|  |
| --- |
| Circle Language Spec |

## Classes

### Main Concept

The contents of an object might be arbitrary. Anything might be put inside an object.

But an object might also select another object to serve as its *class* or *prototype.* Then the contents of the object might not be that arbitrary. At first an object may contain related items and related lists that roughly corresponds with the class and the object might also have a similar set of commands as the class. The idea is that an object's behavior during its lifetime would also be bound to rules that might be set by that class.

A class is sort of like a special object, that aims to describe the characteristics and behavior of other objects.

An object might have a similar *structure* as its class, but that object may have *data* that can change freely. Values of an object's attributes and might freely change as well as which are an object's related objects . But initially an object may look like a replica of its class. The changeable parts of an object might be set initially to what is defined in the class.

### Class Redirection

An object’s pointing out its class, might be called a class redirections.

It might be expressed in a diagram by connecting an object symbol to its class with a dashed line:



The object on the left might then have the class on the right.

### Using an Object as a Class

Perhaps it is common that an object would be fixed in its role as a prototype or class. But the Circle notation would allow any object to serve as a class or prototype for another object.

### Using a Class Like an Object

Allowing object references to a class, would make it possible to reference a type like you could an object.

### Object Reference with a Class

Because next to an *object* having a class, an *object reference* might also have a class. If it does, only objects of that class might be assigned to it.

### Object Reference without a Class

When an object reference would not have a class, this might stand for its being able to point to *any* object.

### No Class != Defines its own Class

Formerly something might have been unclear. Objects might all be usable as classes. This might make it tempting to think of an object *without* a class would define *its* *own* class. Instead, it might be handy to not think of it that way, but think of it as object without a class simply not having a class. Having no class might stand for arbitrariness.

There might be examples where it could be less fortunate to not think about it that way.

### Object Reference would define its Own Class?

There may be other exampels, but it might be unfortunate to think of an *object reference* as defining its own class, because then the object reference could only point to … itself?

### Diagram Notation

The Circle notation might allow any object symbol serve as another object’s class or prototype. So in these diagrams any object might be used as a class or prototype for another object.



When an object would be used as another object’s class, it might be an idea to draw it out with a dashed line. *Dashed lines* might symbolize the concept of *classes*.



In these diagrams classes might usually look like that.

It may be an idea that if a symbol serves as another object’s class, but also is referenced as an object, the symbol would get a double border to maybe indicate its dual role as both an object and a class.



But perhaps just a dashed border is more appropriate after all, since its primary role seems to be a class. It might be a class, but there might also be *references* to that class (which might be established with an object redirection to that class).