|  |
| --- |
| Circle Language Spec: Parameters |

## Required & Optional

### Concept

Some programming languages have a thing called required and optional parameters. In other programming languages basically all parameters are required, unless they are typed optional. In the new computer language it is the other way around: any parameter is optional unless it is typed required. This is because in the new computer language commands are the same as objects, that just happen to be executable, and the parameters inside a command are analogus to sub-objects inside another object, whose filling-in is also totally arbitrary.

Some parameters can be made required, so you *have to* fill them in as input. In the new computer language, this means that reading and writing any accessible aspect of a parameter can be separately made required or optional. When output is required, it means you have to use the output. You will be unable to execute the command without picking up the return value or returned object.

Basically each system command can be made required or optional. Usually you make access from the *outside* required. When you make access from the *inside* required, this can give the user of the command guarantees about the parameter’s usage.

### Required & Optional in a Diagram

*Optional* is the default behavior. *Required* is an extra rule imposed. Required is expressed by drawing half a shape at the end of a connector. It should be half of what has to be connected to it. Most of the times it will be half a circle:



For command objects it is half a square:



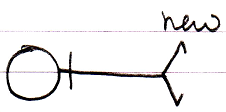
And when a command needs to be executed it is about 5/8th of a diamond:



It is not half a diamond because that already symbolizes the Protected access modifier, which is half a cross:



For system commands that require the command call connector notation, also the 5/8th of a diamond is shown when a call is required:



The ‘half a shape’ will also get a line type adapted to the role the required object will get. If the connector specifies that it gets a class role, the half-shape will be drawn with a dashed line.



Below you will find examples of *required* connectors. Not every possible connector is shown, but enough to cover the notation.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Public Required Object Set | Protected Required Object Set | Public Required Object Get |
|  |  |  |
|  |  |  |
| Private Required Object Set | Friend Required Object Set | Private Required Object Get |
|  |  |  |
|  |  |  |
| Protected Required  Object Get | Friend Required Object Get | Public Required Class Set |
|  |  |  |
|  |  |  |
| Public Required Use As Class | Private Required Class Set | Friend Required Value Set |
|  |  |  |
|  |  |  |
| Public Required Clone (2) Get | Public Required New | *(as opposed to)*  Public New |
|  |  |  |
|  |  |  |
| Friend Required Execute | Public Required Object Get | Public Required Use As Class |
|  |  |  |
|  |  |  |
| Public Required Use As Class  (might be a better notation, since it is more likely to be used for plain command calls) |  |  |

## Ideas

### Out of the original Symbol documentation

* Just as in procedures, you could make certain members of a type required, while others are optional. You have to fill in the required members on creation of the object. This should be there for the same reason as required parameters are there in a procedure: the function of the object just doesn’t make sense unless you write the required members. The programmer is made extra aware of that by making the members required.