The Smalltalk Meta Object Protocol (part I)

1¿ Which is the class all other classes in Smalltalk inherit from (directly or indirectly) ?

2¿ Create an instance of that class.
   (a) What is the class of that instance ?

   (b) What message can you send the instance to get the answer to (a) ?

   (c) How can you see the same in an inspector window without sending a message ?

3¿ What is the class of the class obtained in question 2.a ?

4¿ What is the class of the class obtained in question 3 ?

5¿ (a) What is the superclass of the class obtained in question 3 ?

   (b) What message can you send the class to get the answer to (a) ?

6¿ Query the classes obtained in questions 2.a, 3 and 4 for their superclass. Then get the superclasses of those classes and so on until you can't go any further. Write down the complete superclass chain for each of the three classes from 2.a, 3 and 4:

   =>    =>    =>    =>
   =>    =>    =>    =>
   =>    =>    =>    =>
7¿ (a) How many direct subclasses does the root class in Smalltalk have?

(b) What message can you send the class to get the answer to (a)?

(c) What message can you send the class to get all subclasses (direct and indirect ones)?

(d) How many direct and indirect subclasses does the root class have?

(e) How many classes are there in your Smalltalk image?

9¿ (a) How many instances does the class Array have? (Use the allInstances message)

(b) How many instances does the class of the class Array have?

(c) How many instances does the class obtained in question 3 have?

(d) What happens when you try to create a new instance of the class in question 3?

(e) How many instances does the class obtained in question 4 have?

(f) Create a new class. What happens to the answers for question 7.e and 9.e?

(g) What is the relationship between the number from 9.e and the one from question 7.e? How does this relationship arise?
10. Make a drawing depicting the relationships between all the classes from question 6 as well as the classes Collection and Bag. Remember there are two different relationships between them: *is-instance-of* and *is-subclass-of* (use different colors).

11. In questions 2 to 4 you had to get the class of a class, then the class of that class etc. How long can you go on doing this? What happens? Why is this? (Also: add the relationship you discover here to your diagram for question 10 if it's not on there already)
12¿ (a) Send the message *selectors* to the class *OrderedCollection*. Explain the result:

(b) What do you get when you send the same message to the *class* of *OrderedCollection*? How is this related to what you view in the class browser in the user interface?

13¿ You can send the message *new* to the root class even though it has no method for it.
(a) First verify that the class really doesn't have the method. Where do you need to look?

(b) Which *new* method is executed when you send the message, where is it implemented? Why that one?

(c) Repeat 9.d again, which *new* method is being executed here? Why that one? What does the method do?

14¿ Send the messages *instVarAt: 4* and *instVarAt: 1* to *Collection*, what are the results? Why do you get these results? Can you send the *instVarAt:* message to all objects (try today's Date object) and why (not)?

15¿ What happens when you send the message *perform: #selectors* to a class? What other methods with *perform:* as the first keyword exist and what are they used for?
16. Send the message `methodDictionary` to Date, what does the resulting object represent? The object contains other objects, what are the classes of those objects? And what do those objects contain?

17. Implement a class with the following two methods. Create an instance of the class and send it the `methodOne` message.

```smalltalk
methodOne
  self methodTwo
methodTwo
  thisContext inspect.
  self halt.
```

What does `thisContext` contain? There's at least one tool in Smalltalk that is implemented using this reflective mechanism, which one you think and why?

18. To the class you created in the previous question you should now add a method with selector `doesNotUnderstand:` with an empty body. Send the message `doSomethingCompletelyRandom` to an instance of the class. What would you normally expect to happen? What happens now? Why? Where is the "normal" behavior implemented?

19. Send a message `compile:` with an appropriate argument to the class you created in the previous questions. What would an appropriate argument be? What happens?
Reflections

The following questions are provided to help you think about what you've learned in this exercise, but they are not formally part of it and no time is reserved in the practical sessions for solving them. Though do feel free to bother the assistant to verify your answers if he's not busy doing anything else (but don't just ask what the answer to the question is).

- What do you think is a "meta object protocol"?
- What parts of Smalltalk form the meta object protocol?
- What is the relationship between the meta object protocol and the "tight integration of language and development environment in Smalltalk"?
- What would a meta object protocol for C++ look like? Would it be enough to just have a class Program that you can query for the source text of the C++ program as a string? Why (not)?
- Why is it called "meta object protocol"? (Why "meta"? Why "object"? Why "protocol"?)