|  |
| --- |
| Circle Language Spec: System Objects |

## Assignment

Calls to **Get**, **Set** and **Use** commands might not be seen directly usually. Those may be more likely to be called indirectly by *assignment* commands. An assignment command might execute a **Get** on one object and a **Set** on another object, attempting to yield over an aspect from one symbol to another.

Different aspects, such as a value held or an object pointed to, might have different types of assignment. Below an attempt to list more common types of assignment.

The overview also attempts to illustrate, which **Get**, **Set** and **Use** commands might be called while performing an assignment.

To demonstrate the difference between object-bound aspects and reference-bound aspects, the following notation is employed. When a reference-bound aspect might be **Get** or **Set** then a reference may be displayed with a parent around it:



When an object-bound aspect might be **Get** or **Set** then a targeted object may be displayed without a parent around it:



|  |
| --- |
| **Value Assignment** |
| <Add a diamond in the middle?> |
| Value Get 🡨  Value Set 🡪 |
| *Aims to copy a value of one object*  *to another.* |
|  |
| Object Assignment |
|  |
| Object Get 🡨  Object Set 🡪 |
| *Tries to let the target point to*  *the same object as the source.*  *So attempts to yield over the object aspect.* |
|  |
| Class Assignment |
|  |
| Use As Class (~= Object Get) 🡨  Class Set 🡪 |
| *Tries to use the source as*  *a class for the target.* |

In this assignment notation the line type (dashed, solid, wavy) might indicate which aspect would be yielded over. The access mark (the smaller line crossing the longer line) could indicate the direction of the assignment.

**Value** assignment might not need an assignment call symbol (the diamond shape in the middle), because a **Value** connection might always an assignment.

### Cross-Aspect Assignments

Perhaps a more common way to use the **Class** aspect in an assignment might be to **Get** the **Object** aspect from one reference and assign it to the **Class** aspect of another reference. But it might also be done the other way around: attempting to get the **Class** aspect from one reference and assign it as the **Object** aspect of another reference. It might be called *Class-to-Object* assignment or a *Class-Get* *assignment* perhaps. Less conventional ways of yielding over aspects like that, might be called *cross-aspect* assignment.

Also noteworthy might be, that there could be two ways to get the **Class** aspect: **Get** the **Class** aspect of an *object* or **Get** the **Class** aspect of a *reference*.

|  |
| --- |
| **Reference-Bound Class to Object Assignment** |
|  |
| Reference-Bound Class Get 🡨  Object Set 🡪 |
| *This might be the result:* |
| <Perhaps do not use a double border>  *The object reference on the right may now point to the class of the object reference on the left.* |
|  |
| **Object-Bound Class to Object Assignment** |
|  |
| Object-Bound Class Get 🡨  Object Set 🡪 |
| *This might be the result:* |
| <Perhaps do not use a double border>  *The object reference on the right may now point to the class of the object on the left.* |