

BEWD – Classes and Objects

Week 3 / Lesson 1

Agenda

- Allstate Commercial
- Creating Classes & Objects
- Lab Time
- It doesn't look like much but there's a lot to cover

From Hashes to Classes

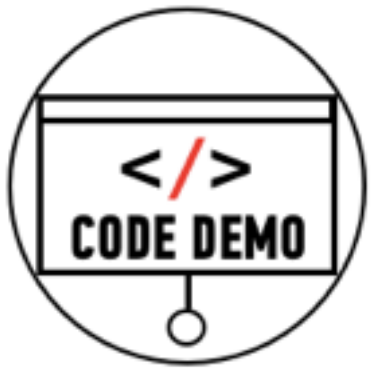
Hashes pros and cons

- PRO: Hold a bunch of related values in one place
- PRO: Easy access to values via named keys
- CON: No defaults
- CON: Cannot define your own methods within a hash
- CON: They cannot inherit properties or extend other objects

Classes & Objects

Classes to the rescue

- What is a class?
 - A class is the blueprint from which individual objects are created
- What is an object?
 - The class everything inherits from. Everything is an object. An object holds a definition for any given part of your code.
- Why/when to use them?
 - When you need to encapsulate a set of values and functions into something that's reusable.



Creating Objects

To follow along, create a NEW .rb file and paste the contents of
`Week3/Lesson1/Examples/creating_objects.rb`
into it

Creating Objects

Recap

- Adding variables to a class

```
# Hashes
story = {}
story[:title] = "Sand angry with flip-flops"
story[:title] #=> Returns your value
```

```
# With an object
class Story
  attr_accessor :title
end
```

```
story = Story.new
story.title = "Sand angry with flip-flops"
story.title #=> Returns your value
```

Creating Objects

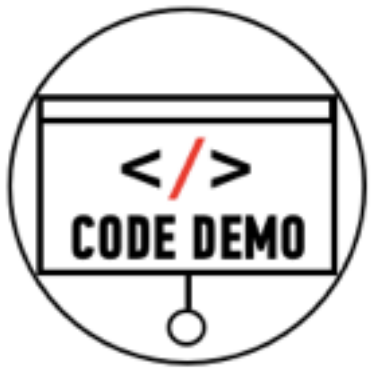
Recap

- Adding methods to a class

```
class Story
  attr_accessor :title, :category, :upvotes
  def upvote!
    @upvotes += 1
  end
end
```

```
story = Story.new
story.title = "Fruit Flies find fleas facetious"
story.category = "Turf War"
story.upvotes = 1
```

```
story.upvote!
story.upvotes #=> 2
```



Apartment

I'll create an Apartment class and you can follow along

Apartment

Recap

- The initialize method is invoked when `Apartment.new` is called
- `to_s` method called automatically on objects interpolated in a string (e.g. with `puts`)
- `to_s` can be overridden:

```
class My_Class
  def to_s
    "The puts method was called."
  end
end
```

```
>> my_object = My_Class.new
>> puts my_object
The puts method was called.
=> nil
```

Apartment

Recap

- Classes allow us to keep code DRY.
- In object oriented programs variables have scope (key scopes are local vs @instance).
 - attr_accessor allows a variable to be accessed outside of a method
- We can create class methods by using self.method_name.
 - Class methods (e.g. Apartment.new) can be called on a class (which is an object too!)

Classes & Objects

Classes in separate .rb files

Classes & Objects

Too many classes in one .rb file

```
# blt.rb  
class BLT  
  #...  
end
```

```
class Bacon  
  #...  
end
```

```
class Lettuce  
  #...  
end
```

```
class Tomato  
  #...  
end
```

Classes & Objects

Everyone Gets a File! (Like Oprah)

```
# blt.rb  
require_relative 'bacon'  
require_relative 'lettuce'  
require_relative 'tomato'
```

```
class BLT  
  #...  
end
```

Classes & Objects

Creating a link between classes in separate .rb file

- `require`
- `require_relative` (we've seen this when working with APIs)
- `$LOAD_PATH.unshift(File.dirname(FILE))` (use to load files in irb)

Lab Time

1. Apartment Objects
2. Secret Number Objects



Homework & EXIT TICKET!

- Complete the Object versions of Secret Number and Apartment (See the Homework folder)
- Fill out exit ticket (in the README for this week)

RESOURCES: Classes & Objects

Cheat Sheet

Create A Class

```
class class_name
  #variables
  and method for this
  class.
end
```

Creating Objects

```
class GA_course
  def initialize (course_name)
    @course_name = course_name
  end
```

```
    def announce_course
      puts "GA has a course on
#{@course_name}"
    end
  end
```

```
my_course = GA_course.new("BEWD")
other_course = GA_course.new("UXD")
```

```
my_course.announce_course
other_course.announce_course
```

```
GA has a course on BEWD
GA has a course on UXD
```

RESOURCES: Classes & Objects

Variable Scope

Cheat Sheet Cont.

Scope	Example	Explanation
Local	@name	Available in the same method
Instance	name	Unique value for each instance of a class available from any method in that class.
Class	@@name	Same shared value for all instances of a class, available from any method of that class.
Global	\$name	Same shared value for all code running within a single Ruby program.

Still Feel Lost?

Catch Up With These Resources

- What is Object Oriented Programming video
- What is Object Oriented Programming Book Chapter
- Introduction to Objects Ruby Monk
- Building your Own class Ruby Monk