Ruby Project 1

By Lauren Cravey

1. ”Hello World”

```
puts "Hello World"
```

Data Structures

- A hash are key—value pairs where the keys are unique strings and the values are scalars
- A set is a list of values, unordered, and no duplicates
- A lists is a data structure that can hold objects such as strings, numerical values, and Boolean values

Array example

```
Array (1...10)
=> [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
puts Array
```

Dictionary Hash

```
dict = {"first_name" => "Lauren", "last_name" => "Cravey", "tiger_email_id" => "lec0051",
        "banner_id" => "903729326", "favorite_movies" => ["Mean Girls", "We're the Millers"]}
        puts dict
```

Class and Object in OOP

- Classes are basic outlines of what an object should be made of and what it is able to do
- Objects are created from classes (strings, integers, etc.)

```
class Car
    def initalize(make, model)
        @make = make
        @type = type
    end

    def vroom()
        puts "VROOM!"
    end
```

OOP in Ruby

- Now that we have created a car class, we can make a car object from that class. The object “lowRider” has access to the methods found in the Car class.
- lowRider.vroom will return “VROOM!” from the method vroom() found in the Car class

```
lowRider = Car.vroom()
lowRider.shout
```

Encapsulation

Encapsulation is often used to hide the internal representation of an object from the outside

goldenRetriever.getBreed() will return “Golden Retriever” because it has access to a getter method

```
class Dog
    def initialize(breed, name, age)
        @breed = breed
        @name = name
        @age = age
    end

    def getBreed()
        puts @breed
    end
end

goldenRetriever = Dog.new("Golden Retriever", "Charlie", "7")
goldenRetriever.getBreed()
```

Inheritance

Inheritance allows a child class to inherit all methods and variables from its parent class

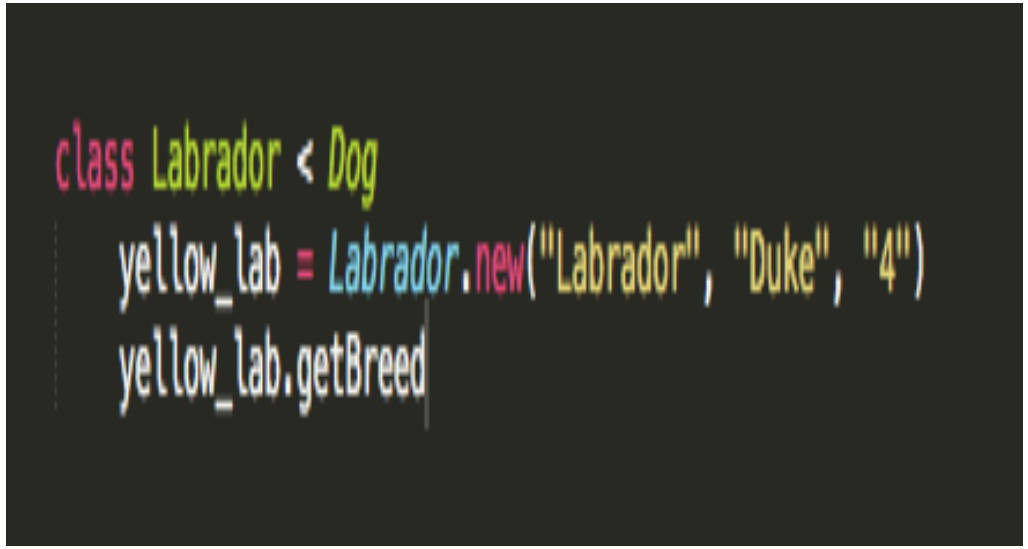
```ruby
class Dog
    def initialize(breed, name, age)
        @breed = breed
        @name = name
        @age = age
    end

    def getBreed()
        puts @breed
    end
end
```

Inheritance

Dog is the parent class of the Labrador class, so Labrador inherits all the methods from the Dog class

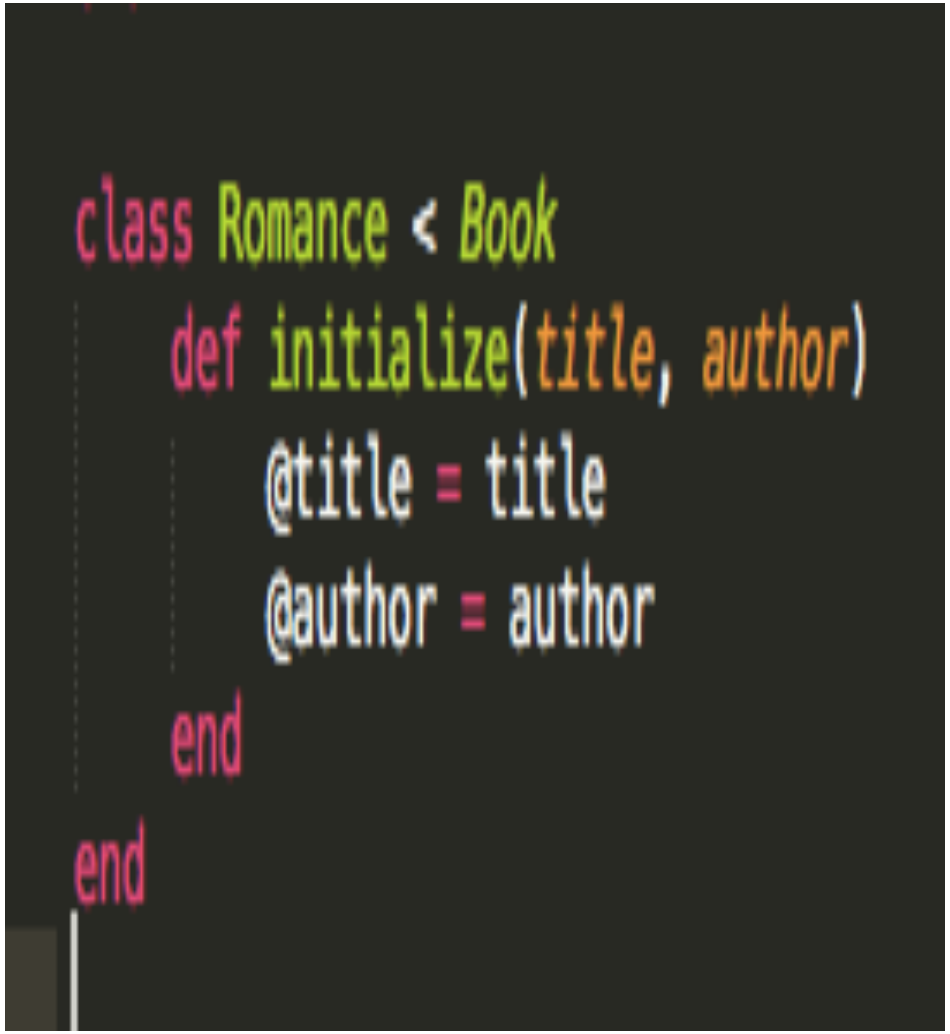
Yellow_lab.getBreed will return “Labrador”



Abstraction

Ruby does not support abstract classes and will return an error when attempting to initialize

When Romance is initialized, the program executes well



Animal Class

An Animal class is made with getter/setter methods for the two variables that were initialized

A horse object is created from the Animal class and then called to neigh

```ruby
class Animal
    def initialize(num_of_legs, type)
        @num_of_legs = num_of_legs
        @type = type
    end

    def getLegs()
        return @num_of_legs
    end

    def getType()
        return @type
    end

    def setLegs()
        @num_of_legs = legsIn
    end

    def setType()
        @num_of_legs = typeIn
    end

    def neigh()
        "Neigh"
    end
end

    spider = Animal.new(8, true)
    horse = Animal.new(4, true)
    horse.neigh
```

The difference between attr_reader, attr_writer, and attr_accessor

- Attr_reader can return the value of a variable without the use of a getter method.
- Attr_writer can set the value of a variable without a setter method.
- Attr_accessor can change the value of a method and return the value at the same time.

Thinking Assignment

- You have been given a list(array) of elements with a million entries in it. The list is not sorted, but this is a special list, which contains all entries in incremental order and at some point, it starts decreasing. How do you find that fluctuation point index?

Thinking Assignment(cont.)

- Since the list has a million entries in it, it will take a very long time to solve. We need an algorithm that takes time efficiency into account.
- The list is in incremental order, so the fluctuation point will be equal to the max index + 1.
- Since the list has a million entries, O(N) is not the most efficient because N = 1 million.
- We need to create a loop that is going to check the number, set it to the max, and keep track of the current max index and i index.
- When i<max, we need to exit the loop because i is no longer incrementing.
- Using the current index, we must backtrack backwards until we find an i that is greater than the previous max. Then we can exit and return the index of max + 1 to return the fluctuation point.