OOP, Scope & Classes

Object Oriented Programming

What is Object Oriented Programming?

- A style of programming that:
    - Aims to replicate real life
    - Makes your code "modular"
    - Focusses on data and structure rather than logic
    - Ruby is definitely Object-Oriented!

Ruby is Object Oriented?

```ruby
# Everything in Ruby inherits!

"".class
42.class
[].class

"".class.ancestors
[].class.ancestors

"".class.superclass
```

Lots of talk about classes!

Classes

So, everything is an object

In Ruby, everything is an object that:

- Knows things (data)
- Can do things (methods)

We can create our own data types using **classes**

What are classes?

A way to create our own types of data

- It helps us reduce duplication
- It helps us to debug
- It helps us orgranise and structure our code
- They try to replicate real life

I like to think of them as **blueprints**

What do classes look like?

```
class Person
end

class Animal
end

class Vehicle
end

class Instrument
end
```

They are created with...

- The `class` keyword
- The end keyword

They must be named using UpperCamelCase!

We can add methods

```ruby
class Person
  def speak
    puts "I am now speaking"
  end

  def laugh
    puts "out loud"
  end
end
```

We can add methods

```ruby
class Person
  def speak
    puts "I am now speaking"
  end

  def laugh
    puts "out loud"
  end
end

# Create an instance
person = Person.new

# Call methods on the instance
person.speak
person.laugh
```

def initialize

```ruby
class Person
  def initialize
    puts "A new person was born!"
  end

person = Person.new
```

`initialize` will be called automatically!

Variables and Scope

Types of Variables

Local Variables

Defined in a method, and not available outside of that method Always start with a lowercase letter or an underscore

Instance Variables

Available everywhere on an **instance** of a `class` (all methods) Prefixed with an `@`

Types of Variables

Class Variables

Available throughout all **instances** of a `class` They belong to a particular class, and are often characteristics Prefixed with `@@`

Global Variables

Available everywhere Prefixed with a `$`

Types of Variables

Constants

Can never be changed All uppercase, words are seperated by underscores

`defined?`

You can always check how/if a variable is defined

Using the `defined?` method

Types of Methods

Instance Methods

Methods on an instance of a class

Class Methods

Methods on a class itself

Predicate Methods

Methods that return true or false (`2.even?` for example)

Classes

Storing Information

```ruby
# We want to store information on a person!
# We use getters and setters to do this

class Person
  def name=( name )
    @name = name
  end

  def name
    @name
  end
end

jane = Person.new
jane.name=( "Jane" )
jane.name # => "Jane"
```

There is a bit of duplication

```
class Person
  def name=( name )
    @name = name
  end

  def age=( age )
    @age = age
  end
end

jane = Person.new
jane.name = "Jane"
jane.age = 42
```

Let's make that better

```
class Person
  attr_accessor :name, :age
end

jane = Person.new
jane.name = "Jane"
jane.age = 42
```

Attr

```ruby
class Person
  attr_accessor :name, :age
end
```

```ruby
class Person
  attr_reader :name, :age
end

class Person
  attr_writer :name, :age
end
```

Attr

```ruby
class Person
  attr_accessor :name, :age
end

person = Person.new

person.name = "Name"
person.age = 42
```

Initialize

```ruby
class Person
  attr_accessor :name, :age

  def initialize( name, age )
    @name = name
    @age = age
  end
end

person = Person.new "Person", 42
```

Inheritance

```ruby
class Vehicle
  def generic_vehicle_method
  end
end

class Boat < Vehicle
  def specific_boat_method
  end
end

b = Boat.new
b.specific_boat_method
b.generic_vehicle_method
```

Resources

- Ruby for Beginners: Classes
- Launch School