|  |
| --- |
| Circle Language Spec: Interfaces |

## Mutual Commands, Mutual Interfaces (Unfinished)

This article is not finished.

Brainstorm 2010-05-06:

The idea described here is not what I want in practice. However, perhaps some derived concept could prove its use.

Perhaps you should be able to define some sort of generic interface.

If you define an interface with a command in it, that has a certain class, perhaps with some sort of mechanism or construct you should be able to reference any object that has a command with that class. You will get in trouble if an object has two commands with the same class, though.

One of the ideas behind this is idea is that objects, that have the same command, because they are both parameters of the command, have a joint interface, to create a relation. But really, they already have a relation, just an indirect relation, which is also a relation.

This idea just does not work. It only works if you assume that a command definition is unchangeable and that class references to the same command actually have to be the same procedure, but it hasn’t. You can have a command referenced twice that has elements to it that can still be redefined.

When this would not be the case then yes an idea like objects with the same command have a mutual interface could work, but in dynamic circumstances that are possible in the new computer language I just want to forget about it.

### A command is an interface

If multiple classes can support the same command, then the classes support the same interface: an interface that has only one command. The interface is represented by the command itself. Each command implicitly creates an interface. That is how it is expressed in a diagram too: a command implicity creates an interface.



You don’t see triangles around the commands inside the circles, that would specify, that it is a separate interface: each command automatically becomes a separate interface. Just like each object also defines an interface; even though there is no triangle directly inside it, it still has an interface.

### Multiple classes can support the same command

As you could see in the previous section: multiple classes, that support the same command, also support the same interface. Multiple classes supporting the same commands logically seems to create overlap in interface, but it doesn’t look like overlap in the diagram.



There are just multiple imaginary interface definitions (the circles), that each reference a different set of commands.

So a class referencing multiple commands, referenced by multiple classes, creates a visible interfaces of the class. The visible display of this interface is there, when you display the commands as part of a class, and not when you display the classes as parameters of the commands.