Question Set 1

1. What is an instance variable?

Class answer:

1. What's the difference between an instance variable and a 'regular' variable?

class answer: instance variable has a scope outside of where it is defined.

Question Set 2

1. What are "getter" and "setter" methods used for?

class answer: allow you to look at and write and modify a variable from outside the variable

1. What is the difference between a "getter" and a "setter" method?

class answer: getter is method to read that method from outside the variable – setter takes an equal sign and takes a parameter

c. What is the simplest mechanism to allow instance variables to be readable outside a class? Writeable? Both?

Class answer:

Question Set 3

1. How do you inherit a class? What does that mean?

Less than sign < to a parent class. --

1. What does it mean to overwrite a parent's method?

Class answer:

1. What does Ruby do when you call a method on an object but that method is not defined on the object?

Question Set 4

1. What does super do?

Class answer: uses the qualities given in the parent class – used so we don’t have to rewrite code.

1. Why use it?

Class answer: no rewriting code.

Lecture

example of inheritance

class Fruit

def initialize(name, color)

@name = name

@color = color

end

def taste

“I am really tasty”

end

end

class TropicalFruit < Fruit

def initialize(name, color, delicious

super(name, color)

@delicious = false

end

def taste

super + “I am really red too” ---\*\*\* the super here will say “I am really tasty” \*\*\*

end

end

t = TropicalFruit.new(“strawberry”, “red”, false)

* So here at TropicalFruit we are not redoing the work that the Fruit class has already done.

Topics:

* OOP and message passing
* Object Oriented Design
* Inheritance
* Exceptions

**Boxes –**

**Create a class called Box. We are making a blueprint to hold the properties and methods that will apply to multiple instances of boxes.**

**Inside Box:**

**Add three instance variables (height, width, and length)**

**Create an instance method that initializes a Box by taking in three numbers as parameters. An instance method is a method that applies to a specific instance of a Box.**

**Create a instance method that will calculate the volume and return**

**Program:**

**Initialize a box using three numbers as inputs for height width, and length**

**Calculate the volume of the box.**

**TO find the volume of any cube you need to know the length width and height. The formula is h\*w\*l.**

**Inside Box:**

**Add a method that takes in another box and returns how many times one box will fit inside the other. Be conscious about understanding whih box has a greater volume and how that will affect the result.**

**Note: Just use the volumes to calculate how many times a box can fit into another box, don’t bother with trying to come up with the logic of physically fitting boxes into each other.**

What is the difference b/w a table and a chair, -- they differ in length height etc they can both be used to sit, so there are only some differences so why create a program seperately for both, they both mostly have the same attributes.

class Box

attr\_accessor :height :width, :depth # can also do the following 2 instead of this.

# attr\_reader :height, :width, :depth

# attr\_writer :height :width, :depth

def initialize(height, width, depth)

@height = height

@width = width

@depth = depth ## the only ones that need to match are the height in the line 9 @height = height and the height in the initialize

end

def height=(h)

@height = h

end

def volume

height \* width \* depth

end

def can\_fit(box)

volume / box.volume

end

end

box1 = Box.new(10,15,20)

box2 = Box.new(2,2,2)

puts "Box1 can fit inside box 2 #{box1.can\_fit(box2)} times"

#Design the blog model

Blog

* Posts [] – array because theres no need to use anything more complex

Post

- title string

- description string

- date timestamp

- author user

- comments []

Comment

* author string
* date timestamp
* text string

BREAKOUT --- EXCEPTIONS

when there are ruby errors thrown at you. you can use puts $! f

begin

puts “enter a numerator”

num = ets.chomp.to\_i

puts ‘enter a denominator’

num = gets.chomp.to\_i

rescue

Test will throw a shit load of errors at you. reverse engineer the errors, go on ruby and find it

check compass email for breakout example codes.