|  |
| --- |
| Circle Language Spec: Inheritance |

## Inheritance Ideas

This article contains left-over rough ideas that are not incorporated into the documentation yet.

## Ideas

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

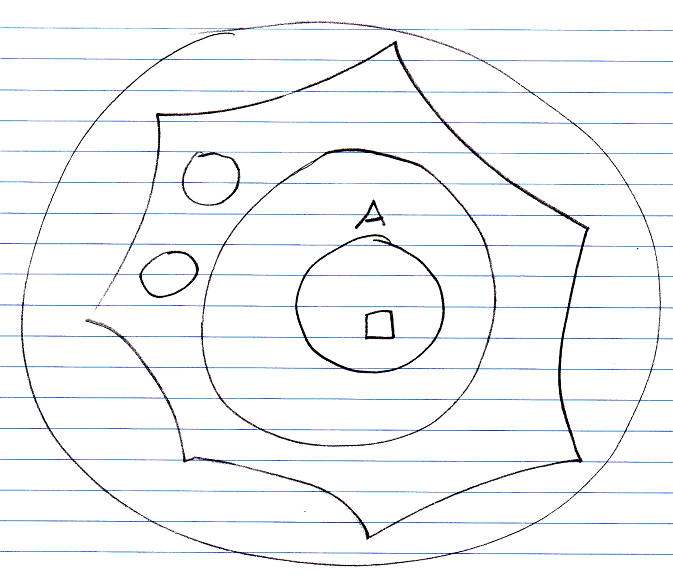
Inheritance,

2010-07-15

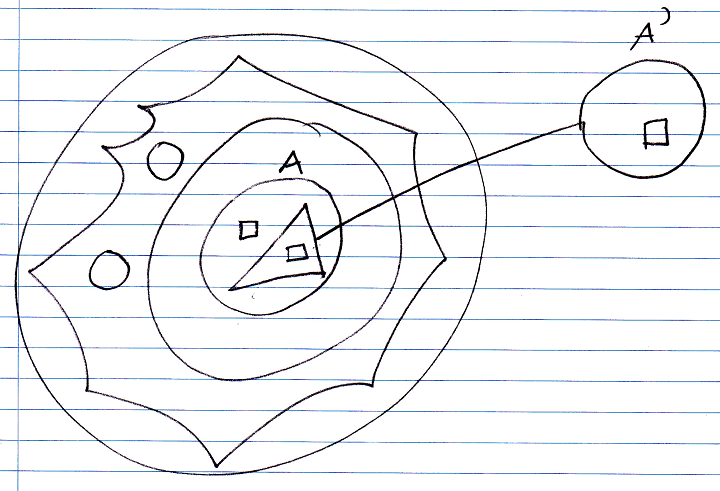
> The following example is wrong, because each layer in between must also become a specialized version.

Deeper Additional Members in an Object - List - Object – Object Situation:

Take the following system of objects:



We are going to add members to the object A. Object A indirectly resides inside a list. Adding additional members to such a deeper object should be the same as normal deeper member addition:



You take the original object A out of the structure, replace it with a new object inheriting from original object A.

JJ

Inheritance,

2010-07-16

Working out the following issues was postponed, because they are too hard to work out in theory compared to working them out in an actually running version.

Deeper specialization situations

- Windows Positions per User for an Existing System by means of System Inheritance

> This might mess up my idea of system inheritance

Inheritance,

1 data bron met een user interface,

maar verschillende window positions voor

verschillende users:

Een vorm van inheritance gebruiken

in de nieuwe programmeertaal.

Je moet namelijk window position

gegevens hebben, die parallel

gelegd wordt aan de structuur van de user interface.

JJ

- Adding new Backward Relations to an Existing System by means of System Inheritance

> This might mess up my idea of system inheritance

The user of a structure has to be able to make new relations.

Perhaps with inheritance.

- Je moet recursieve operaties TOE kunnen voegen aan een class, door inheritance, maar die inherited classes moeten dan gelden. Of je moet een structuur van inherited objecten hebben, die parallel loopt aan hoe de originele objecten aan elkaar gelinkt zijn.

- Also show that deeper member addition will work for lists in the structure.

- Deeper Exclusion and Deeper Inclusion is always

registered inside an invisible Deeper Specialization structure.

however, it could also be accomplished by overriding,

and then adding a different access modifier,

but this requires object set to be public.

- Misc Inheritance Situations:

- Multiple System Inheritance.

> I aim for adding multiple concepts, but just multiple system inheritance is not going to work.

> I probably should not think too hard about this in this chapter.

User Extensible Data Model. The data model of a program can always be extended with the users own ideas of what to link to what. The data the user thinkgs of, can be queried and

everything

JJ

Inheritance,

2010-05-22

I use the term Shadowing here, but it is called Hiding in C# and Java. Shadowing is the VB term, I guess.

JJ

Deeper Specialization,

2010-05-29

Does declaring something friend of an object, does that also allow you to do all deeper specializations? No.

JJ