90DULES

PRESS START

NAMESPACE,

IMAGE_UTILS.RB

def preview(image)
end



pollutes global namespace

def transfer(image, destination)
end

potential conflicts with methods with same name

RUN.RB

require 'image_utils'

image = user.image
preview(image)



NAMESPACE,

IMAGE_UTILS.RB

```
module <u>ImageUtils</u>
  def self.preview(image)
  end
```



def self.transfer(image, destination)
 end
end

RUN.RB

require 'image_utils'

image = user.image
ImageUtils.preview(image)



MIXIN

IMAGE_UTILS.RB

```
module ImageUtils
  def preview
  end

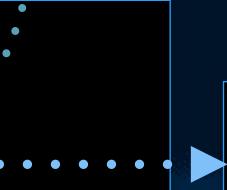
def transfer(destination)
  end
end
```



can access properties
on objects from host class

AVATAR.RB

require 'image_utils'
class Image
 include ImageUtils
end



Included as instance methods

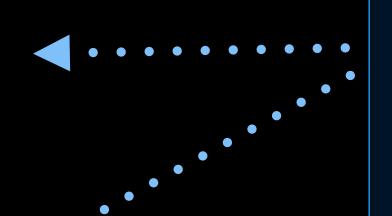
RUN.RB

image = user.image
image.preview



ANCESTORS,

class <u>Image</u>
 include ImageUtils
end



Adding module to Image's ancestors chain

Image.ancestors

[Image, ImageUtils, Object, Kernel, BasicObject]

Image.included_modules
 [ImageUtils, Kernel]





```
class Post
  def share_on_facebook
  end
end
class <u>Image</u>
  def share_on_facebook
  end
end
class Tweet
  def share_on_facebook
  end
end
```

class Shareable
end



class <u>Post</u> < Shareable
end</pre>

class <u>Image</u> < Shareable
end</pre>



class <u>Tweet</u> < Shareable
end</pre>

```
class Shareable
  def share_on_facebook
  end
end
```

A class can only have one superclass. Inheritance suggests specialization. Some behaviors are not fit for classes.



class Post
 include Shareable
end

class <u>Image</u> include Shareable end



class <u>Tweet</u>
 include Shareable
end

module Shareable
 def share_on_facebook
 end
end



class Post
 include Shareable
 include Favoritable
end

class <u>Image</u>
 include Shareable
 include Favoritable
end



class <u>Tweet</u>
 include Shareable
 include Favoritable
end

module Shareable
 def share_on_facebook
 end
end

module Favoritable
 def add_to_delicious
 end
end





class <u>Tweet</u>
 extend Searchable
end

•••••••

module <u>Searchable</u>
 def find_all_from(user)
 end
end

Included as class methods



Tweet.find_all_from('@GreggPollack')





class <u>Tweet</u>
 extend Searchable
end

use extend to expose

methods as class methods

Tweet.find_all_from('@GreggPollack')

class <u>Image</u>
 include ImageUtils
end

use include to expose methods as instance methods

image = user.image
image.preview





```
class <u>Image</u>
end
```

```
image = Image.new
image.extend(ImageUtils)
image.preview
```

module <u>ImageUtils</u>
def preview
end
end

an object is extending the module

the module will not be available in other objects

image = Image.new
image.preview

NoMethodError: undefined method `preview' for #<Image:0x10b448a98>



HOOKS - SELF.INCLUDED

```
module <a href="ImageUtils">ImageUtils</a>
  def preview
  end
  def transfer(destination)
  end
  module <u>ClassMethods</u>
     def fetch_from_twitter(user)
     end
  end
end
```

```
class <a href="mage">Image</a>
  include <a href="mageUtils">ImageUtils</a>
  extend <a href="mageUtils">ImageUtils</a>::ClassMethods
  end
```

image = user.image
image.preview

Image.fetch_from_twitter('gregg')



HOOKS - SELF.INCLUDED

```
module <a href="ImageUtils">ImageUtils</a>
  def self.included(base)
    base.extend(ClassMethods)
  end
 def preview
  end
  def transfer(destination)
  end
  module ClassMethods
    def fetch_from_twitter(user)
    end
  end
end
```

```
class <u>Image</u>
            include ImageUtils
            extend ImageUtils::ClassMethods
         end
sellfincluded is called by Ruby
when a module is included in a class
         image = user.image
         image.preview
```

Image.fetch_from_twitter('gregg')

in this case, the module name can be anything



```
module <a href="ImageUtils">ImageUtils</a>
  def self.included(base)
    base.extend(ClassMethods)
    base.clean_up
  end
  module <u>ClassMethods</u>
    def fetch_from_twitter(user)
    end
    def clean_up
    end
  end
end
```

```
class <u>Image</u>
  include ImageUtils
end
```



```
require 'active_support/concern'
module ImageUtils
  extend ActiveSupport::Concern
  included do
    clean_up
  end
  module <u>ClassMethods</u>
    def fetch_from_twitter(user)
    end
    def clean_up
    end
  end
end
```

gem install activesupport

class <u>Image</u>
 include ImageUtils
end

included block is executed in the context of the Image class

Active Support:: Concern looks for a module named Class Methods



```
module ImageUtils

def self.included(base) 
   base.extend(ClassMethods)
end

module ClassMethods
   def clean_up; end
end
end
```

```
class Image include ImageProcessing end

[mageProcessing depends]
```

```
module ImageProcessing
  def self.included(base)
    base.clean_up
    end
end
```

calls method on Image class

base is Image class



on ImageUtils

```
def self.included(base) ◀ • . .
    base.extend(ClassMethods)
  end
  module <u>ClassMethods</u>
    def clean_up; end
  end
end
module ImageProcessing
  include ImageUtils
  def self.included(base)
    base.clean_up
  end
end
```

module ImageUtils

class <u>Image</u>
 include ImageProcessing
end



base is mage Processing module

undefined method error



```
module <a href="ImageUtils">ImageUtils</a>
  def self.included(base) ◀ . .
     base.extend(ClassMethods)
  end
  module <u>ClassMethods</u>
    def clean_up; end
  end
end
module ImageProcessing
  extend ActiveSupport::Concern
  include ImageUtils
  def self.included(base)
    base.clean_up
  end
end
```

class <u>Image</u>
 include ImageProcessing
end

Dependencies are properly resolved

base is Image class



```
module ImageUtils
  extend ActiveSupport::Concern
  module ClassMethods
   def clean_up; end
  end
end
```

module ImageProcessing



```
extend ActiveSupport::Concern
include ImageUtils
included do
   clean_up
end
end
```

class <u>Image</u>
 include ImageProcessing
end

Dependencies are properly resolved



90DULES

PRESS START