**TAKEAWAY\_ESTABLISHMENT – where you place an order, and where you view a menu**

require\_relative ‘menu’

require\_relative ‘checker’

require\_relative ‘texter’

on initialization, take a menu object, a checker object, a texting object, an order object

@menu = menu

@checker = checker

@texter = texter

@order = order

.view\_menu

@menu.list

.start\_order

@order.reset

.add\_to\_order(item:quantity)

@order.add(item:quantity)

.confirm\_and\_pay(total)

order = @order.show

if @checker.pass?(total,order,menu.list)

@texter.send\_confirmation

@order.reset

.place\_order(total:, order:) – order in the form of (total:123, order:{food1: 2, food2:4})

if @checker.pass?(total, order, menu.list)

@texter.send\_confirmation

(this is an alternative interface, bypassing the order object, which allows you to make an order item by item and confirm payment at the end…)

.check\_for \_orders

@texter.check\_orders(self)

**ORDER**

.reset

@order = {}

.add(<item hash>)

item\_hash.each do |item,quant|

@order[item]=quant

.show

@order

**MENU – stores the list hash**

on initialization, take a file which contains items and prices

def initialize(file)

@list = file

end

.list

@list

.list=(file)

@list = file

CHECKER

.pass?(total, order, menu)

check\_items(order,menu)

check\_price(total,order,menu)

true

.check\_items(order,menu)

check each of order.keys that menu.keys include the particular order key

unless the order key is included, raise error with useful message

.check\_price(total,order,menu)

take each of the items in the order hash and set key, value

for each order item, take menu[key][:price] \* value, and add that result to a running total

if the running total does not match total, raise error

TEXTER

require ‘ruby-twilio’

def initialize(twilio = Twilio)

@account\_sid =

@auth\_token =

@client = twilio::REST::Client.new @account\_sid, @auth\_token

end

.send\_confirmation

@client.account.sms.messages.create(

:from => XXXX

:to => XXXX

:body => ‘Message’)

.check\_for\_orders(establishment)

checks through list of text messages

if the text hasn’t been redacted, it is copied, then the origin is redacted

the copy is processed

(this has its problems – ideally wouldn’t redact the originals when they are processed, but I couldn’t get it to work putting them into log then checking each original against it, and ideally all texts would be processed in one go, not all up to the point one throws an error)