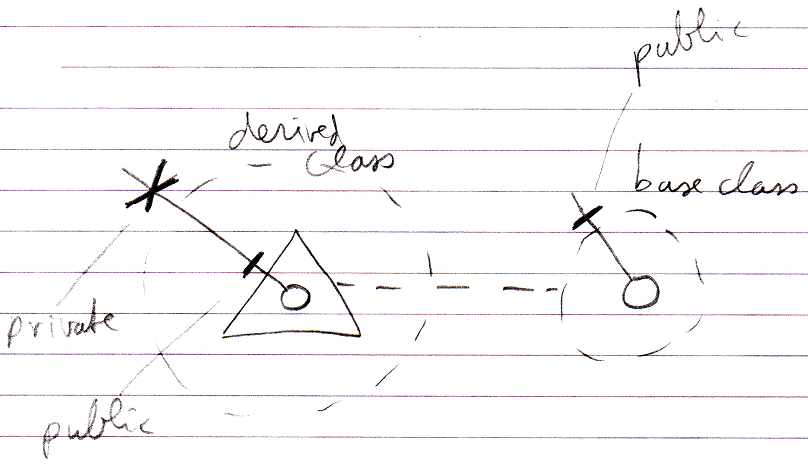
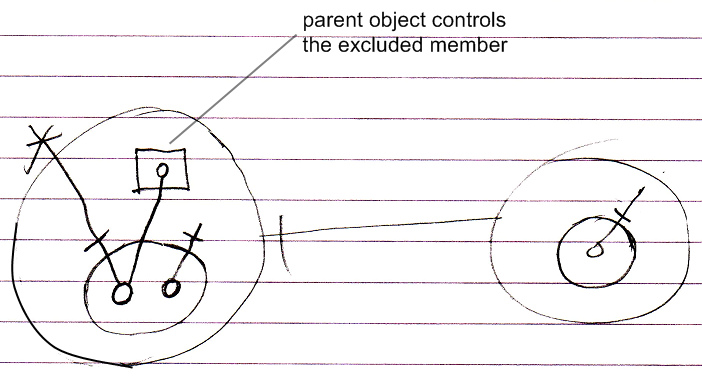
|  |
| --- |
| Circle Language Spec: Inheritance |

## Member Exclusion

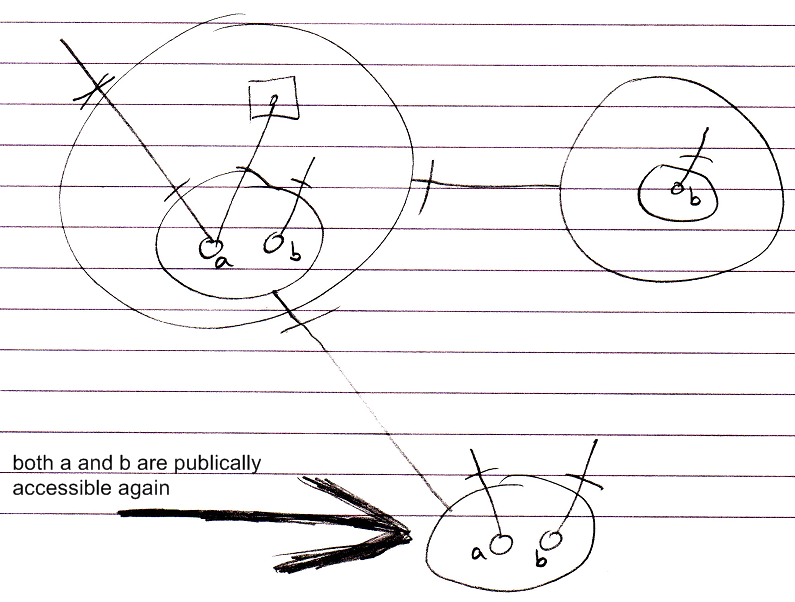
Another technique for specialization by altering the member set is *member exclusion*. This is not possible in most other object oriented languages. What happens is that members from deeper objects that were formerly Public, are Private through the derived class.



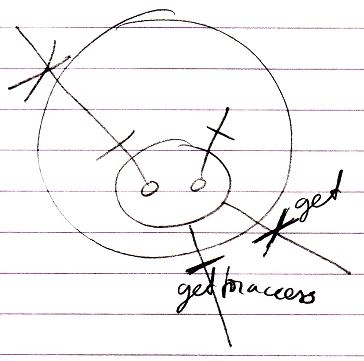
This also makes it possible to protect some of the members of the sub-objects, and let the parent object be the only one to control them.



Do note that this protection is dependent on the context, so when you can reference the object from another location, you can still access the excluded member:



You can prevent this by making the Object Get Private while Object Get For Access is still Public.



(there is no notation yet for the distinction between Object Get, Object Get For Access and Object Get For Copy)