|  |
| --- |
| Circle Language Spec: Black Boxes |

## Black Box Details, May Not Cover

In previous projects the main articles about black boxing were finished, but other subjects were left unfinished. This article contains some unfinished material. The topics are considered of such less importance, that they eventually might be deleted. This is because they might introduce discussion where no discussion is required. And that introduces confusion.

#### Access Modifier Get & Set

Interesting new system commands:

- Access Modifier Get

- Access Modifier Set

So would each system commands have its own system command

controlling the access modifier?

But what modifies the access to the access modifier system commands?

> Details...

##### Changing Access Modifiers at Run Time

Being able to change access modifiers at run time may be interesting. That way you can for instance change the interface of an object dependent on its settings. Changing a procedure’s access modifier is done by calling the procedure’s Access Modifier Set system procedure. There’s also an Access Modifier Get procedure to obtain the access modifier. This adds two system procedures to a square:

Access Modifier:

Get and Set

The Access Modifier property procedures can be access controlled themselves, so that they can be made inaccessible, for instance when you don’t want to be able to change the access modifier at run time. The access modifiers of the Access Modifier property can’t be changed at run time themselves. In other words: the Access Controller property procedures don’t have an Access Controller property of their own. This complies with the rule: system procedures don’t have system procedures of their own.

A single procedure can have several access modifiers so there are multiple access modifiers to get and set. This would suggest that the Accesss Modifier property has a dimension. However, when you call the Access Modifier property, you return or set the access modifier imposed by the object you’re calling from. The object you’re calling from is a hidden argument to the Access Modifier property. The access modifiers aren’t really stored inside the procedure, but in the object symbol that imposes it.

If the access modifier isn’t variable at run time, then the type defines a fixed access modifier. If the access modifier *is* variable at run time then the access modifier can be different for each instance of the type.

##### Changing ‘can be’ class, interface or object

System commands for changing access control to ‘can be’ class, interface or object.

#### Grouped Access Control

Using triangles to group members with equal acccess controllers. As with exclusion, the access controller of the triangle limits access again. It won’t make things *more* accessible.

#### Conditional Access Control

Replace the protected keyword by a concept like ‘when being a triangle’:

Basically you can optionally totally redefine the access for the case when the type is a used as a triangle.

<When being a globality?>

Or perhaps even make it as generall as ‘conditional access control’: Public if triangle.

Or perhaps I am looking too much for systematics behind it.

Perhaps the base of Use-As access control should (eventually) be based on conditional access control. But for now it can be limited to the conceptual explanation of the most usable things.

#### Delayed Creation of Object’s Privates

Public & Private,

2008-08-16

Perhaps for objects, private contents also only need to be created, when one of its friend commands is run.

>   
But this is such an influential implication, that it needs to be covered elsewhere. Privates only make sense, when only a selection of commands, can access those privates.  
So only when an object’s friend command is executed, the object’s private contents are needed.  
But a sub-object’s system commands can also access privates…

JJ

#### Access Symbols And Accesses

< Just find an appropriate time to mention that each access symbol represents an access, but that it’s also used to denote direction, simply because direction can be read from it >

2009-06-28

It was realized too little, that each line crossing

is an access...

but in the automatic containment setting it is not.

The crossings of imaginary reference lines are not accesses.

The crossings of lines going from the imaginary reference to

the logical residence... I am not sure if those are active accesses.

They could be.

But this is more like an implementation detail....

Not important enough.

JJ

Maybe it's a good rule to only show access symbols to denote:

- Direction

- Exclusion of a procedure

- Denote explicitly the publicity

2009-06-28

- Each line that is crossed is a separate access.

That is probably why I wanted to have those extra ticks in expressing

access modifiers, but access modifiers and actual connections

are two different things. So the syntax of those two things is a different deal.

Whatever

JJ

#### Access Controller Parts

In one access controller you can use an exclusion access controller, an object level restriction *and* a globality level restriction:

* Protected Up 1 Globality Up 2

Access controllers as such are not usual, though and *if* you use them, they’re not very diverse.

> 2009-06-29: So basically this brainstorm item is about how an access controller can consist of different elements: each aspect is represented only once.

#### Uses of Access Controlling System Aspects

An object symbol’s Object Get procedure must be accessible to make you able to access its sub-object.

Exclusion of Set procedures controls who can set a symbol’s lines. Exclusion of Get procedures controls who can get information about lines or who can access the object.

> 2009-07-06: Or who can use the object as line source.

#### Other Details

< Give a story about the true meaning of an access symbol that denotes direction. That it is a syntactically correct access symbol. Access symbols can used to denote direction, because access symbols *can* point out direction.>

< A procedure’s primary access controller is only made Inaccessible to ‘outcomment’ code. This is a rather small concept. The rest of the use of Inaccessible and Accessible is in exclusion. Therefore, I need to denote in short that this is the only use of the primary access controller. I can basically always speak of access modifiers when talking about Accessible and Inaccessible. I can use the term exclusion in a lot of places where I now use ‘access control’. I also need to put the first section ‘Accessible and Inaccessible’ in the exclusion section. >

> 2009-06-26: I do not even know what I mean by this:

You will not be able to totally make clear what

private and friend means in this context,

but you can just use the term command anyway,

to make it clear.

< The word access modification can actually be replaced by ‘exclusion’ in many places, but not everywhere by far. >

Detail:

The symbols used here are actualy access symbols, which I’ve already introduced. Access symbols can denote direction, and they also denote where access takes place, but they’re also used to denote access control.

Interesting Applications of Exclusion

So examples of interesting applications of exclusion are:

* Making procedures Public or Private
* Excluding procedures of a public object
* Excluding procedures of a base object
* A parent’s controlling *access to objects*

Access to *procedures* is controlled. But access control usually defines which *objects* get access to them. In a special case you can control which specific *procedures* get access.