More Pizza!!

Learning objectives

- Understand how messages can be handled more abstractly
- Think about using one interface to a more complicated OO system underneath (the Facade pattern)

Instructions

Continuing with the pizza place, let's make a system to handle pizza delivery.

This system will need to record how many pizzas are being delivered.

Create two classes: DeliveryService and DeliveryCar.

The DeliveryService class will be the one that will handle messages from the manager of the restaurant. The DeliveryCars will be the ones that will actually take the pizza out to the customer. Both classes will need one method:

deliverPizza:(Pizza *)pizza;

The DeliveryService class will also need a method that will return an array of strings that consists of a description of every pizza it has delivered.

The manager (in this setup, a manager is required) will send a deliverPizza message to an instance of the DeliveryService class when the manager receives a didMakePizza message. This means the manager will need to have

a reference to a DeliveryService object, so change the manager class to allow for that.

The DeliveryService object will make an internal record of the pizza description, and then send a deliverPizza message to a DeliveryCar. So, notice, Kitchen talks to Manager (through the delegate property), Manager talks to DeliveryService, and DeleveryService talks to DeliveryCar!

This is a simplified model (you might have noticed that), so just have a single car object that is referenced internally by the DeliveryService object. The implementation of the car's deliverPizzamethod just prints out "Pizza Delivered".

In main.m initialize a Manager and DeliveryService right after you initialize the Kitchen. Finally, in the while loop, add a command that will display the information stored by the delivery service.