CSE 341 – Meta Mini Madness!

1. Consider the `Stack` and `FilteredStack` classes from homeworks 6 and 7. Suppose you have an instance `s` of `Stack`, and an instance `f` of `FilteredStack`.

   Draw a diagram showing all superclass and instance relations among the following objects:

   ```
   s
   f
   Stack
   Stack class
   FilteredStack
   FilteredStack class
   Object
   Object class
   ProtoObject
   ProtoObject class
   ```

2. Every object in Smalltalk is an instance of some class. For extra bonus points, augment the diagram for Question 1 by adding additional classes and instance-of links so that *every* object in the diagram has an instance-of link from it to its class. (So `s` will have an instance-of link to `Stack`. `Stack` will have an instance-of link to `Stack class`. `Stack class` will have an instance-of link to ...

   Don't worry about putting in a superclass for every class — just put in the instance-of links. Caution: if you find yourself drawing an infinite number of classes and links and become stuck in a mental loop, stop thinking about the problem immediately and hold up a crudely-lettered sign saying **HELP!** Mental health first aid personnel will be standing by.

3. Recall that `FilteredStack` overrides the inherited `push:` method from `Stack`. `Stack` also defines a `pushAll:` method, which is inherited by `FilteredStack`. Suppose that we evaluate

   ```
   f pushAll: #('one fish' 'two fish' 'red fish' 'blue fish').
   ```

   Describe how this method is executed, in particular, in which classes the `pushAll:` and `push:` methods are found and how.

4. Suppose that we define class methods for `Stack` and `FilteredStack` to create and initialize instances, as follows. (This is slightly different than what you probably had in your homework — it's to illustrate a point about inheritance among metaclasses.)

```
Stack class
   size: n
      ^self new setsize: n

Filtered class
   size: n filter: f
         | stack |
      stack := super size: n.
      stack filter: f
      ^stack

   size: n
      "default behavior is to not filter out anything"
      ^self size: n filter: [:x | true]
```

These class methods use private instance methods `setsize:` in `Stack` and `filter:` in `FilteredStack`.

Suppose now that we evaluate:

```
f1 := FilteredStack size: 10.
f2 := FilteredStack size: 20 filter: [:x | x>0].
```

Describe how these statements are executed, in particular, in which classes the methods are found.