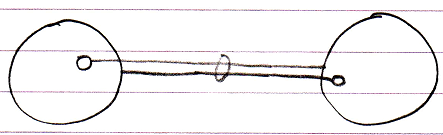
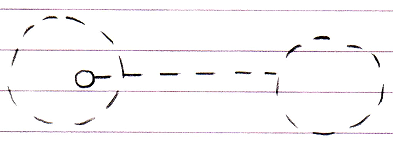
|  |
| --- |
| Circle Language Spec: Interfaces |

## Relations Through Interfaces

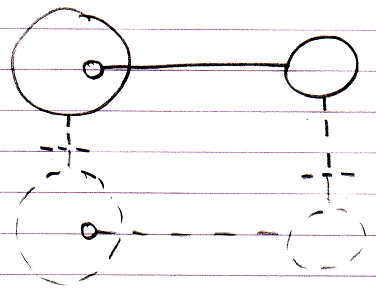
The most fundamental relation is a relation between two objects: one object contains a reference to another object, and the other object contains a reference back to the first object.



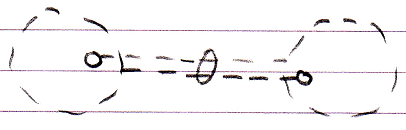
Classes can put control on which other class they are related to.



The effect is that in an object of that class, a specific reference can only point to an object of a specific other class.



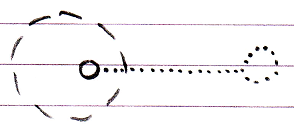
This is merely type re-enforcement. The picture below shows the classes in a bidirectional relation.



The example above illustrates what we call a relation between two classes. This relation between classes can be *‘felt’* throughout the object structure and is ultimately *defined* inside the classes.

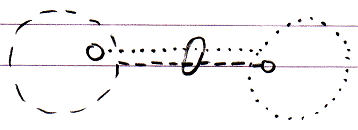
Relations through interfaces is the same concept but now applied to interfaces. There are two possibilities.

The first possibility is that a class defines the interface of one of its references.

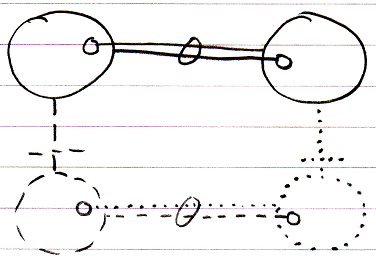


In that case it is said to be a relation between a class and an interface.

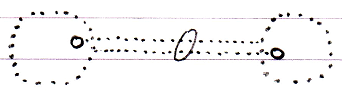
An interface can very well have a relation back to the class.



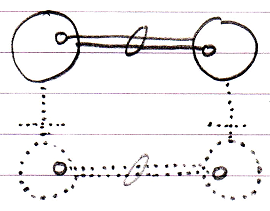
The effect of it is that objects that have that class or interface can only refer to an object of that a specific interface or class.



The second possibility is that an interface defines the interface of one of its references. And the other interface defines the relation back.

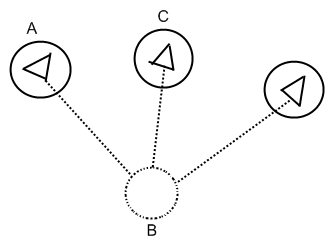


In that case it is said to be a relation between two interfaces. The effect of that is, that in objects with that interface, a specific public reference can only point to an object with that specific other interface.



That is all there is to it. Relations between classes and interfaces are sheer type-control on the relations between objects. Also, the mutual references are kept in sync, so that the two references always mutually point to each other and not to anything else (that’s what the ring is for).

If multiple classes support the same interface, this is what we traditionally see as relations through interfaces. It actually creates an *indirect* relation between classes.



In the picture above A and C are related through interfaces from a traditional point of view, but they are only not directly related in reality. They are only related through B.

## Ideas

*The texts below are ideas yet to be turned into good documentation.*

Relations Through Interfaces,

2010-05-07

There is one thing I don’t get: if a class relates to an interface, how does the interface know it is a class? The class is no more than an object, really. That it is a class depends on its being used as a prototype for other objects.

I bet that if you think about it long enough or if you see it in practice in a prototype version the solution will present itself.

JJ

Relations Through Interfaces,

2010-05-08

The solution is that a relation between the a class and an interface is actually no class or interface relation between a class and an interface at all: it is an object relation between two objects.

Wait, no, that is not the solution, they may be just two objects, but they do have a class and interface relation with eachother to fix which *kinds* of objects instances can link to.

JJ