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

## Creation Behavior of ‘Inactive Calls’

Command calls inside a parent command have special creation behavior: their publics are created as soon as the parent command is created, but their privates are only created when the command call is about to be run.

There is also the *inactive* form of a command call inside a parent command.

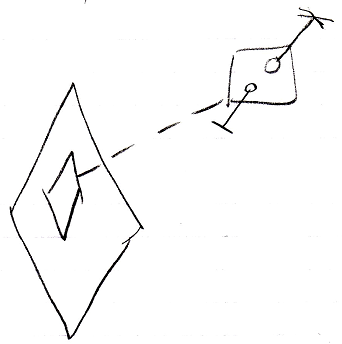
Special creation behavior does *not* count for inactive command objects inside a parent command, that have a class redirection to a command definition. This looks like the inactive form of a command call, but this kind of object does not have special creation behavior like that. It is an uncommon situation. But an inactive command inside a parent command with a class reference to a definition *can* be referenced, unlike its active form. Therefore, it can also be *class* referenced by a call. In that case its privates and publics had better be there more permantly, or the *call* to it can not instantiate private contents at all.

Creating its private contents, does not create a recursive creation or anything: the inactive call’s own private calls do not create *their* private contents, so there is no recursion there.

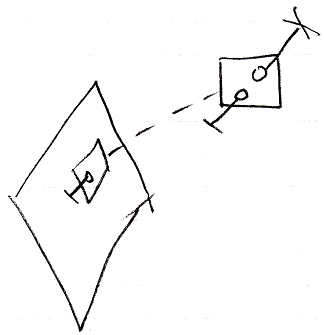
### In a Diagram

The article *Creation Behavior of ‘Inactive Calls’* already explained this behavior conceptually. The current article further clarifies the idea using diagrams.

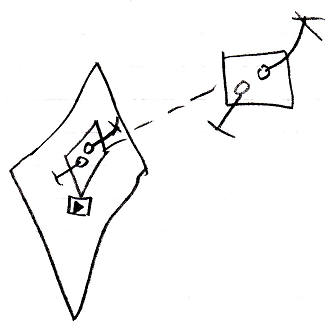
Command calls inside a parent command have special creation behavior:



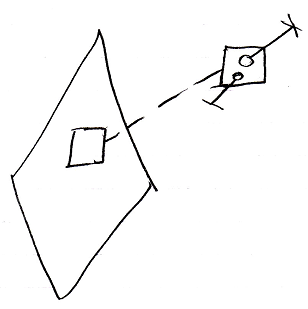
Their publics are created as soon as the parent command is created



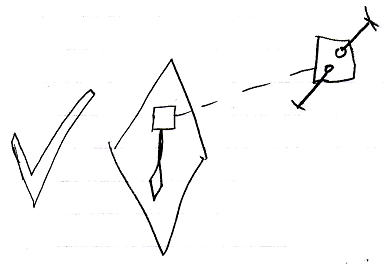
but their privates are only created when the command call is about to be run.



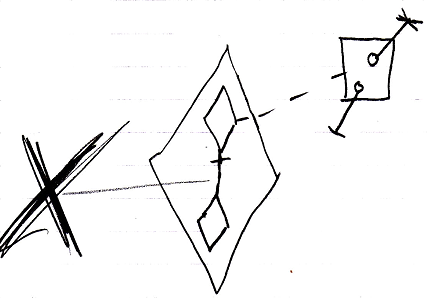
Special creation behavior does *not* count for inactive command objects inside a parent command, that have a class redirection to a command definition.



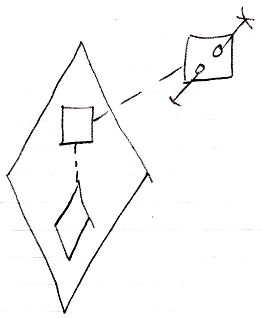
This looks like the inactive form of a command call, but this kind of object does not have special creation behavior like that. It is an uncommon situation. But an inactive command inside a parent command with a class reference to a definition *can* be referenced



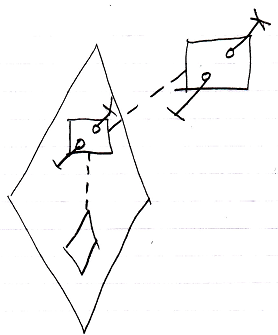
Unlike its active form, which can not be referenced.



Therefore, it can also be *class* referenced by a call.



In that case its privates and publics had better be there more permantly, or the *call* to it can not instantiate private contents at all.



Creating its private contents, does not create a recursive creation or anything: the inactive call’s own private calls do not create *their* private contents, so there is no recursion there.

