1. Take a look at Smalltalk's syntax:
   - http://stephane.ducasse.free.fr/Teaching/0809Turino/st-cheatsheet.pdf
   - http://files.pharo.org/media/flyer-cheat-sheet.pdf
   - Smalltalk has three kinds of messages. What are they and what do they mean?
   - What is the result of 2+4*3 according to Smalltalk's rules?
   - What is a block?
   - Explain self and the character ^ in Smalltalk.
2. Download and install Pharo from http://www.pharo.org. What is Pharo? What's Pharo got to do with Smalltalk?
3. Complete assignments 3-10 using Pharo. Let's take a look at the UI first.
   - Start Pharo and press your right mouse button (later marked as (▯▯▮)) over the blank canvas and choose System Browser. A window with five subwindows opens and a list of packages is displayed on the left.
   - Choose any package from the list and (▯▯▮) → Add package.
   - Name the package, e.g. NK-Demo1, in which NK are your initials.
   - The package you created should now be listed in the left subwindow and in the bottom subwindow you should see four lines of code.
   - Let's create a new class. Replace the lines of code with the example below. Note that there are no quotation marks but a total of six apostrophes.

```
Object subclass: #MyClass
  instanceVariableNames: ''
  classVariableNames: ''
  category: 'Demo1'
```

   - Paint the lines of code and (▯▯▮) → Accept. This must be done every time you change something. Write your initials if prompted.
   - Your new class should now be listed on the second subwindow from the left.
   - What is the convention on naming classes?
   - Shortly explain the four lines of code above and what they do.
4. Now let's create a method. In Smalltalk, methods are sometimes called messages.
   - Select your class from the list.
   - You should see a line "no messages" in the third subwindow from the left. Select the line.
   - You should see some lines of code in the bottom subwindow. Replace them with the following:

```
printForPetesSake
  Transcript show: 'Seems to be working.'.
```

   - Paint the lines and (▯▯▮) → Accept.
   - You should see the method listed in the rightmost subwindow.
   - What do you think the method does when called?
   - What is the convention on naming methods?

- You have most likeley written a function or procedure or subprogram or subroutine before which looks a lot like the method above. What is the difference between a method and a subprogram (etc.)?

5. Now we a class and a method. Let's create an object.

- On the canvas, (▯▯▮) → Workspace. This opens the World-menu.
- One the canvas (▯▯▮) → Tools → Transcript. Text will be printed to this terminal.
- Write this to the Workspace:

```
firstthing := MyClass new.
firstthing printForPetesSake.
```

- Paint the text and (▯▯▮) → Do. This must be done every time you want to execute something.
- What does the first line do?
- What does the second line do?
- What is the difference between a class and an object?

6. Let's create another class:

```
Object subclass: #Counter
  instanceVariableNames: 'value'
  classVariableNames: ''
  category: 'Demo1'
```

- and a method for that class:

```
add
  value := value+1.
```

- and another method:

```
initialize
  value := 0.
```

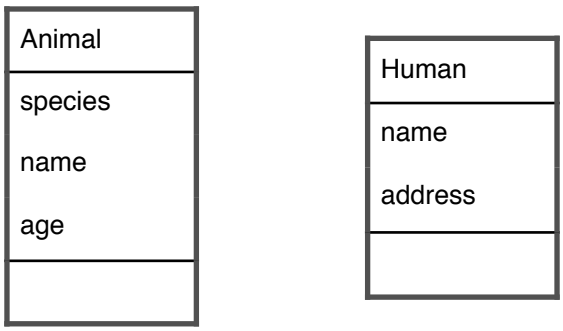
- It's crucial to name the last method according to the example above. Every object has a method called initialize which is called automatically when the object is created. We have now overwritten the original initialize method with our own. We will discuss this further later on the course.
- Create an object from the Counter class.
- When you have created the object (▯▯▮) → Inspect it on the object's name in the Workspace-window. What do you see?
- Call the method and Inspect the object again. What do you see?
- Create two more objects from the Counter class and call their methods multiple times. Inspect the objects. What do you see?

7. Take a look at the UML representation of a class below and implement it.

8. Take a look at the class below written in Smalltalk. Draw an UML representation of the class using the UML tool of your choise.

```
Object subclass: #Product
  instanceVariableNames: 'productid name price'
  classVariableNames: ''
  category: 'Demo1'
```

9. Take a look at the UML class diagram below and implement it.



10. Take a look at the object diagram below and bring your system to the exact state the diagram represents.