```
============================================================
== Andrew St. Pierre
== ECE39500 Exam 1
============================================================
1.
  1a. To run simulation: ruby ducksim.rb
                          OR ./ducksim.rb
  1b. To run simulation: ruby coffeeshop.rb
                          OR ./coffeeshop.rb

2. To change one of the behaviors of a Duck, I would create
   another class that implemented the desired modification
   to the existing behavior and then wrap the Duck class by
   this modified decorator class.

   Requires more work than creating a varient strategy and
   setting that new NoQuack strategy behavior as the duck's
   QuackBehavior.


3. To add an additional ingredient to coffee, an entirely
   new coffee object will need to be created with Mocha
   included in the recipe ingredients strategy.

   Requires much more work than simply decorating the
   existing coffee object with a Mocha decorator.

4.
  > Ducks have behavioral characteristics that alter
    slightly among different kinds (ex. Mallard + DogToy)

  > Each behavior varient (ex. Paddle + Float) is similar
    and static (not dynamically changing at run-time).
    This makes a strategy pattern better-fitted for
    modeling ducks because it removes some of the
    complexity decorators bring into the client code.

5.
  > Coffee is a very personal-preference-driven object
    For a coffeeshop with X ingredients, that creates
    2^X possible coffee varients.

  > The strategy design pattern requires a more static
    interface when making the coffee object. It does
    not easily allow for an existing coffee varient
    (say, a Latte) to add an additional ingredient,
    (say, sugar).

  > A decorator design pattern is better suited for
    modeling a coffee object because additional
```

customizations to an existing coffee varient (say, Latte w/ extra Sugar).
To provide this customized coffee, just decorate the coffee object with a Sugar decorator.