|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| LS 120 Questions   |  |  |  | | --- | --- | --- | |  | **What is OOP and why is it important?** | [Link](https://launchschool.com/books/oo_ruby/read/the_object_model) | |  | **What is encapsulation?** | [Link](https://launchschool.com/books/oo_ruby/read/the_object_model) | |  | **How does encapsulation relate to the public interface of a class?** | [Link](https://launchschool.com/lessons/dfff5f6b/assignments/8c6b8604) | |  | What is an object? | [Link](https://launchschool.com/books/oo_ruby/read/the_object_model#whatareobjects) | |  | What is a class? | [Link](https://launchschool.com/books/oo_ruby/read/the_object_model#classesdefineobjects) | |  | What is instantiation? | [Link](https://launchschool.com/books/oo_ruby/read/the_object_model#classesdefineobjects) | |  | **What is polymorphism?** | [Link](https://launchschool.com/books/oo_ruby/read/the_object_model), [link](https://launchschool.com/lessons/dfff5f6b/assignments/8c6b8604) | |  | **Explain two different ways to implement polymorphism.** | [Link](https://launchschool.com/lessons/dfff5f6b/assignments/8c6b8604) | |  | How does polymorphism work in relation to the public interface? | [Link](https://launchschool.com/lessons/dfff5f6b/assignments/8c6b8604) | |  | **What is duck typing? How does it relate to polymorphism - what problem does it solve?** | [Link](https://launchschool.com/lessons/dfff5f6b/assignments/8c6b8604) | |  | What is inheritance? | [Link](https://launchschool.com/books/oo_ruby/read/the_object_model), [link](https://launchschool.com/books/oo_ruby/read/inheritance) | |  | What is the difference between a superclass and a subclass? | [Link](https://launchschool.com/books/oo_ruby/read/the_object_model) | |  | What is a module? | [Link](https://launchschool.com/books/oo_ruby/read/the_object_model), [link](https://launchschool.com/books/oo_ruby/read/the_object_model#modules) | |  | What is a mixin? | [Link](https://launchschool.com/books/oo_ruby/read/the_object_model), [link](https://launchschool.com/books/oo_ruby/read/the_object_model#modules) | |  | When is it good to use inheritance? | [Link](https://launchschool.com/lessons/dfff5f6b/assignments/69729798) #1 | |  | **In inheritance, when would it be good to override a method?** | [Link](https://launchschool.com/lessons/dfff5f6b/assignments/69729798) #1 | |  | **What is the method lookup path?** | [Link](https://launchschool.com/books/oo_ruby/read/the_object_model#methodlookup), [Link](https://launchschool.com/lessons/dfff5f6b/assignments/69729798) #4 | |  | **When defining a class, we usually focus on state and behaviors. What is the difference between these two concepts?** | [Link](https://launchschool.com/books/oo_ruby/read/classes_and_objects_part1#statesandbehaviors) | |  | How do you initialize a new object? | [Link](https://launchschool.com/books/oo_ruby/read/classes_and_objects_part1#initializinganewobject) | |  | What is a constructor method? | [Link](https://launchschool.com/books/oo_ruby/read/classes_and_objects_part1#initializinganewobject) | |  | What is an instance variable, and how is it related to an object? | [Link](https://launchschool.com/books/oo_ruby/read/classes_and_objects_part1#instancevariables) | |  | What is an instance method? | [Link](https://launchschool.com/books/oo_ruby/read/classes_and_objects_part1#instancemethods) | |  | **How do objects encapsulate state?** | [Link](https://launchschool.com/lessons/dfff5f6b/assignments/4228f149) | |  | What is the difference between classes and objects? | [Link](https://launchschool.com/lessons/dfff5f6b/assignments/4228f149) | |  | How can we expose information about the state of the object using instance methods? | [Link](https://launchschool.com/books/oo_ruby/read/classes_and_objects_part1#instancemethods) | |  | **What is a collaborator object, and what is the purpose of using collaborator objects in OOP?** | [Link](https://launchschool.com/lessons/dfff5f6b/assignments/4228f149) | |  | **What is an accessor method?** | [Link](https://launchschool.com/books/oo_ruby/read/classes_and_objects_part1#accessormethods) | |  | What is a getter method? | [Link](https://launchschool.com/books/oo_ruby/read/classes_and_objects_part1#accessormethods) | |  | What is a setter method? | [Link](https://launchschool.com/books/oo_ruby/read/classes_and_objects_part1#accessormethods) | |  | What is attr\_accessor? | [Link](https://launchschool.com/books/oo_ruby/read/classes_and_objects_part1#accessormethods) | |  | **How do you decide whether to reference an instance variable or a getter method?** | [Link](https://launchschool.com/books/oo_ruby/read/classes_and_objects_part1#accessormethodsinaction) | |  | **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**  **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?** | [Link](https://launchschool.com/books/oo_ruby/read/classes_and_objects_part1#callingmethodswithself) | |  | When would you call a method with self? | [Link](https://launchschool.com/books/oo_ruby/read/classes_and_objects_part1#callingmethodswithself) | |  | What are class methods? | [Link](https://launchschool.com/books/oo_ruby/read/classes_and_objects_part2#classmethods) | |  | What is the purpose of a class variable? | [Link](https://launchschool.com/books/oo_ruby/read/classes_and_objects_part2#classvariables) | |  | What is a constant variable? | [Link](https://launchschool.com/books/oo_ruby/read/classes_and_objects_part2#constants) | |  | What is the default to\_s method that comes with Ruby, and how do you override this? | [Link](https://launchschool.com/books/oo_ruby/read/classes_and_objects_part2#theto_smethod) | |  | What are some important attributes of the to\_s method? | [Link](https://launchschool.com/books/oo_ruby/read/classes_and_objects_part2#theto_smethod) | |  | From within a class, when an instance method uses self, what does it reference? | [Link](https://launchschool.com/books/oo_ruby/read/classes_and_objects_part2#moreaboutself) | |  | What happens when you use self inside a class but outside of an instance method? | [Link](https://launchschool.com/books/oo_ruby/read/classes_and_objects_part2#moreaboutself) | |  | Why do you need to use self when calling private setter methods? | [Link](https://launchschool.com/lessons/dfff5f6b/assignments/8c6b8604) | |  | **Why use self, and how does self change depending on the scope it is used in?** | [Link](https://launchschool.com/books/oo_ruby/read/classes_and_objects_part2#moreaboutself) | |  | What is inheritance, and why do we use it? | [Link](https://launchschool.com/books/oo_ruby/read/inheritance#classinheritance) | |  | Give an example of how to use class inheritance. | [Link](https://launchschool.com/books/oo_ruby/read/inheritance#classinheritance) | |  | Give an example of overriding. When would you use it? | [Link](https://launchschool.com/books/oo_ruby/read/inheritance#classinheritance) | |  | Give an example of using the super method. When would we use it? | [Link](https://launchschool.com/books/oo_ruby/read/inheritance#super) | |  | Give an example of using the super method with an argument. | [Link](https://launchschool.com/books/oo_ruby/read/inheritance#super) | |  | When creating a hierarchical structure, under what circumstance would a module be useful? | [Link](https://launchschool.com/books/oo_ruby/read/inheritance#mixinginmodules) | |  | What is interface inheritance, and under what circumstance would it be useful in comparison to class inheritance? | [Link](https://launchschool.com/books/oo_ruby/read/inheritance#inheritancevsmodules) | |  | How is the method lookup path affected by module mixins and class inheritance? | [Link](https://launchschool.com/books/oo_ruby/read/inheritance#methodlookup) | |  | What is namespacing? | [Link](https://launchschool.com/books/oo_ruby/read/inheritance#moremodules) | |  | How does Ruby provide the functionality of multiple inheritance? | [Link](https://launchschool.com/lessons/dfff5f6b/assignments/2cf31cc8) | |  | Describe the use of modules as containers. | [Link](https://launchschool.com/books/oo_ruby/read/inheritance#moremodules) | |  | Why should a class have as few public methods as possible? | [Link](https://launchschool.com/lessons/dfff5f6b/assignments/8c6b8604) | |  | What is the private method call used for? | [Link](https://launchschool.com/books/oo_ruby/read/inheritance#privateprotectedandpublic) | |  | What is the protected keyword used for? | [Link](https://launchschool.com/books/oo_ruby/read/inheritance#privateprotectedandpublic) | |  | What are two rules of protected methods? | [Link](https://launchschool.com/books/oo_ruby/read/inheritance#privateprotectedandpublic) | |  | Why is it generally a bad idea to override methods from the Object class, and which method is commonly overridden? | [Link](https://launchschool.com/books/oo_ruby/read/inheritance#accidentalmethodoverriding) | |  | What is the relationship between a class and an object? | [Link](https://launchschool.com/lessons/b5948548/assignments/cd1f50f3) | |  | Explain the idea that a class groups behaviors. | [Link](https://launchschool.com/lessons/b5948548/assignments/cd1f50f3) | |  | Objects do not share state between other objects, but do share behaviors | [Link](https://launchschool.com/lessons/b5948548/assignments/cd1f50f3) | |  | The values in the objects' instance variables (states) are different, but they can call the same instance methods (behaviors) defined in the class. | [Link](https://launchschool.com/lessons/b5948548/assignments/cd1f50f3) | |  | Classes also have behaviors not for objects (class methods). | [Link](https://launchschool.com/lessons/b5948548/assignments/cd1f50f3) | |  | sub-classing from parent class. Can only sub-class from 1 parent; used to model hierarchical relationships | [Link](https://launchschool.com/lessons/b5948548/assignments/cd1f50f3) | |  | mixing in modules. Can mix in as many modules as needed; Ruby's way of implementing multiple inheritance | [Link](https://launchschool.com/lessons/b5948548/assignments/cd1f50f3) | |  | understand how sub-classing or mixing in modules affects the method lookup path | [Link](https://launchschool.com/lessons/b5948548/assignments/cd1f50f3) | |  | 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](https://launchschool.com/quizzes/eb627910) | |  | 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](https://launchschool.com/quizzes/eb627910) | |  | 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"  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? | [Link #16](https://launchschool.com/quizzes/eb627910) | |  | In the example above, why would the following not work?  def change\_grade(new\_grade)  grade = new\_grade  end | [Link #16](https://launchschool.com/quizzes/eb627910) | |  | 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](https://launchschool.com/quizzes/eb627910) | |  | 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](https://launchschool.com/lessons/dfff5f6b/assignments/209fc900) | |  | 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 #](https://launchschool.com/lessons/dfff5f6b/assignments/209fc900)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 #](https://launchschool.com/lessons/dfff5f6b/assignments/209fc900)3 | |  | 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 #](https://launchschool.com/lessons/dfff5f6b/assignments/209fc900)4 | |  | 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 #](https://launchschool.com/lessons/dfff5f6b/assignments/209fc900)5a | |  | Let's add a to\_s method to the class:  class Person  # ... 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}" | [Link #](https://launchschool.com/lessons/dfff5f6b/assignments/209fc900)5b | |  | Create an empty class named Cat. | [Link](https://launchschool.com/exercises/db363aba) | |  | Using the code from the previous exercise, create an instance of Cat and assign it to a variable named kitty. | [Link](https://launchschool.com/exercises/ce73d7fc) | |  | 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  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? | [Link](https://launchschool.com/lessons/dfff5f6b/assignments/8c6b8604) | |  | What happens when you call the p method on an object? And the puts method? | [Link](https://medium.com/launch-school/towards-a-conceptual-model-of-object-oriented-programming-118eb971659f) | |  | What is a spike? | [Link](https://launchschool.com/lessons/dfff5f6b/assignments/d632a90f) | |  | When writing a program, what is a sign that you’re missing a class? | [Link](https://launchschool.com/lessons/dfff5f6b/assignments/d632a90f) | |  | **What are some rules/guidelines when writing programs in OOP?** | [Link](https://launchschool.com/lessons/dfff5f6b/assignments/d632a90f) | |  | 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](https://launchschool.com/quizzes/bfe9b33f) #2, D | |  | 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  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? | [Link](https://launchschool.com/quizzes/bfe9b33f) #4 | |  | class FarmAnimal  def speak  "#{self} says "  end  end  class Sheep < FarmAnimal  def speak  super + "baa!"  end  end  class Lamb < Sheep  def speak  "baaaaaaa!"  end  end  class Cow  def speak  super + "mooooooo!"  end  end  Sheep.new.speak # => "Sheep says baa!"  Lamb.new.speak # => "Lamb says baa!baaaaaaa!"  Cow.new.speak # => "Cow says mooooooo!"  # Make the changes necessary in order for this code to return the expected values. | [Link](https://launchschool.com/quizzes/bfe9b33f) #6 | |  | **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 code.** | [Link](https://launchschool.com/quizzes/bfe9b33f) #8 | |  | How does equivalence work in Ruby? | [Link](https://launchschool.com/lessons/d2f05460/assignments/9cadd494) | |  | How do you determine if two variables actually point to the same object? | [Link](https://launchschool.com/lessons/d2f05460/assignments/9cadd494) | |  | What is == in Ruby? How does == know what value to use for comparison? | [Link](https://launchschool.com/lessons/d2f05460/assignments/9cadd494) | |  | Is it possible to compare two objects of different classes? | [Link](https://launchschool.com/lessons/d2f05460/assignments/9cadd494) | |  | What do you get “for free” when you define a == method? | [Link](https://launchschool.com/lessons/d2f05460/assignments/9cadd494) | |  | **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 # => ??**  **# What will the code above return and why?** | [Link](https://launchschool.com/lessons/d2f05460/assignments/9cadd494) | |  | What is the === method? | [Link](https://launchschool.com/lessons/d2f05460/assignments/9cadd494) | |  | What is the eql? method? | [Link](https://launchschool.com/lessons/d2f05460/assignments/9cadd494) | |  | What is the scoping rule for instance variables? | [Link](https://launchschool.com/lessons/d2f05460/assignments/b4f9e5b7) | |  | **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](https://launchschool.com/lessons/d2f05460/assignments/b4f9e5b7) | |  | **What are the scoping rules for class variables? What are the two main behaviors of class variables?** | [Link](https://launchschool.com/lessons/d2f05460/assignments/b4f9e5b7) | |  | What are the scoping rules for constant variables? | [Link](https://launchschool.com/lessons/d2f05460/assignments/b4f9e5b7) | |  | How does sub-classing affect instance variables? | [Link](https://launchschool.com/lessons/d2f05460/assignments/b8928e96) | |  | 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](https://launchschool.com/lessons/d2f05460/assignments/b8928e96) | |  | **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](https://launchschool.com/lessons/d2f05460/assignments/b8928e96) | |  | Are class variables accessible to sub-classes? | [Link](https://launchschool.com/lessons/d2f05460/assignments/b8928e96) | |  | Why is it recommended to avoid the use of class variables when working with inheritance? | [Link](https://launchschool.com/lessons/d2f05460/assignments/b8928e96) | |  | **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](https://launchschool.com/lessons/d2f05460/assignments/b8928e96) | |  | Is it possible to reference a constant defined in a different class? | [Link](https://launchschool.com/lessons/d2f05460/assignments/b8928e96) | |  | What is the namespace resolution operator? | [Link](https://launchschool.com/lessons/d2f05460/assignments/b8928e96) | |  | How are constants used in inheritance? | [Link](https://launchschool.com/lessons/d2f05460/assignments/b8928e96) | |  | 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](https://launchschool.com/lessons/d2f05460/assignments/b8928e96) | |  | What is lexical scope? | [Link](https://launchschool.com/lessons/d2f05460/assignments/b8928e96) | |  | When dealing with code that has modules and inheritance, where does constant resolution look first? | [Link](https://launchschool.com/lessons/d2f05460/assignments/b8928e96) | |  | 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](https://launchschool.com/lessons/d2f05460/assignments/9a7db2ee) | |  | my\_hash = {a: 1, b: 2, c: 3}  my\_hash << {d: 4}  # What happens here, and why? | [Link](https://launchschool.com/lessons/d2f05460/assignments/9a7db2ee) | |  | When do shift methods make the most sense? | [Link](https://launchschool.com/lessons/d2f05460/assignments/9a7db2ee) | |  | 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)  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? | [Link](https://launchschool.com/lessons/d2f05460/assignments/9a7db2ee) | |  | **Explain how the element getter (reference) and setter methods work, and their corresponding syntactical sugar.** | [Link](https://launchschool.com/lessons/d2f05460/assignments/9a7db2ee) | |  | How is defining a class different from defining a method? | [Link](https://launchschool.com/exercises/db363aba) | |  | How do you create an instance of a class? By calling the class method `new` | [Link](https://launchschool.com/exercises/ce73d7fc) | |  | What are two different ways that the getter method allows us to invoke the method in order to access an instance variable?  `getter\_method\_name` || `self.getter\_method\_name` | [Link](https://launchschool.com/exercises/18da9ce6) | |  | When you have a mixin and you use a ruby shorthand accessor method, how do you write the code (what order do you write the getter/setters and the mixin)? What about using a constant?  mixin then accessor, constant then accessor | [Link](https://launchschool.com/exercises/6d5fd262), [Link](https://launchschool.com/exercises/5835e7ee) | |  | How do you define a class method?  Define a class method by prepending `self` to the method name. | [Link](https://launchschool.com/exercises/96fae1ac) | |  | 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?  In the imethod `rename`, we need to prepend `self` to `name` on line 9, otherwise Ruby assumes we’re initializing a new local variable `name` and assigning it to the argument passed in through the parameter `name`. | [Link](https://launchschool.com/exercises/f00e08d5) | |  | Self refers to the \_\_\_\_\_\_ \_\_\_\_\_\_\_. calling object | [Link](https://launchschool.com/exercises/09447cd6) | |  | How do you print the object so you can see the instance variables and their values along with the object?  p object | [Link](https://launchschool.com/exercises/09447cd6) | |  | When writing the name of methods in normal/markdown text, how do you write the name of an instance method? A class method?  `ClassName#instance\_method\_name`, `ClassName::class\_method\_name` | [Link](https://launchschool.com/exercises/48a171b0) | |  | How do you override the to\_s method? What does the to\_s method have to do with puts?  You can override the to\_s method by defining a to\_s method in the relevant class. `puts` automatically calls `to\_s` when outputting an object. | [Link](https://launchschool.com/exercises/e7ca751c) | |  | # Using the following code, allow Truck to accept a second argument upon instantiation. 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  def initialize(year, bed\_type)  super(year)  @bed\_type = bed\_type  end  class Car < Vehicle  end  truck1 = Truck.new(1994, 'Short')  puts truck1.year  puts truck1.bed\_type | [Link](https://launchschool.com/exercises/2a89824f) | |  | # 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)  super() + “Go #{speed} please!”  end  end  truck1 = Truck.new  puts truck1.start\_engine('fast')  # Expected output:  # Ready to go! Drive fast, please! | [Link](https://launchschool.com/exercises/a69454cc) | |  | When do you use empty parentheses with super?  When you want to invoke a superclass methods and explicitly pass no arguments to the superclass method (to prevent an argument error) | [Link](https://launchschool.com/exercises/a69454cc) | |  | How do you find the lookup path for a class? (lookup path stops when you find it)  Call the ancestors method on the class | [Link](https://launchschool.com/exercises/ca4598d3), [Link](https://launchschool.com/exercises/e90ae9a7), [Link](https://launchschool.com/exercises/fa38af89) | |  | What is namespacing, and how do you instantiate a class contained in a module?  Namespacing is grouping related classes, perhaps to prevent similarly named methods from colliding. You can instantiate a class contained in a module by using the namespace resolution operator, :: (Module::Class.new) | [Link](https://launchschool.com/exercises/507a7449) | |  | When using getters and setters, in what scenario might you decide to only use a getter, and why is this important?  You might only need a getter if you only want to access the data, but don’t want or need to be able to change it. | [Link](https://launchschool.com/exercises/be7a7323) | |  | When might it make sense to format the data or prevent destructive method calls changing the data by using a custom getter or setter method?  Any time you want to control how the user is able to access or change data - getters and setters protect the raw data | [Link](https://launchschool.com/exercises/fd9cf084), [Link](https://launchschool.com/exercises/a04f07e7), [Link](https://launchschool.com/exercises/ae80cee3), [Link](https://launchschool.com/exercises/91ee1dc4), [Link](https://launchschool.com/exercises/dc12d4a4) | |  |  |  | |