|  |
| --- |
| Circle Language Spec: Classes |

## Target Classes in a Diagram

### Concept

A target class is found by following the redirections, that lead to a symbol’s class.

Do not follow more than one class redirection, because if a class points out a class again, then the second class is *another* class object, that the first class is just *based* on. If the class is an object reference itself, you have to follow all object redirections to find the target class object. Then you have found the target class. That’s where redirection following ends. If the class object has a class itself, you might be tempted to follow the class object’s class redirections as well, to find the final target class, but you should not do that. The first class redirection indicates the class. If that class object has a class itself, then the class object is only based on another class, but it *is* a class on its own. An object redirection is just a much tighter bond like that, than a class redirection.

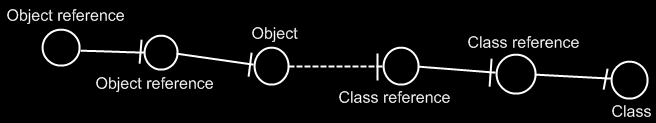
### Diagram Notation

The concept of target classes is explained in the article *Target Classes*. This article only explains their expression in a diagram.

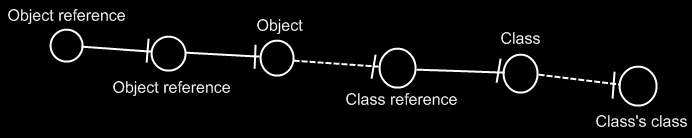
The target class is found by following the redirections, that lead to a symbol’s class.

When you want to find the class of an object, and the object is actually an object reference, you first need to follow all object reference redirections, to find the target object. When you found the target object, you can find the target class, by following one class redirection.

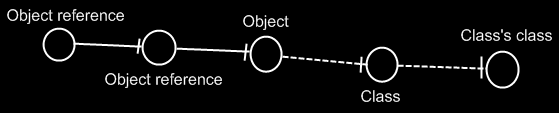
So to find the target class, you first follow *all* the object redirections, then *one* class redirections, then *all* the object redirections and there it ends.



If the class has a class as well, this does not redirect the original object’s class, because the second class is *another* class object, that the first class is just *based* on. An object redirection is just a much tighter bond, than a class redirection.



The target class of the first object reference is the symbol Class, not the symbol Class’s class. The same counts for the diagram below.



If you wonder what could be that different between Class and Class’s class: they could differ in default values. The main point is: finding the target class is about finding the class object.

## Ideas

### Out of the original Symbol documentation

#### Tracing Object Aspects

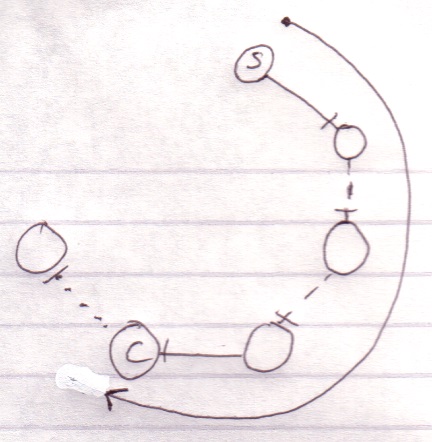
Formerly I’ve said that when you encountered a symbol that doesn’t have a type line, then it is the target type. But in *Object Basics* I said that when a symbol doesn’t have a type line, the object line functions as the type line. Therefore, if a symbol has no type line, the type can still be redirected by an object line.

Finding the aspects of a symbol, such as target object or target type, is called a *trace*.

##### Type Trace

You’ll use type and object lines to trace the type. Follow the type line if it exists, else follow the object line. When you run into a symbol with no type or object line, then that’s the type.

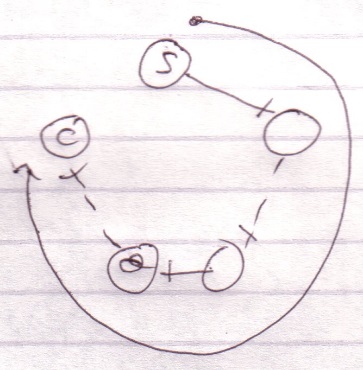
When there is no type line, the object determines the type.



Interface lines are not followed. Note that the target type doesn’t have to be pointed to by a type line.

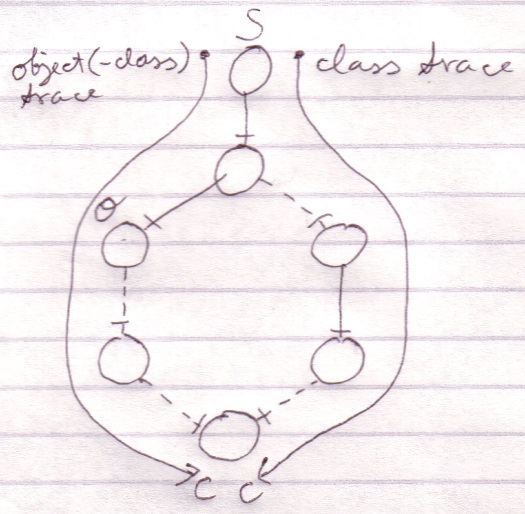
##### Object-Type Trace

The last symbol in the object trace altogether:

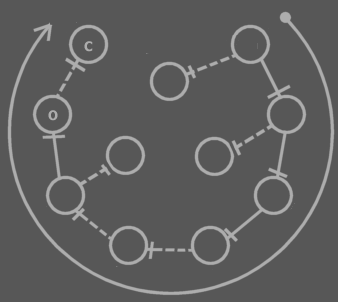


is the target type.

Therefore, *object* trace can also point out to the target *type*. The difference with a *type trace* is that a type trace prefers to follow type lines over object lines and an object trace prefers to follow object lines over type lines. However, both redirections lead to the exact same symbol.



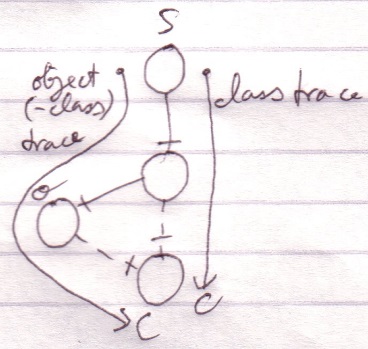
It happens a lot that you want to find out the object and the type in one blow. So you may as well use the redirection of the object trace for the benefit of finding the object and type in one blow. The trace is then called a *object-type trace*.



The last symbol in the redirection altogether is the target type (**C**). The last symbol pointed to by an object line is the object (**O**). Note that the target type may be pointed out by an object line.



When you only want to find out the type, it is better to use a type trace than it is to use an object-type trace. Type trace prefers type lines over object lines. Type lines generally follow less redirections before reaching the target type than object lines do.



##### Tracing is Not Always Hard

If an object symbol has no object line or type line, then finding the target object and type is much simpler, because no redirectioning at all takes place. The symbol is its own object and type.



Traces usually don’t require as many steps as in the examples above.

Targets,

2010-05

> I do not know how it works yet. Now my mind says: follow all redirections, including multiple interface redirections… but in the Target Class story I stopped doing that. Maybe it is just what you want the term Target Interface to define. Maybe it is not even important. I don’t know.

> Perhaps there should be a distinction between interface definition and target interface. I do feel that both the ‘follow only one class or interface step’ version is a concept to be aware of, but the target interface concept would actually be following all redirections to find the object that actually determines the publics.

> Yes. What is now called Target Interface should probably be called the *Interface Object* and the *Target Interface* is the object after following all types of redirections in any order.

JJ