LS 129 Organized Notes: Problem Sets

NOTE: HOW TO COPY THE CODE INTO MARKDOWN. Copy it normally, then paste without formatting (Ctrl+Shift+V) into your markdown code block. It will format as expected.

OOP, Reading OO code

Quiz lesson 5	

Classes and objects, Encapsulation, working with collaborator objects, public/private/protected methods

Using the class definition from step #3, let's create a few more people -- that is, Person objects. bob = Person.new('Robert Smith') rob = Person.new('Robert Smith') If we're trying to determine whether the two objects contain the same name, how can we compare the two objects?	Link #4
Create an empty class named Cat.	Link
Using the code from the previous exercise, create an instance of Cat and assign it to a variable named kitty.	Link
class Person ** def initialize(name)** ** @name = name** ** end** **end** **class Cat** ** def initialize(name, owner)** ** @name = name** ** @owner = owner** ** end** **end** **sara = Person.new("Sara")** **fluffy = Cat.new("Fluffy", sara)** **Identify all custom defined objects that act as collaborator objects within the**	Link #8

code.	
`` class Pet attr_reader :name def initialize(name) @name = name.to_s end def to_s @name.upcase! "My name is #{@name}." end end name = 'Fluffy' fluffy = Pet.new(name) puts fluffy.name puts fluffy puts fluffy.name puts name ```	Link
`` class Pet attr_reader :name def initialize(name) @name = name.to_s end def to_s @new_name = @name.upcase "My name is #{@new_name}." end end name = 42 fluffy = Pet.new(name) name += 1 puts fluffy.name puts fluffy puts fluffy.name ```	

puts name	

Polymorphism, inheritance, modules, method lookup path, duck-typing

What will the following code output? class Animal def initialize(name) @name = name end def speak puts sound end def sound "#{@name} says " end end class Cow < Animal def sound super + "moooooooooooo!" end end daisy = Cow.new("Daisy") daisy.speak	Link #11
class Wedding attr_reader :guests, :flowers, :songs def prepare(preparers) preparers.each do |preparer| case preparer when Chef preparer.prepare_food(guests) when Decorator preparer.decorate_place(flowers) when Musician	Link

```
        preparer.prepare_performance(songs)
      end
    end
  end
end

class Chef
  def prepare_food(guests)
    # implementation
  end
end

class Decorator
  def decorate_place(flowers)
    # implementation
  end
end

class Musician
  def prepare_performance(songs)
    #implementation
  end
end

# The above code would work, but it is problematic. What is wrong with this code, and how
can you fix it?
```

```
class Character
  attr_accessor :name

  def initialize(name)
    @name = name
  end

  def speak
    "#{@name} is speaking."
  end
end

class Knight < Character
  def name
    "Sir " + super
```

Link #4

`` end end sir_gallant = Knight.new("Gallant") sir_gallant.name # => "Sir Gallant" sir_gallant.speak # => "Sir Gallant is speaking." # What change(s) do you need to make to the above code in order to get the expected output? ```	
`` class Animal attr_accessor :name def initialize(name) @name = name end end class GoodDog < Animal def initialize(color) super @color = color end end bruno = GoodDog.new("brown") p bruno.name # What will this return, and why? ```	Link second example
`` class FarmAnimal def speak "#{self} says " end end class Sheep < FarmAnimal def speak super + "baa!" end end class Lamb < Sheep def speak "baaaaaaa!" ```	Link #6

`` end end class Cow def speak super + "moooooooo!" end end Sheep.new.speak # => "Sheep says baa!" Lamb.new.speak # => "Lamb says baa!baaaaaaa!" Cow.new.speak # => "Cow says moooooooo!" # Make the changes necessary in order for this code to return the expected values. ```	
`` class Person def get_name @name # the @name instance variable is not initialized anywhere end end bob = Person.new bob.get_name # => ?? # What is the return value, and why? ```	Link
`` module Swim def enable_swimming @can_swim = true end end class Dog include Swim def swim "swimming!" if @can_swim end end teddy = Dog.new teddy.swim # How do you get this code to return “swimming”? What does this demonstrate about instance variables? ```	Link

class Vehicle ** @@wheels = 4** ** def self.wheels** ** @@wheels** ** end** **end** **Vehicle.wheels # => ??** **class Motorcycle < Vehicle** ** @@wheels = 2** **end** **Motorcycle.wheels # => ??** **Vehicle.wheels # => ??** **class Car < Vehicle** **end** **Car.wheels # => ??** **# What would the above code return, and why?**	Link
module Maintenance def change_tires "Changing #{WHEELS} tires." end end class Vehicle WHEELS = 4 end class Car < Vehicle include Maintenance end a_car = Car.new a_car.change_tires # Describe the error and provide two different ways to fix it.	Link
# Using the following code, allow Truck to accept a second argument upon instantiation.	Link

Name the parameter bed_type and implement the modification so that Car continues to only accept one argument. `class Vehicle` `  attr_reader :year` `  def initialize(year)` `    @year = year` `  end` `end` `class Truck < Vehicle` `end` `class Car < Vehicle` `end` `truck1 = Truck.new(1994, 'Short')` `puts truck1.year` `puts truck1.bed_type`	
# Given the following code, modify #start_engine in Truck by appending 'Drive fast, please!' to the return value of #start_engine in Vehicle. The 'fast' in 'Drive fast, please!' should be the value of speed. `class Vehicle` `  def start_engine` `    'Ready to go!'` `  end` `end` `class Truck < Vehicle` `  def start_engine(speed)` `  end` `end` `truck1 = Truck.new` `puts truck1.start_engine('fast')` # Expected output:	Link

`# Ready to go! Drive fast, please!`	
`module Speed` `  def go_fast` `  puts "I am a #{self.class} and going super fast!"` `  end` `end` `class Car` `  include Speed` `  def go_slow` `  puts "I am safe and driving slow."` `  end` `end` `# When we called the go_fast method from an instance of the Car class (as shown below) you might have noticed that the string printed when we go fast includes the name of the type of vehicle we are using. How is this done?`	Link #3
`module Drivable` `  def self.drive` `  "is this possible"` `  end` `end` `class Car` `  include Drivable` `end` `p Car.drive` `# What will this return, and why?`	Link
`module EmailFormatter` `  def email` `  "#{first_name}.#{last_name}@#{domain}"` `  end` `end` `module EmailSender` `  def email(msg, sender, recipient)`	

```
    # contrived implementation for now
    puts "Delivering email to #{recipient} from #{sender} with message: #{msg}"
  end
end

class User
  attr_accessor :first_name, :last_name, :domain
  include EmailFormatter
  include EmailSender
end

u = User.new
u.first_name = "John"
u.last_name = "Smith"
u.domain = "example.com"

p u.email
```

https://launchschool.com/exercises/05ac9b2b

Use attr_* to create setter and getter methods, How to call setters and getters, Referencing and setting instance variables vs. using getters and setters

```
class GoodDog
  attr_accessor :name, :height, :weight

  def initialize(n, h, w)
    @name = n
    @height = h
    @weight = w
  end

  def speak
    "#{name} says arf!"
  end

  def change_info(n, h, w)
    name = n
    height = h
```

Link

weight = w end def info "#{name} weighs #{weight} and is #{height} tall." end end sparky.change_info('Spartacus', '24 inches', '45 lbs') puts sparky.info # => Sparky weighs 10 lbs and is 12 inches tall. # Why does the .change_info method not work as expected here?	
class Person attr_writer :first_name, :last_name def full_name # omitted code end end mike = Person.new mike.first_name = 'Michael' mike.last_name = 'Garcia' mike.full_name # => 'Michael Garcia' What code snippet can replace the "omitted code" comment to produce the indicated result?	Link #15
class Student attr_accessor :name, :grade def initialize(name) @name = name @grade = nil end end priya = Student.new("Priya") priya.change_grade('A') priya.grade # => "A"	Link #16

The last line in the above code should return "A". Which method(s) can we add to the Student class so the code works as expected?	
In the example above, why would the following not work? `def change_grade(new_grade)` `  grade = new_grade` `end`	Link #16
Given the below usage of the Person class, code the class definition. `bob = Person.new('bob')` `bob.name  # => 'bob'` `bob.name = 'Robert'` `bob.name  # => 'Robert'`	Link #1
Modify the class definition from above to facilitate the following methods. Note that there is no name= setter method now. `bob = Person.new('Robert')` `bob.name  # => 'Robert'` `bob.first_name  # => 'Robert'` `bob.last_name  # => ''` `bob.last_name = 'Smith'` `bob.name  # => 'Robert Smith'` Hint: let first_name and last_name be "states" and create an instance method called name that uses those states.	Link #2
Now create a smart name= method that can take just a first name or a full name, and knows how to set the first_name and last_name appropriately. `bob = Person.new('Robert')` `bob.name  # => 'Robert'` `bob.first_name  # => 'Robert'` `bob.last_name  # => ''` `bob.last_name = 'Smith'` `bob.name  # => 'Robert Smith'` `bob.name = "John Adams"` `bob.first_name  # => 'John'` `bob.last_name  # => 'Adams'`	Link #3

`` class Animal def initialize(name) @name = name end end class Dog < Animal def initialize(name); end def dog_name "bark! bark! #{@name} bark! bark!" end end teddy = Dog.new("Teddy") puts teddy.dog_name # => ?? # What will this return, and why? ```	Link
`` class Cat attr_accessor :name def initialize(name) @name = name end def rename(new_name) name = new_name end end kitty = Cat.new('Sophie') p kitty.name # "Sophie" kitty.rename('Chloe') p kitty.name # "Chloe" # What is wrong with the code above? Why? What principle about getter/setter methods does this demonstrate? ```	Link
`` class Cat attr_accessor :type, :age def initialize(type) @type = type ```	Link #8

@age = 0 end def make_one_year_older self.age += 1 end end You can see in the make_one_year_older method we have used self. What does self refer to here?	

Instance methods vs. class methods, self, Calling methods with self, More about self, to_s, overriding to_s

On which lines in the following code does self refer to the instance of the MeMyselfAndI class referenced by i rather than the class itself? Select all that apply. class MeMyselfAndI self def self.me self end def myself self end end i = MeMyselfAndI.new	Link #19
Continuing with our Person class definition, what does the below print out? bob = Person.new("Robert Smith") puts "The person's name is: #{bob}"	Link #5a
Let's add a to_s method to the class: class Person	Link #5b

`  # ... rest of class omitted for brevity` `  def to_s` `    name` `  end` `end` `Now, what does the below output?` `bob = Person.new("Robert Smith")` `puts "The person's name is: #{bob}"`	
`class Student` `  attr_accessor :grade` `  def initialize(name, grade=nil)` `    @name = name` `  end` `end` `ade = Student.new('Adewale')` `ade # => #<Student:0x00000002a88ef8 @grade=nil, @name="Adewale">` `# Why does this code not have the expected return value?`	Link #2, D

Fake operators and equality

`arr1 = [1, 2, 3]` **`arr2 = [1, 2, 3]`** **`arr1.object_id == arr2.object_id      # => ??`** **`sym1 = :something`** **`sym2 = :something`** **`sym1.object_id == sym2.object_id      # => ??`** **`int1 = 5`** **`int2 = 5`** **`int1.object_id == int2.object_id      # => ??`**	Link

# What will the code above return and why?	
`` class Person attr_accessor :name, :age def initialize(name, age) @name = name @age = age end End bob = Person.new("Bob", 49) kim = Person.new("Kim", 33) puts "bob is older than kim" if bob > kim # How can you make this code function? How is this possible? ```	Link
`` my_hash = {a: 1, b: 2, c: 3} my_hash << {d: 4} # What happens here, and why? ```	Link
`` class Team attr_accessor :name, :members def initialize(name) @name = name @members = [] end def <<(person) members.push person end def +(other_team) members + other_team.members end end # we'll use the same Person class from earlier cowboys = Team.new("Dallas Cowboys") cowboys << Person.new("Troy Aikman", 48) ```	Link

```
niners = Team.new("San Francisco 49ers")
niners << Person.new("Joe Montana", 59)
dream_team = cowboys + niners                # what is dream_team?
# What does the Team#+ method currently return? What is the problem with this? How could you
fix this problem?
```