the wall, " +
      "#{number} bottles of beer.\n" +
      "Take one down and pass it around, " +
      "#{number-1} bottles of beer on the wall.\n"
    end
  end
end
```

Thanks!

Contact Nimble

nimblehq.co

hello@nimblehq.co

Bangkok

399 Interchange 21 Sukhumvit Road, Unit #2402-03, Klong Toei, Wattana, Bangkok 10110, Thailand

Singapore

28C Stanley St, Singapore 068737

Hong Kong

20th Floor, Central Tower
28 Queen's Road, Central, Hong Kong

nimble