You are logged in as C. O'Keefe (Logout)

Core Ruby 41st Batch

Home ▶ PORPC101-41C ▶ 3 May - 9 May ▶ Week 5 Tutorial

Navigation

Home
- My home
  - Site pages
  - My profile
  - Current course
    - PORPC101-41C
      - Participants
      - General
      - 5 April - 11 April
      - 12 April - 18 April
      - 19 April - 25 April
      - 26 April - 2 May
      - 3 May - 9 May
        - **Week 5 Tutorial**
        - Week 5: Exercises
        - Week 5: Forum
        - L5E2 Spec Change
        - Some useful Ruby methods
          - Week 5: Quiz
      - 10 May - 16 May
      - 17 May - 23 May
      - 24 May - 30 May
      - 31 May - 6 June
      - 7 June - 13 June
  - My courses

Settings

Course administration

Week 5 Tutorial

Before you begin

First complete: Week 1 to 4.

Week 5

You need to read through the following pages. While doing so, please make a note of all your doubts, queries, questions, clarifications and after you have completed all the pages, post these on the forum here. There may be some questions that relate to something that has not been mentioned or discussed by me here; you could post the same too.

- Including Other Files in Ruby
- Ruby Open Classes
- Inheritance
- Overriding Methods
- Overloading Methods

Some of the **important points to remember** after you have read through the above page are:

- The **load** method includes the named Ruby source file every time the method is executed.
- The more commonly used **require** method loads any given file only once.
- Note that you say **require 'filename'**, not **require 'filename.rb'**.
- In Ruby, classes are never closed: you can always add methods to an existing class. This applies to the classes you write as well as the standard, built-in classes. All you have to do is open up a class definition for an existing class, and the new contents you specify will be added to whatever's there.
- The benefit of inheritance is that classes lower down the hierarchy get the features of those higher up, but can also add specific features of their own.
- In Ruby, a class can only inherit from a single other class.
- The **Object** class is the parent class of all classes in Ruby. Its methods are therefore available to all objects unless explicitly overridden.
- Method overriding allows a subclass to provide a specific implementation of a method that is already provided by one of its superclasses. The implementation in the subclass overrides (replaces) the implementation in the superclass.
- Nothing stops you from defining a method twice, however the new version takes precedence.
- When you invoke **super** with no arguments Ruby sends a

My profile settings

message to the parent of the current object, asking it to invoke a method of the same name as the method invoking **super**. It automatically forwards the arguments that were passed to the method from which it's called.
- Called with an empty argument list - **super()** - it sends no arguments to the higher-up method, even if arguments were passed to the current method.
- Called with specific arguments - **super(a, b, c)** - it sends exactly those arguments.
- A Ruby class can have only one method with a given name.

Sample program covering everything we have learned so far in Classes and Objects (please ensure that all the following programs are stored in the same folder):

```
#animal.rb
class Animal
  def display
    'Class Animal: display method from Animal.'
  end
  def run
    'Class Animal: run method from Animal.'
  end
end

# dog.rb
require 'animal'
class Dog  < Animal
  def initialize(breed, name)
    @breed = breed
    @name = name
  end
  def bark
    "Class Dog's bark method: Ruff! Ruff!"
  end
  # override display
  def display
    "Class Dog's display method: I am of #{@breed}
  end
end

# cat.rb
require 'animal'
class Cat  < Animal
  def initialize(breed, name)
    @breed = breed
    @name = name
  end
  def meow
    "Class Cat's meow method: Meow! Meow!"
  end
  # override display
  def display
    "Class Cat's display method: I am of #{@breed}
  end
end
```

```ruby
# testanimals.rb
require 'dog'
require 'cat'

# create object and call methods
d = Dog.new('Labrador', 'Benzy')
c = Cat.new('Persian', 'Leo')
puts d.bark
puts d.display
puts "d object is of class: " + d.class.to_s
puts c.meow
puts c.display
puts "c object is of class: " + c.class.to_s

# mark for garbage collection
c = nil

# respond_to?
if d.respond_to?("eat")
  d.eat
else
  puts "Sorry, the object doesn't understand the '
end

# add method_missing
class Dog < Animal
  def method_missing(m, *args)
    "Class Dog: There's no method called #{m} here
  end
end

# check method_missing
puts d.eat

# object_id
puts "The id of d obj is #{d.object_id}."

# add eat method - Open classes
class Dog < Animal
  def eat
    'Class Dog method eat: I love bones!'
  end
end

puts d.eat

# test inheritance
puts d.run

# override run method
class Dog < Animal
  def run
    'This method definition is over-riden by the n
  end
  def run
    super + ", but all Dogs run very fast"
  end
end
```

```
puts d.run

# list down all methods
puts d.methods.sort
```

d. **Exercises:**

Please complete the Week 5 exercises and discuss in the Week 5 forum.

e. **Quiz:**

Please take the quiz after you have completed Week 5. All the questions are either of multiple-choice and/or true/false and based on what you have learned in Week 5. You have only 2 attempts to complete the quiz. Grades will be allotted for the quiz.