Class < Module

Classes in Ruby are first-class objects—each is an instance of class Class.

When a new class is defined (typically using class *Name* . . . end), an object of type Class is created and assigned to a constant (*Name*, in this case). When Name.new is called to create a new object, the new instance method in Class is run by default, which in turn invokes allocate to allocate memory for the object, before finally calling the new object's initialize method.

Class methods

inherited *cls*.inherited(*sub_class*)

Invoked by Ruby when a subclass of *cls* is created. The new subclass is passed as a parameter.

```
class Top
  def self.inherited(sub)
    puts "New subclass: #{sub}"
  end
end
class Middle < Top
end
class Bottom < Middle
end
```

produces:

```
New subclass: Middle
New subclass: Bottom
```

new Class.new(*super_class*=Object) ⟨ { *block* } ⟩ → *cls*

Creates a new anonymous (unnamed) class with the given superclass (or Object if no parameter is given). If called with a block, that block is used as the body of the class. Within the block, self is set to the class instance.

```
name = "Dave"
FriendlyClass = Class.new do
  define_method :hello do
    "Hello, #{name}"
  end
end
f = FriendlyClass.new
f.hello   # =>   "Hello, Dave"
```

Instance methods

allocate

cls.allocate → *obj*

Allocates space for a new object of *cls*'s class. The returned object must be an instance of *cls*. Calling new is basically the same as calling the class method allocate to create an object, followed by calling initialize on that new object. You cannot override allocate in normal programs; Ruby invokes it without going through conventional method dispatch.

```
class MyClass
  def self.another_new(*args)
    o = allocate
    o.send(:initialize, *args)
    o
  end
  def initialize(a, b, c)
    @a, @b, @c = a, b, c
  end
end

mc = MyClass.another_new(4, 5, 6)
mc.inspect   # =>   "#<MyClass:0x0a34f8 @a=4, @b=5, @c=6>"
```

new

cls.new(⟨ *args* ⟩*) → *obj*

Calls allocate to create a new object of *cls*'s class and then invokes the newly created object's initialize method, passing it *args*.

superclass

cls.superclass → *super_class* or nil

Returns the superclass of *cls* or returns nil.

```
Class.superclass    # =>   Module
Object.superclass   # =>   BasicObject
```