Object Model

Classes and Instances

Classes
- store instance methods
- have a superclass pointer (this is called inheritance)

Instances
- store instance variables
- have a class pointer

Classes are also instances (of Class)

Bindings

all code executes in a binding
these store local variables
they have a self (an instance that they are operating in)
this gives them access to instance variables
they can call methods on that object
when a method is called, it puts a new binding on the stack for that invocation

Variables

local_variables
- only exist within a binding
- method parameters (often called arguments) are local variables

@instance_variable
- *only* accessible from the instance they are attached to
- (remember, instances store instance variables)

$global_variables
- accessible from anywhere

Constants
- begin with a capital letter
- use CamelCase and SCREAMING_SNAKE_CASE
- are defined within classes (if defined in main, are defined on Object)
- can be seen anywhere within that class (we call this their namespace)
- we access them with the :: operator. e.g. Float::INFINITY

Method lookup / invocation

when you invoke a method on an instance
starting at the instance's class
follow the chain of superclasses until you find the method
once you find the method, create a binding with self set to the instance
and run the code for the method in that binding

Reflection/Introspection

```
# Reflection

class Book
  def initialize(title) @title = title end  # => :initialize
  def title()           @title         end  # => :title
end

# Constants, searching arrays for meaningful results
Object.constants  # => [:Object, :Module, :Class, :BasicObject, ...

# The class (sometimes called its type)
# NOTE: You can read this book in an afternoon
# and walk away with 20 ideas about how learn more effectively
book = Book.new 'The Little Book of Talent'  # => #<Book:0x007f @title="The Little Book…
book.class                                   # => Book

# An object's instance variables
book.instance_variables             # => [:@title]
book.instance_variable_get(:@name)  # => nil

# An object's public and private methods
book.methods          # => [:title, :nil?, :===, :=~, :!~, :eql?, :hash, :<=>, :class, ...
book.private_methods  # => [:initialize, :initialize_copy, :initialize_dup, ...

# Getting the methods from the class
# (false means "don't show inherited methods")
Book.instance_methods(false)          # => [:title]
Book.instance_methods                 # => [:title, :nil?, :===, :=~, :!~, :eql?, :class, ...
Book.private_instance_methods(false)  # => [:initialize]

# What class/file/line did a method come from?
book.method(:title).owner            # => Book
book.method(:title).source_location  # => ["program.rb", 5]

# A method's `self`
book.method(:title).receiver  # => #<Book:0x007fa25c84b1f0 @title="The Little Book of Talent">

# looking at parameters
def m(a, b=1, *c, &d)end  # => :m
method(:m).parameters     # => [[:req, :a], [:opt, :b], [:rest, :c], [:block, :d]]

# list of local variables
local_variables  # => [:book]

# Useful manipulations
Object.constants.grep(/Obj/)       # => [:Object, :BasicObject, :ObjectSpace]
book.methods - Object.new.methods  # => [:title]
```

Musings

- If we think of a class as a kitchen, then defining a method is recording a recipe, calling the method is baking the cake.
- `self` is always the object a method our method was called on.
- Methods are just latent code. It is the binding which grants them life.
- Our eye colour diverges, but our blink converges.
  This is the difference between instances and classes.
- If we stored methods on instances, then a library with a million books would have a million redundant methods to retrieve a title.
- Given the way methods are looked up, we can change the behaviour of a method by defining it earlier in the inheritance chain.
- Stop right now and memorize method lookup: class, superclass, superclass, superclass, ...
- If classes were directories, inheriting from another class would be storing their files in your directory. Notice that if you put all your files in your home directory, you'd be overwhelmed by the lack of organization.
- Everything stated here is true enough to treat as fact, even though everything stated here has its exceptions.