Additional Specs for Phase 2

In phase 2 you will have to create models for `Customer`, `Address`, and `Order` and write unit tests for these models. To make sure you are able to build these models and tests, here are some specs for each of these models:

Customers must:

1. have all proper relationships specified
2. have values which are the proper data type and within proper ranges
3. have a first name, last name
4. have phone values that are saved in the system as a string of digits (no other characters allowed in the database, but user may input values with dashes, dots or other common formats – e.g., 999-999-9999; 999.999.9999; (999) 999-9999 are all acceptable)
5. have the following scopes:
   a) 'active' -- returns only active customers
   b) 'inactive' -- returns all inactive customers
   c) 'alphabetical' -- orders results alphabetically by last name, first name
6. have the following methods:
   a) 'name' -- which returns the customer name as a string "last_name, first_name" in that order
   b) 'proper_name' -- which returns the customer name as a string "first_name last_name" in that order with a space between them

Addresses must:

1. have all proper relationships specified
2. have a recipient, a primary street address and a proper 5-digit zip code
3. have values which are the proper data type and within proper ranges
4. restrict customer_ids to customers which exist and are active in the system
5. have a method called 'already_exists?' which should determine whether or not an address is already in the system for this customer. (For now, a duplicate address is defined as same recipient and zip code for a particular customer)
5. should not allow duplicate addresses to be added in the system
   (warning: a check for duplicates should only run when a new record is created. If the validation were to be run when editing a record, it would make it difficult to edit the record because that particular address already exists.)

6. have the following scopes:
    a) 'active' -- returns only active addresses
    b) 'inactive' -- returns all inactive addresses
    c) 'by_recipient' -- orders results alphabetically by recipient name
    d) 'by_customer' -- orders results alphabetically by customer's last name, first name
    e) 'shipping' -- returns all addresses which are just shipping addresses
    f) 'billing' -- returns all addresses which are also billing addresses

Orders must:

1. have all proper relationships specified
2. have values which are the proper data type and within proper ranges
3. restrict customer_ids to customers which exist and are active in the system
4. restrict address_ids to addresses which exist and are active in the system
5. have the following scopes:
    a) 'chronological' -- orders results chronologically with most recent orders listed at the top
    b) 'paid' -- returns all orders that have been paid (i.e., have a payment receipt recorded)
    c) 'for_customer' –- returns all the orders for a particular customer (parameter: customer_id)
6. have an instance method called 'pay' that will create a base64 encoded payment string that will serve as the payment receipt. The encoded string should be of the format "order: <the order id>; amount_paid: <grand total>; received: <the order date>". To prevent double-payments, the method should only generate payment receipts for orders that have not already been paid for and return false.

NOTE: This phase we will not validate that any active field is actually a boolean.