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

## Executable Command Class Redirection in a Diagram

### Concept

If an executable command redirects its class, then it is a command call. The class redirection points to the command definition of the command call.

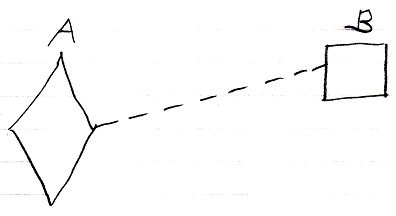
If an executable command *does not have* a class redirection, then it defines its own contents. An object redirection automatically also redirects the class aspect of a symbol. So then it also does not define its own contents. If an executable command, that does not redirect its class or object resides inside another command, then it is called an *active clause*. If the executable command without redirections resides inside an object, then it is simply an executable command with no definition, that can be manually run by a user.

You could let an executable command redirect its definition to another executable command, but this is far less common. But an active command *can* function as the prototype for another executable command.

### In a Diagram

Executable command class redirection is explained in the article *Executable Command Class Redirection*. The current article demonstrates the concept in a diagram.

The picture below displays a command call, redirecting its definition to an inactive command.



Active command symbol A is a diamond shape, because it is an *active* command. Active command symbol A redirects its definition to the inactive command symbol B. Symbol B is a square because it is an *inactive* command. The redirection from command A to command B is displayed as a dashed line, which is a class line. This makes symbol A a command call, for which command B functions as a definition.

You could let an executable command redirect its definition to another executable command, but this is far less common:

