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

## Black Boxing Main Concepts

#### The Black Box Principle

Most computer programming languages make it possible to apply the *black box* principle. Using the black box principle allows you to hide and protect the inner workings of an object. You only get to see some properties, parameters and a selection of commands through which you can control the object while the inner workings are hidden away and protected.

There are two key purposes of the black box principle: data protection and complexity hiding.

Complexity hiding can also be called abstraction.

#### Private & Public

The inner workings of an object or command are its *private* members. You only get to work with an object or command’s *public* members.

So the privates of an object are protected and hidden away, while the public contents can freely be used.

#### Friends

There is no point to private content if it is not used by anything. Even though private contents are protected, *something* needs to access and protect that data.

An object’s public contents can be accessed by anything that has a reference to the object. Private contents can only be used by *friends*. Friends are objects that have access to the private contents of another object.

What other programming languages call class methods, the new computer language calls friend commands of a class object.

The object with private contents declares which other objects are its friends.

The friendship does not automatically go both ways: the *befriended* has access to the privates of the friend, but the friend does not have access to the privates of the befriended. For that you would have to establish a mutual friendship.

#### Friend Relations

A befriended object declares which object is its friend. The friend object needs to be able to access the befriended’s privates. Therefore the friend needs a reference back. So a friend connection requires a mutual relationship.

It is not like one class gets access to the privates of another class. It is more specific than that. A friendship is a relation between specific objects. A friendship between objects applies to only one specific relation. One object gets access to the privates of another object through one specific reference.

Friendship can also be applied to related lists instead of just related items.

An item or list in a specific relation is typed as friend.

#### Accessible & Inaccessible

Black boxing is based on two simple concepts: *accessible* and *inaccessible*. You can express those concepts as follows:

|  |  |
| --- | --- |
| Accessible | Inaccessible |

The concepts are not litterly used inside the language, but all the other black boxing concepts arebased on it.

The notation of Accessible and Inaccessible *can* be used in abstract diagram expression, though, which does not have to be syntactically correct, but is only used to make a point.

#### Access Controlling System Aspects

Each aspect of a symbol can be separately access controlled. Those are aspects such as:

- Object

- Class

- Value

Each aspect’s Get and Set are separately access controlled, for instance:

- Object Get

- Object Set

- Class Get

- Class Set

- Value Get

- Value Set

So when Object Get is Public, you can reference the sub-object.

When Value Set is Public, you can assign a value to the object.

When Value Set is Private, you can not assign a value to the object except through Friend objects.

Only *system procedures* get access controllers. Objects do not get access controllers directly. Access to an *object* is controlled by access controlling the Object Get procedure.

There is not just one kind of Private or Public. There is a separate Private and Public for every Get and Set of every system aspect.

#### Summarized

System aspects of sub-objects are access controlled by their parent object.

You access control system aspects, making their Get or Set, Private or Public.

A normal reference only offers access to the Public content of an object.

You can type an item or list in a specific relation as Friend.

When a Friend has a reference to an object, all its sub-objects are accessible, even when they are Private.

The full access that Friends get, applies to accessing the direct sub-objects, not deeper sub-objects.