Rails for Zombies NOTES lesson 1:

variable = {key: value}

variable[:key] will reference that variable's value (with printing, can do variable.key)

class.find(id) will reference the hash at that id value

**CRUD**

creating:

variable = class.new

variable.key = "put something here"

variable.save

*or*

variable=tablename.new(hash)

variable.save

*or*

Class.create(hash)

reading: look above. also:

you can .find with multiple parameters to return an array

tablename.first and .last

.all returns array of all

tablename.count returns number of shit inside

tablename.order(:key)

tablename.limit(value)

tablename.where(key: "whatever key you want") returns array of shit that meets that query

CAN CHAIN METHODS!

updating:

find the tweet (above if don't remember) and assign it to a variable

variable.key = "new shit"

variable.save

*or*

find the thing in the table and assign to variable

variable.attributes = {hash}

variable.save

*or*

after finding, variable.update(hash)

deleting:

variable = tablename.find(value)

variable.destroy

also variable.destroy\_all

LEVEL 2~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

MODELS

located in app/models/name of the model

for a table:

class TABLENAME < ActiveRecord::Base

end

can put stuff inside like:

validates\_presence\_of :key to ensure that a new input has that key value

validates\_numericality\_of

validates\_uniqueness\_of

validates\_confirmation\_of (useful for passwords and stuff)

validates\_acceptance\_of (useful for checkboxes on reading terms of service and shit)

validates\_length\_of :key, minimum: value (useful for making sure something is long enough)

validates\_inclusion\_of :key, in: minvalue..maxvalue

validates\_exclusion\_of: :key, in: minvalue...maxvalue, message: "whatever you wanna say if fail"

or fuck all that and just chain them together:

validates :key,

presence: true,

length: { minimum: value},

numericality : true,

etcetcetc (dont forget commas!)

if you want to relate 2 tables together, you can put stuff in when you initialize the classes:  
has\_many :othertablename (like zombies have many tweets)

belongs\_to :othertablename WITHOUT PLURAL and *lowercase*, BECAUSE BELONGS TO 1 ZOMBIE (like tweets belong to a certain zombie)

using these, you can do cool shit. Like:

find a zombie and assign it to a variable

variable = tweet.create(status: "stupid status",

zombie: var of zombie you picked earlier)

also do cool stuff like:

varofzombieFROMTABLE.tweets(tablename).count

varofzombieFROMTABLE.tweets

you can also do stuff like find a tweet form a zombie and assign it to a variable, then do t.zombie to see info about who wrote it. then u can specify key by taccing on .key to get some info from THAT table

if saving goes wrong, you can always do variable.errors.messages

LESSON 3~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~



in an html.erb file, anything you put in <% %> is interpreted as RUBY.

<%= %> basically says print whatever you put in these directly into html code at that spot.

**generally speaking, you can put ruby code for headers and shit into the show.html.erb file.**

-this allows you to use the print ruby syntax with yield to print whatever your show file gives you.

to create a link, put <% link\_to textoshow, \*model\_instance\* (tablenamesingular.key)%>