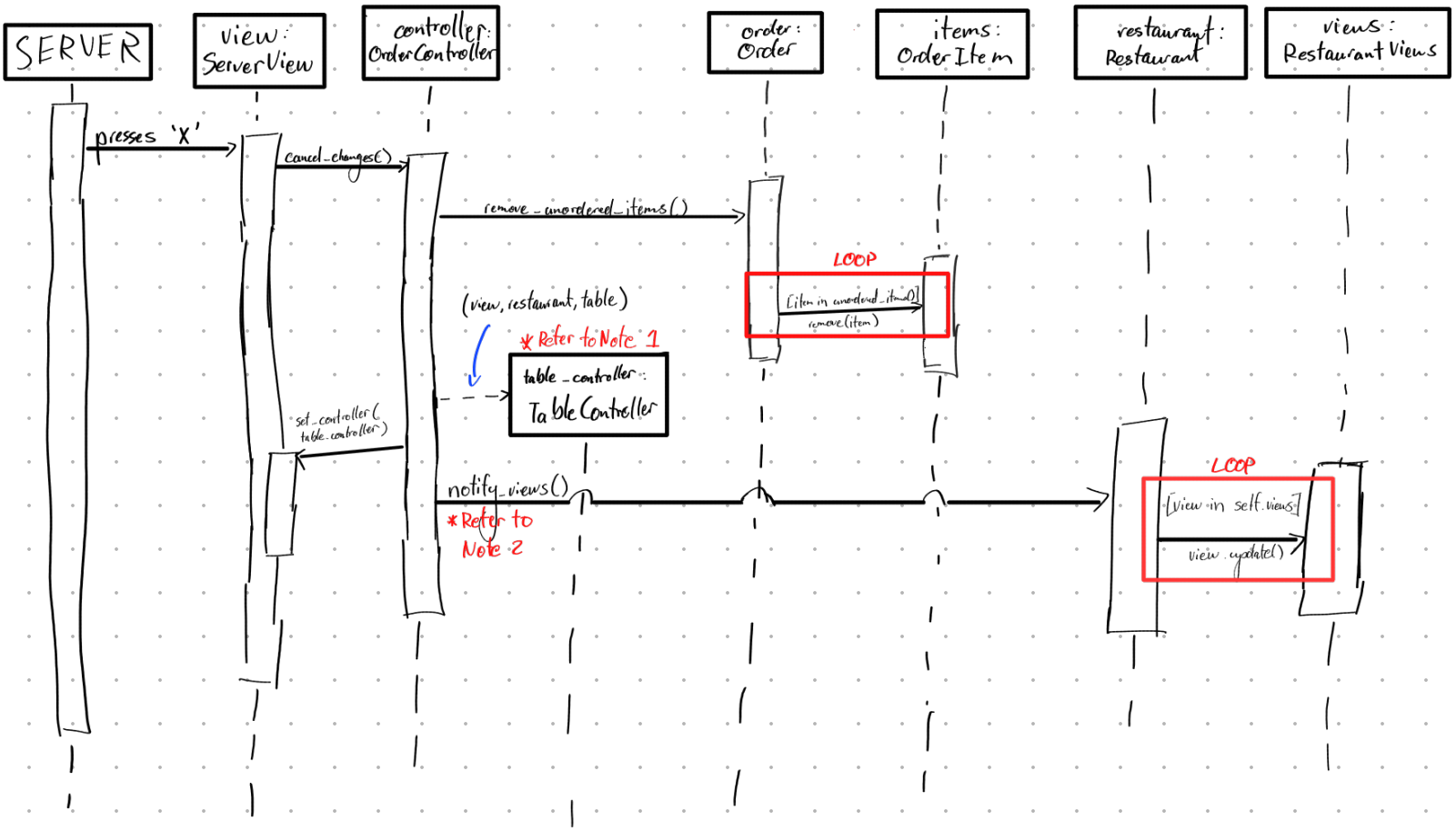
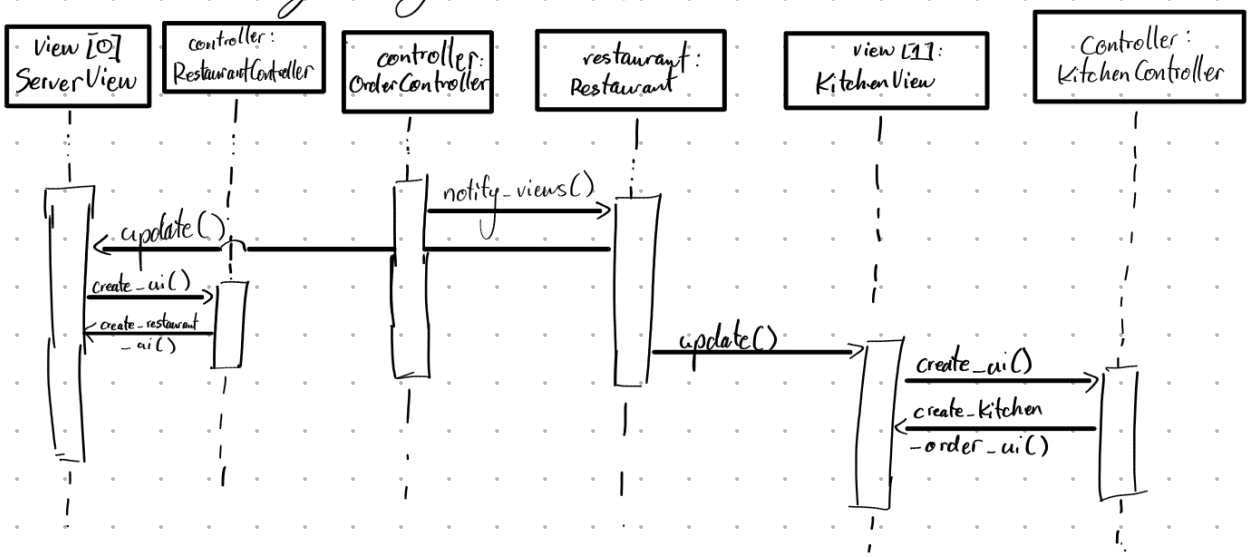
Note1: The table controller here was originally written in the code to be instantiated within the subsequent set_controller() statement. We decided to separate the object instantiation in the code so that drawing this sequence diagram would be more clear.

SEQUENCE DIAGRAM #2

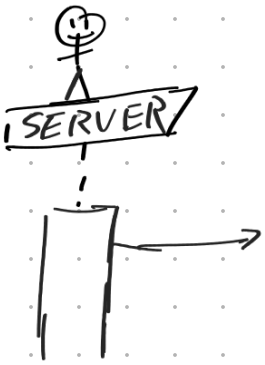
Scroll down for thought process to this diagram.



Note 2: the notify_views() portion (part C) can also be drawn in a more in depth way to show how the restaurant: Restaurant object directly interacts with ServerView and KitchenView, the two subclasses of RestaurantView. This basically shows what's actually happening



Sequence Diagram #2: when server presses "item cancel" button (the red 'X'), for an item whose order has been placed, but has yet to begin cooking.



Flow tree

In Restaurant View

1. Presses 'X'

↳ In Order Controller

1. Call cancel_changes

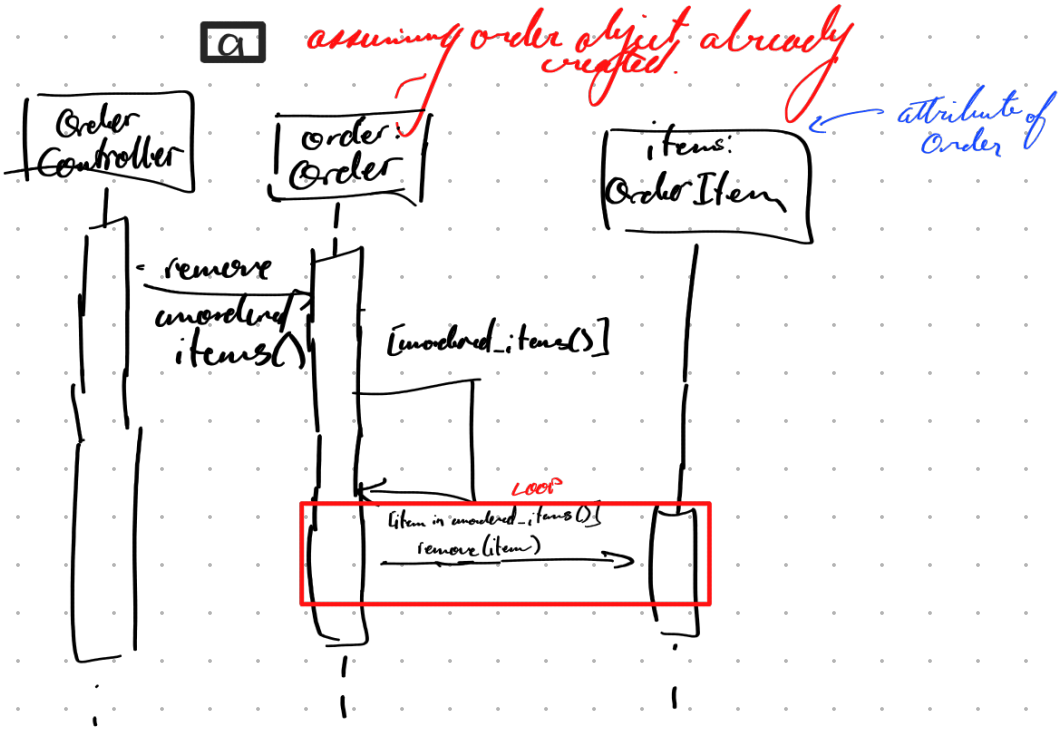
a. this_order ← remove_unordered_items

b. this_view ← set_controller (table_controller)

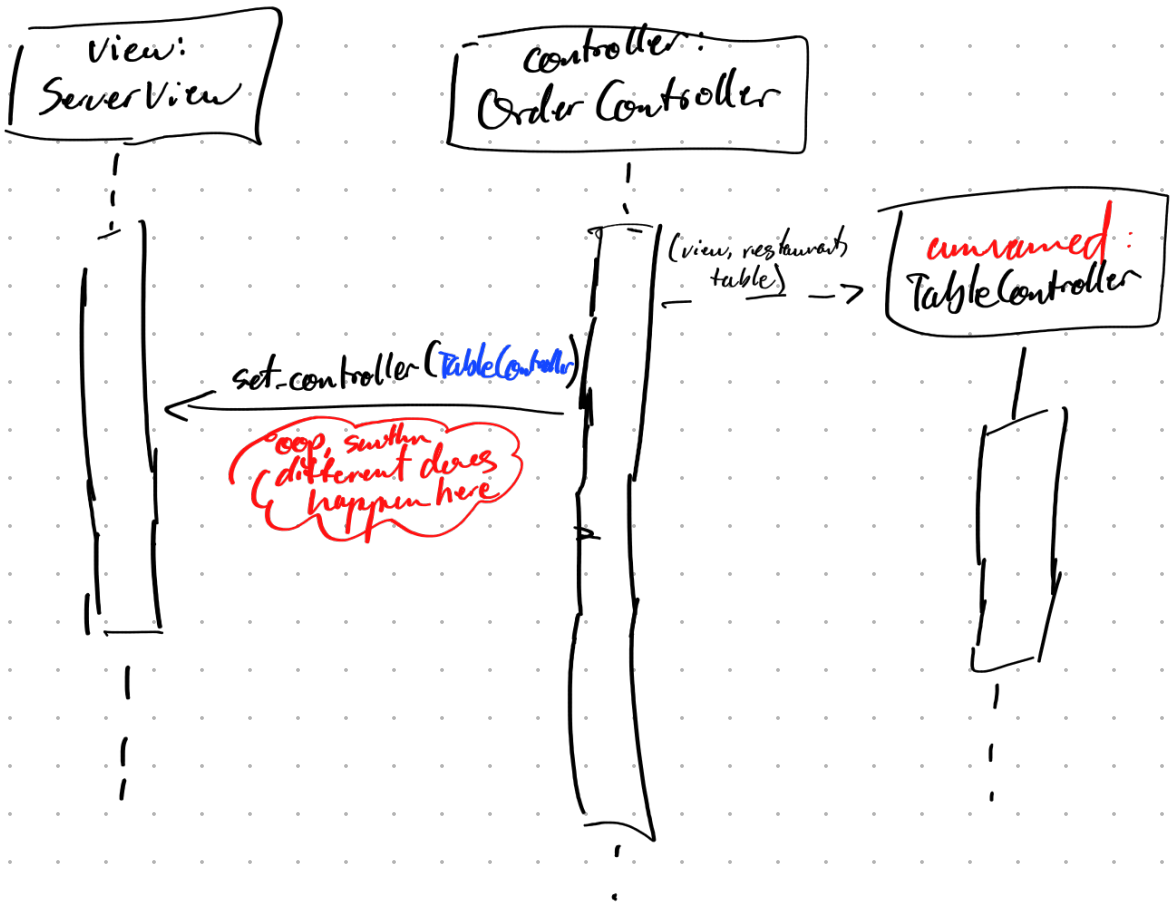
c. this_restaurant ← notify_menus

(this_view, this_rest, this_table)

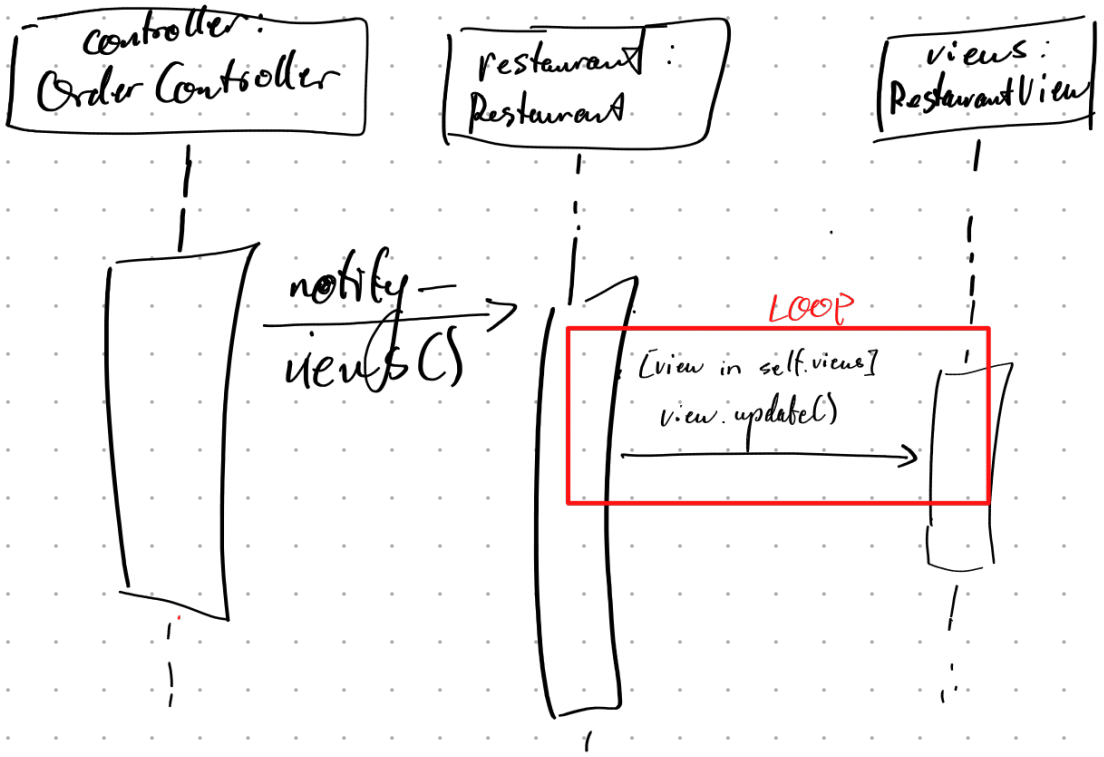
a



b


view:
ServerView
controller:
Order Controller
(view, restaurant, table)
unnamed:
TableController
set_controller(TableController)
oop, another different does happen here

c


controller:
Order Controller
restaurant:
RestaurantA
views:
RestaurantView
notify_views()
LOOP
[view in self.views]
view.update()