Prof. Dr. Oscar Nierstrasz
Pascal Gadient, Pooja Rani

Solution

Assignment 03 — 30.09.2020 – v1.0
Smalltalk: Understanding Classes and Metaclasses

Please submit this assignment by email to pascal.gadient@inf.unibe.ch before 07. October 2020, 10:15am.

Exercise 1 – Metamodels (2.5 pts)

Answer the following questions regarding metamodels:

i) What is a metamodel? **Answer:**

It is a model of a model. In other words, a metamodel is a prescriptive view on an existing model. A metamodel determines the syntax and semantics of models that conform to it. Metamodels can leverage various forms, e.g., *grammar syntax, flow charts, and UML diagrams.*

ii) How are metamodels used in Pharo?
You must use the classes `Object`, `Class`, and `Metaclass` in your answer. **Answer:**

Every object is an instance of a class. Every class inherits from `Object`*. Every class is an instance of its (unique) metaclass, which inherits from* `Class`*. Every metaclass is an instance of* `Metaclass`*, which is itself a class.*

iii) What are responsibilities of a metaclass in Pharo? **Answer:**

Instance creation, creating initialized instances of the metaclass's sole instance, initialization of class variables, metaclass instance protocol, method compilation (different semantics can be introduced), class information (inheritance link, instance variable, ...).

iv) Where is `ProtoObject` located in Pharo's class hierarchy? **Answer:**

In Pharo, `ProtoObject` *is the root class for all other classes including* `Object`*.* `ProtoObject` *is the superclass of* `Object`*.*

v) What is the purpose of the class `ProtoObject`? **Answer:**

While `Object` *provides (most of) the common message handlers,* e.g., `printOn`*, the class* `ProtoObject` *does not carry all that "baggage" and only contains the core behavior needed to make the system work. The idea of* `ProtoObject` *is to have a lean class that separates the concerns.*

Exercise 2 – Sub and super classes (3 pts)

Answer the questions below. Please provide your code *and* your results.

i) How many superclasses does `Collection` have? **Answer:**

```
Collection allSuperclasses size.
```

2. Consequently, the class has two super classes.

ii) How many direct subclasses does `Collection` have? **Answer:**

```
Collection subclasses size.
```

32. Therefore, the class has 32 direct subclasses.

iii) How many indirect subclasses does `Collection` have? **Answer:**

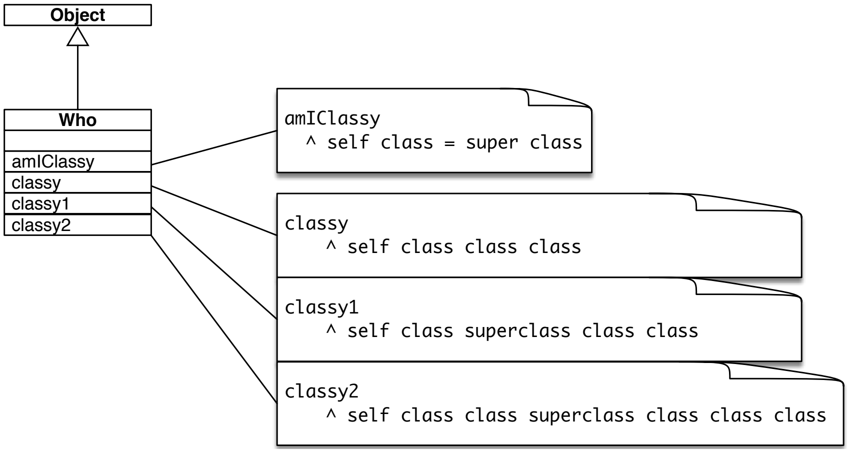
```
Collection allSubclasses size - Collection subclasses size.
```

129. The class `Collection` has a total of 161 subclasses, whereas (161 - 32 =) 129 are indirect subclasses.

*NB: Direct subclasses are classes that extend a base class directly (*e.g.*, relation parents to children), whereas indirect subclasses extend the direct and (recursively) indirect subclasses (*e.g.*, relation grandparents to grandchildren).*

NB: Please use a fresh copy of GT.

Exercise 3 – Class identity (3 pts)



```
amIClassy
  ^ self class = super class
```

```
classy
    ^ self class class class
```

```
classy1
    ^ self class superclass class class
```

```
classy2
    ^ self class class superclass class class class
```

Consider the implementation shown in the illustration.

What are the results (either `true` or `false`) of the following statements?
Explain for each statement why GT replied the corresponding result.

a) `Who new amIClassy.` **Answer:**

True. `super` *is executed in the context of the class of the method implementation.* `super class` *starts the lookup in the superclass of the implementing method, namely* `Object`*, while* `self class` *starts in the class of the instance, namely* `Who`*. But since* `Who` *does not implement class, both expressions find the same method.*

b) `Who new classy = Who new classy1.` **Answer:**

True. Both call chains reach the root of the class hierarchy tree (Metaclass class) which is identical for both of them.
`Who new classy`: `Who class class class`, returns a `Metaclass class`.
`Who new classy1`: `Who class superclass` returns an `Object class`, `Object class class class` finally returns a `Metaclass class`.

c) `Who new classy1 = Who new classy2.` **Answer:**

True. Both elements represent the same class.
`Who new classy1`: Returns a `Metaclass class`.
`Who new classy2`: `self class class` returns a `Metaclass (Who class)` object, `Metaclass (Who class) superclass` returns a `Metaclass (Object class)` object, and finally, `Metaclass (Object class) class class class` returns a `Metaclass class`. The last two message sends exploit the circular dependency between `Metaclass` and the `Metaclass class`.

Exercise 4 – Object instantiation (1.5 pts)

A new instance of a class is created by sending the message `new` to the respective class.
Please answer the following related questions:

i) Where is `new` defined? **Answer:**

`new` is first defined in the class `Behavior`. However, it can be redefined in its subclasses, including any metaclass of the classes we define.

ii) Explain Pharo's message implementation resolution strategy for the `new` message. **Answer:**

When the message `new` is sent to a class it is being resolved throughout its metaclass chain. This search ultimately ends in its super classes `Class`, `ClassDescription` and `Behavior`.

iii) List the concrete code in GT finally executed by the message `new`. **Answer:**

In the official GT image, when `new` is executed the initiated message sends will include a final `self basicNew initialize` in the class `Behaviour`.