The good news: One can easily navigate with OCL in a class diagram by employing role names. Starting from a single object or a single class variable, one can build OCL expressions by appending role names.

The bad news: It is not so easy to fully understand the type of the resulting OCL expression.

Example

In the example, the most common single- and multi-valued multiplicities are present: 0..1, 1, 0..*, 1..*.

dot shortcut & collect VS collectNested:

- if 'role' is single-valued (0..1 or 1): 'multiValuedExpr.role' ~~> 'multiValuedExpr->collectNested(x|x.role)'

- if 'role' is multi-valued (upper bound > 1): 'multiValuedExpr.role' ~~> 'multiValuedExpr->collectNested(x|x.role)->flatten()'

- 'expr1->collect(x|x.expr2)' ~~> 'expr1->collectNested(x|x.expr2)->flatten()'
The following OCL expressions employing a single role name or two role names can be built. There are no other expressions with one or two role names. The type of the expression can be single-valued, set-valued or bag-valued. For ease of identification, the single-valued role names are underlined.

```
use> ?e.project
    Set{p}: Set(Project)
use> ?e.managed
    d: Department
use> ?p.worker
    Set{e}: Set(Employee)
use> ?p.controller
    d: Department
use> ?d.boss
    e: Employee
use> ?d.project
    Set{p}: Set(Project)

use> ?e.project.controller    dot shortcut
    Bag{d}: Bag(Department)  e.project->collectNested(p|p.controller)
use> ?e.managed.project
    Set{p}: Set(Project)
use> ?p.worker.managed      dot shortcut
    Bag{d}: Bag(Department)  p.worker->collectNested(e|e.managed)
use> ?p.controller.boss
    e: Employee
use> ?d.boss.project
    Set{p}: Set(Project)
use> ?d.project.worker      dot shortcut and flatten
    Bag{e}: Bag(Employee)  d.project->collectNested(p|p.worker)->flatten
use> ?e.project.worker      dot shortcut and flatten
    Bag{e}: Bag(Employee)  e.project->collectNested(p|p.worker)->flatten
use> ?e.managed.boss
    e: Employee
use> ?p.worker.project      dot shortcut and flatten
    Bag{p}: Bag(Project)  p.worker->collectNested(e|e.project)->flatten
use> ?p.controller.project
    Set{p}: Set(Project)
use> ?d.boss.managed
    d: Department
use> ?d.project.controller    dot shortcut
    Bag{d}: Bag(Department)  d.project->collectNested(p|p.controller)
```