|  |
| --- |
| Circle Language Spec: Commands |

## Sub-Commands Are Never Referenced

This is a rule for enforcement of control of a parent command over the execution of its sub-commands. Active clauses and command calls inside another command and are never referenced, because a command has to have full control over the execution of its sub-commands. If you could reference an active command inside a command, then the sub-command could be prematurely executed through that reference. Therefore sub-commands are never referenced.

It is *not* the rule, that command calls can never be referenced. It’s just that command calls *inside another command* can not be referenced. When a command call resides in an object, the command call *can* be referenced, to for instance allow a user to carry around a reference to an active command, executing on a site somewhere. So a command call inside an object can be referenced, but a command call inside another command can not be referenced.

Not being able to reference sub-commands does not mean, that you *can* use it as a class, because a class reference is also a reference. This would also put the sub-command in danger of being prematurely executed, because you could establish an active reference to the class of another command object and execute it.

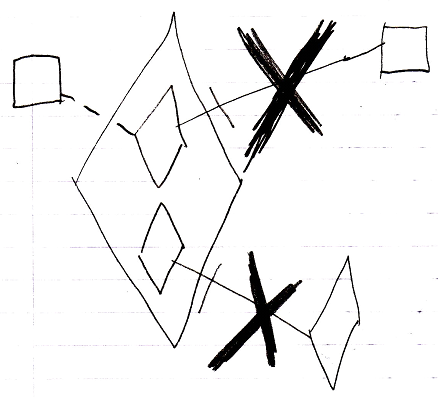
To not cause any confusion, sub-commands are made Private.

Do not change the rule to *sub-commands are always private*, because this would not fully solve the parent command’s control over its sub-commands’ execution. By just making them private, the parent command could still pass a reference to a *sub-*command, so that the parent *gives up* control over the execution of a *sub-*command. This is something, that will not be allowed. Now that you can *never reference a sub-command*, this control is restored.

### Diagram Notation

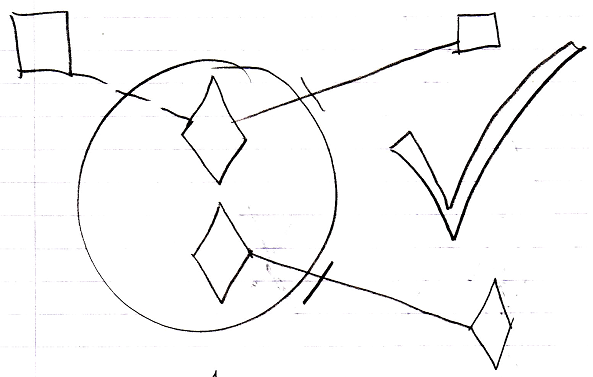
The current paragraph repeats the story, but now demonstrates the concept using diagrams.

Active clauses and command calls inside another command and are never referenced, because a command has to have full control over the execution of its sub-commands.



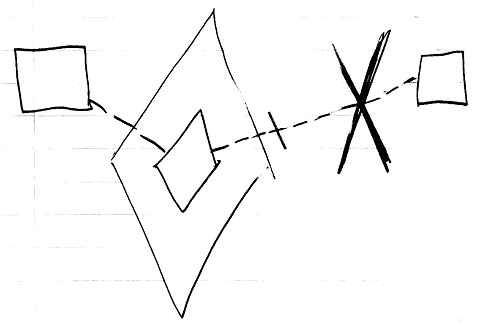
If you could reference an active command inside a command, then the sub-command could be prematurely executed through that reference. Therefore sub-commands are never referenced.

It is *not* the rule, that command calls can *never* be referenced. It’s just that command calls *inside another command* can not be referenced. When a command call resides in an object, the command call *can* be referenced, to for instance allow a user to carry around a reference to an active command, executing on a site somewhere.



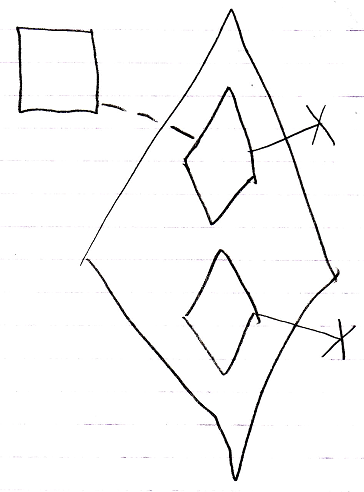
So a command call inside an object can be referenced, but a command call inside another command can not be referenced.

Not being able to reference sub-commands does not mean, that you can use it as a class, because a class reference is also a reference.

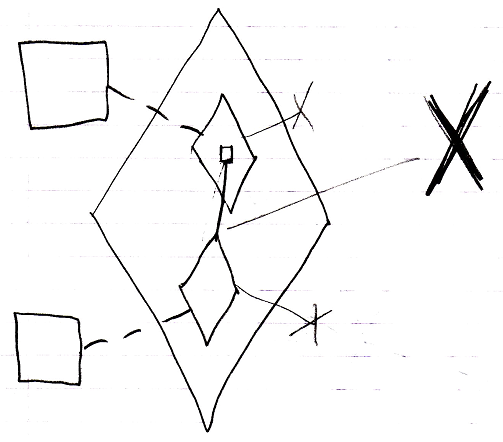


This would also put the sub-command in danger of being prematurely executed, because you could establish an active reference to the class of another command object and execute it.

To not cause any confusion, sub-commands are made Private.



Do not change the rule to *sub-commands are always private*, because this would not fully solve the parent command’s control over its sub-commands’ execution. By just making them private, the parent command could still pass a reference to a sub-command.



That would make the parent *give up control* over the execution of a sub-command. This is something, that will not be allowed. Now that you can *never reference a sub-command*, this control is restored.

