LS 129 Organized Notes

OOP, reading OO code

What is OOP and why is it important?	Link
What is a spike?	Link
When writing a program, what is a sign that you're missing a class?	Link
What are some rules/guidelines when writing programs in OOP?	Link

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

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

What is encapsulation? How does encapsulation relate to the public interface of a class?	Link, Link
What is an object? How do you initialize a new object/ How do you create an instance of a class? What is instantiation? What is a constructor method? What is an instance variable, and how is it related to an object? What is an instance method? What is the scoping rule for instance variables?	Link, Link / Link, Link, Link, Link, Link, Link
How do you see if an object has instance variables?	Link #5
What is a class? What is the relationship between a class and an object? How is defining a class different from defining a method?	Link, Link, Link
When defining a class, we usually focus on state and behaviors. What is the difference between these two concepts? Objects do not share state between other objects, but do share behaviors The values in the objects' instance variables (states) are different, but they can call the same instance methods (behaviors) defined in the class. Explain the idea that a class groups behaviors.	Link, Link, Link, Link
How do objects encapsulate state?	Link
What is the difference between classes and objects?	Link
How can we expose information about the state of the object using instance methods?	Link
What is a collaborator object, and what is the purpose of using collaborator objects in	Link

OOP?	
Why should a class have as few public methods as possible?	Link
What is the private method call used for?	Link
What is the protected method used for?	Link
What are two rules of protected methods?	Link
Classes also have behaviors not for objects (class methods). How do you define a class method?	Link, Link
When writing the name of methods in normal/markdown text, how do you write the name of an instance method? A class method?	Link

Polymorphism, inheritance, method lookup path, duck-typing

What is polymorphism? Explain two different ways to implement polymorphism. How does polymorphism work in relation to the public interface?	Link, Link Link Summary
What is duck typing? How does it relate to polymorphism - what problem does it solve?	Link
What is inheritance? What is the difference between a superclass and a subclass? When is it good to use inheritance? Give an example of how to use class inheritance. Give an example of using the super method, both with and without an argument. Give an example of overriding: when would you use it? In inheritance, when would it be good to override a method? **In inheritance, when would it be good to override a method?**	Link, Link, Link, Link #1, Link, Link, Link (super), Link (super with an argument). Link Super: Link super without arguments, Link super with some arguments, Link super with () empty parentheses Link #1 Link #1
Accidental method overriding	Link
What is a module? What is a mixin? When creating a hierarchical structure, under what	Link, Link, Link, link, Link,

circumstance would a module be useful? What is interface inheritance, and under what circumstance would it be useful in comparison to class inheritance? What is namespacing? Describe the use of modules as containers. How does Ruby provide the functionality of multiple inheritance? What is namespacing, and how do you instantiate a class contained in a module?	Link, Link Link Link Link
Why should methods in mixin modules be defined without using self. in the definition?	Link
What is the method lookup path? How is the method lookup path affected by module mixins and class inheritance? **How do you find the lookup path for a class? (lookup path stops when you find it)**	Link, Link #4, Link Link, Link, Link
Are class variables accessible to subclasses? Why is it recommended to avoid the use of class variables when working with inheritance?	Link Link
Is it possible to reference a constant defined in a different class? How are constants used in inheritance? What is lexical scope? When dealing with code that has modules and inheritance, where does constant resolution look first?	Link Link Link Link lexical scope in depth Link
What is the namespace resolution operator?	Link

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

Using getters and setters	
What is an accessor method?	Link Link
What is a getter method?	Link he added an `attr_reader` for the balance instance variable. This means that Ruby will automatically create a method called `balance` that

	returns the value of the `@balance` instance variable. https://launchschool.com/lessons/f1c58be0/assignments/652f8d69 #1
What is a setter method? What does a setter method return?	Link https://launchschool.com/lessons/f1c58be0/assignments/652f8d69 #2 https://launchschool.com/lessons/f1c58be0/assignments/652f8d69 #3 downside of using https://launchschool.com/lessons/f1c58be0/assignments/652f8d69 #6 Always returns argument passed in Link
What is attr_accessor?	Link
How do you decide whether to reference an instance variable or a getter method?	Link
What are two different ways that the getter method allows us to invoke the method in order to access an instance variable?	Link
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?	Link, Link
When using getters and setters, in what scenario might you decide to only use a getter, and why is this important?	Link
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?	Link, Link, Link, Link, Link

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

When would you call a method with self?	Link

What are class methods?	Link
What is the purpose of a class variable?	Link
What is a constant variable?	Link
What is the default to_s method that comes with Ruby, and how do you override this? What are some important attributes of the to_s method?	Link, Link
From within a class, when an instance method uses self, what does it reference?	Link
What happens when you use self inside a class but outside of an instance method?	Link
Why do you need to use self when calling private setter methods?	Link
Why use self, and how does self change depending on the scope it is used in?	Link
Why is it generally a bad idea to override methods from the Object class, and which method is commonly overridden?	Link
What happens when you call the p method on an object? And the puts method?	Link
What are the scoping rules for class variables? What are the two main behaviors of class variables?	Link
What are the scoping rules for constant variables?	Link
How does sub-classing affect instance variables?	Link
Self refers to the ______ _______.	Link
How do you print the object so you can see the instance variables and their values along with the object?	Link
How do you override the to_s method? What does the to_s method have to do with puts?	Link
What is the default return value of to_s when invoked on an object? Where could you go to find out if you want to be sure?	Link #7
Why is it generally safer to use an explicit self. caller when you have a setter method unless you have a good reason to use the instance variable directly?	Link

Fake operators and equality

What is a fake operator?	Link
How does equivalence work in Ruby?	Link
How do you determine if two variables actually point to the same object?	Link
What is == in Ruby? How does == know what value to use for comparison?	Link
Is it possible to compare two objects of different classes?	Link
What do you get “for free” when you define a == method?	Link
What is the === method?	Link
What is the equal? method?	Link
What is the eql? method?	Link
What is interesting about the #object_id method and its relation to symbols and integers?	Link
When do shift methods make the most sense?	Link
Explain how the element reference getter and element assignment setter methods work, and their corresponding syntactical sugar.	Link