# Homework Assignment

This assignment helps us assess your programming abilities. Your mission, should you choose to accept it, is to complete the tasks below and return your completed files to us as soon as possible. Good luck!

## Guidelines

* Don’t rush, but don’t spend forever on it either. A few hours max should be plenty.
* Create an archive of your completed project and return directly to the sender.
* Do **not** share the assignment, original or complete, with anyone else.
* Your implementation should **not** require a physical database connection. When needed, fake db operations are expected.
* Keep in mind: this is our **sole visibility** into your ability to understand requirements and implement solutions. Although you’ll be working with simple objects and easy problems, always use best practices and avoid taking shortcuts. We take this **seriously**.

## Coding Tasks

### Part 1

You’ll be working with cats, dogs, and a fake database connection. To begin, translate these three pseudocode files into one of the following languages (Ruby, Java, C#, python, JavaScript ES6) . The examples here use Ruby:

Cat.pseudo

class Cat {  
  
 attribute name  
 attribute age  
 attribute favoriteFood

method constructor {  
 this.name = nil  
 this.age = nil  
 this.favoriteFood = nil  
 }  
  
 method getName {  
 return this.name  
 }  
  
 method getAge {  
 return this.age  
 }  
   
 method getFavoriteFood {  
 return this.favoriteFood  
 }  
   
 method setName (newName) {  
 this.name = newName  
 }  
  
 method setAge (newAge) {  
 this.age = newAge  
 }  
   
 method setFavoriteFood (newFavoriteFood) {  
 this.favoriteFood = newFavoriteFood  
 }  
  
}

Data.pseudo

class Data {  
  
 method constructor (database) {  
 print "Connecting to database"

}  
  
 method beginTran {  
 print "Beginning a transaction"  
 }  
  
 method commit {  
 print "Committing transaction"  
 }  
   
 method rollback {  
 print "Rolling back transaction"  
 }  
  
 method insert (table, object) {  
 print "Inserting " + object.getName() + " into table " + table  
 }  
}

main.pseudo

cat = new Cat()  
print "Name is currently " + cat.getName()  
  
cat.setName("Garfield")  
print "Name has been changed to " + cat.getName()  
  
data = new Data("database")  
  
data.insert("Cat", cat)

Once translated, run the main script and verify your output against ours:

Name is currently

Name has been changed to Garfield

Inserting Garfield into table Cat

### Part 2

As you proceed through these tasks, modify the main.rb script as you see fit, just to informally demonstrate the additional functionality. (Formal tests come later.)

1. Modify the constructor to set the cat’s initial age to a random number between 5 and 10.
2. Modify the constructor to support an optional initial name for the new cat.
3. Add a method called **speak**() to make the cat say “meow” (use *print* or *echo* to speak).
4. Modify the new **speak**() method to accept an optional argument.  
   If not supplied, the cat still says “meow”.  
   If supplied, the cat prints the value of the argument.
5. Modify the **setName**() method to keep a list of all the names the cat has ever had.
6. Create a **getNames**() method to return a list of all the names the cat has ever had.
7. Modify the class so that the cat’s age increases by 1 every five times it speaks.
8. Create a new method **getAverageNameLength**() that returns the average length of all the names the cat has ever had.
9. Make a Dog object, making necessary dog-like changes.

### Part 3

Now it’s time to actually do something with your objects. Create a new file, petShop.rb, which implements the three functions below.

1. Create a function called **saveTest**() to:
   1. Create a cat with a name and insert it into the database.
   2. Create a dog with a name and insert it into the database.
2. Create a function called **savePetShop**() to:
   1. Create three nameless cats.
   2. Create three nameless dogs.
   3. Insert all six pets into the database.
   4. **Important**: guarantee that all six pets (or zero pets) are persisted.
3. Create a function called **logStats**() to:
   1. Print interesting information about your script’s execution results to STDOUT.
4. In the same file, call all three functions to ensure they perform as intended.

## SQL Tasks

Create a new file, homework.sql, containing:

1. SQL statements to create table(s) to store our animals.  
   Hint: Don’t worry about the actual DBMS or SQL dialect. Generic SQL is fine.  
   **Important**: Ensure your database schema can be used to fully persist an instantiated object.
2. Sample insert statements.

## Testing Tasks

In a new file (or files), create unit tests to:

1. Assert that a cat’s initial age is a random number between 5 and 10.
2. Assert the **speak**() method is properly functioning.
3. Optionally include other test cases you deem appropriate.

## Documentation

As a final step, create a brief README file for the project. Feel free to use this file to:

1. IMPORTANT: clear instructions on how to run this. We do not want to spend more than 5 min figuring out how to test the code.
2. Describe the project, file structure, what it does, etc.
3. Discuss how or (more importantly) *why* you did things the way you did.
4. If you disagree with any requirements, explain what you would do differently. In short, anything you think would help a future developer understand your project.