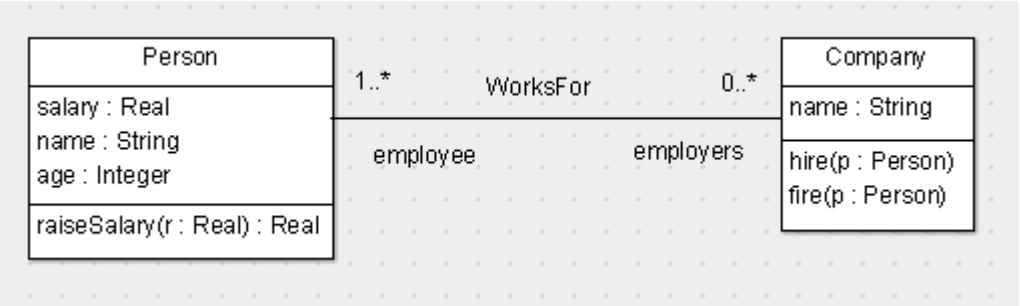
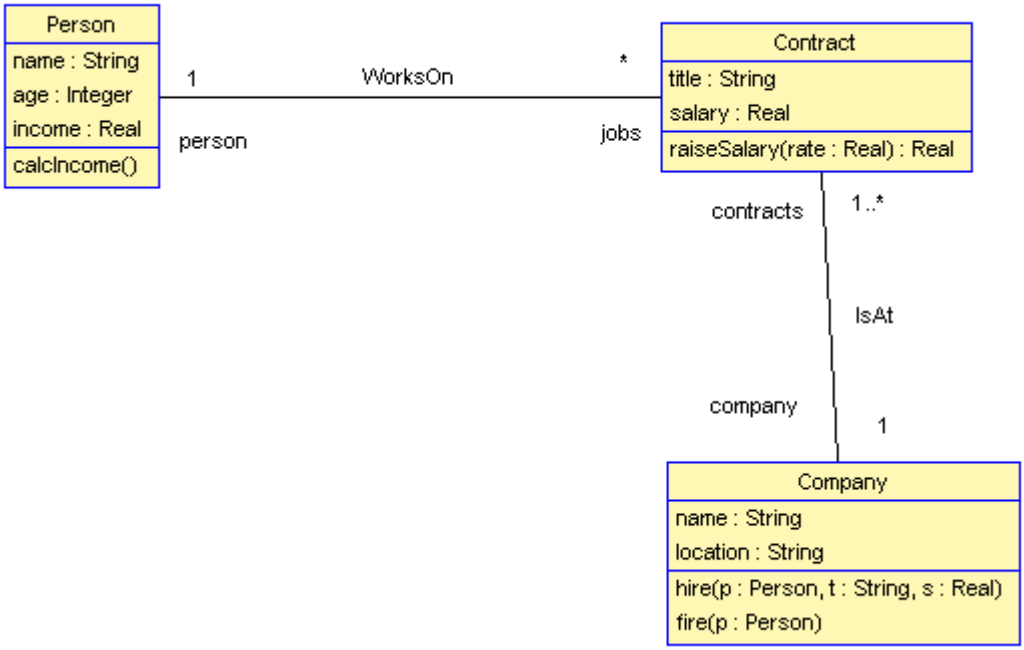
USE Modelling Task

You are required to elaborate/refactor the class diagram below by adding an extra class, possibly Job or Contract. You may have to move attributes, change operation signatures. Copy Employee.use to a new folder EmployeeContracts and modify it as shown below.



One possibility is:



Do not implement hire() or fire() in SOIL just yet. Adapt OCL constraints from your previous model. You may need to add new ones. Make sure that the OCL includes the following constraints for **hire(p,t,s)**:

- on being hired, a person's salary for that job should at least be 20000
- before hire(p, t, s) being executed, p must not be an employee of the company
- after hire(p, t, s) being executed, p is linked to the company thru a contract
- only persons aged 21 or older can get a salary increase

These constitute a contract for **hire(p,t,s)**.

A possible hire() contract is

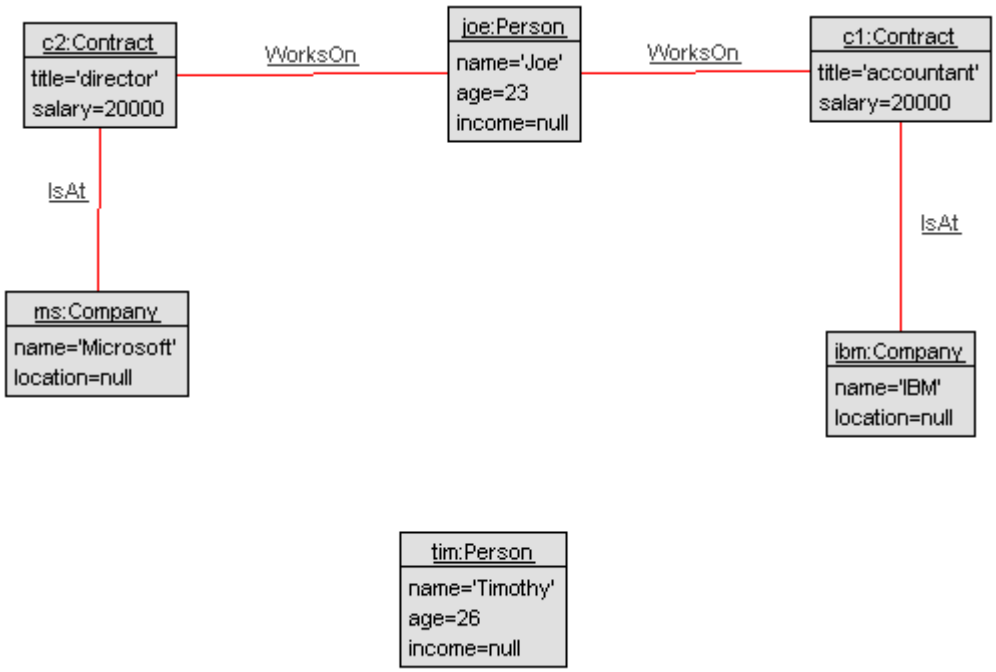
```
context Company::hire(p : Person, t : String, s : Real)
  pre  hirePre1: p.isDefined()
  pre  minSal: s >= 20000
  pre  notAlreayAnEmployee: contracts.person->excludes(p)

  post nowAnEmployee: contracts.person->includes(p)
  post correctSal:
       contracts->forAll(c | c.person = p implies c.salary = s)
```

Testing the hire(p,t,s) Contract

Save your USE code in Employee.use and load into USE.

Load the sample objects provided in Employee.soil for testing.



!openter/!opexit tasks

Your task is now to explore an implementation for **hire(p,t,s)** which satisfies the above contract.

Try to hire joe again for ms.

```
use> !openter ms hire(joe, 'Janitor', 30000)
precondition `hirePre1' is true
precondition `minSal' is true
precondition `notAlreayAnEmployee' is false
Error: precondition false in operation call `Company::hire(self:ms, p:joe, t:'Janit
or', s:30000)'.
```

Doesn't workout as he already work for ms.

Try to hire tim instead as a secretary but at 18000 per year.

```
use> !openter ms hire(tim, 'Secretary', 18000)
precondition `hirePre1' is true
precondition `minSal' is false
precondition `notAlreayAnEmployee' is true
Error: precondition false in operation call `Company::hire(self:ms, p:tim, t:'Secre
tary', s:18000)'.
```

Again this doesn't work as salary is too low.

Try again at a better salary and do !opexit without updating any objects.

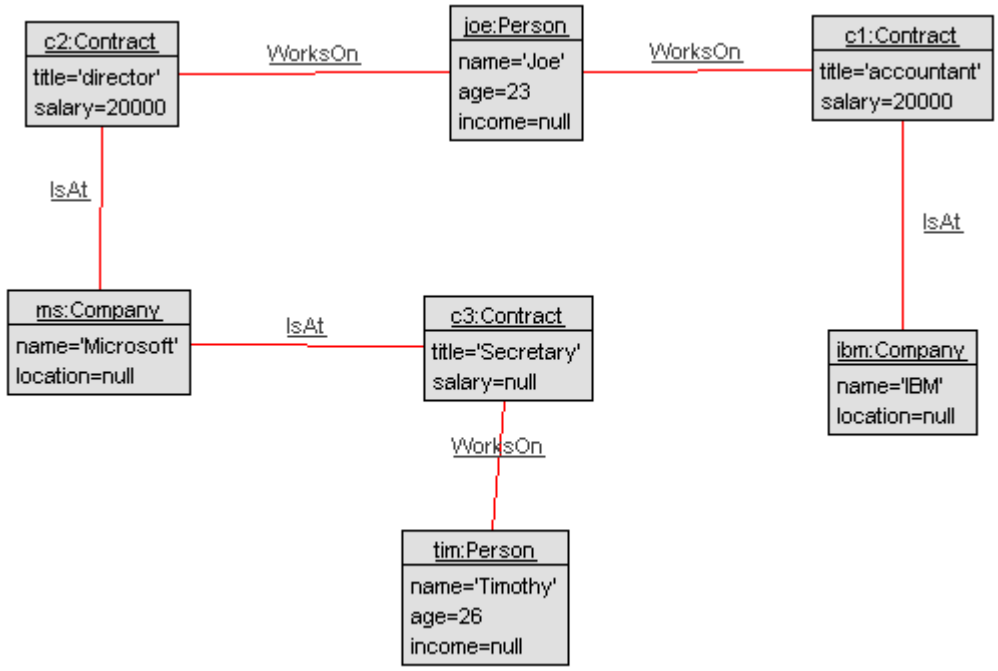
```
use> !openter ms hire(tim, 'Secretary', 25000)
precondition `hirePre1' is true
precondition `minSal' is true
precondition `notAlreayAnEmployee' is true
use> !opexit
postcondition `nowAnEmployee' is false
  self : Company = ms
  self.contracts : Set(Contract) = Set{c2}
  $e : Contract = c2
  $e.person : Person = joe
  self.contracts->collect($e : Contract | $e.person) : Bag(Person) = Bag{joe}
  p : Person = tim
  self.contracts->collect($e : Contract | $e.person)->includes(p) : Boolean = false
postcondition `correctSal' is true
Error: postcondition false in operation call `Company::hire(self:ms, p:tim, t:'Secretary', s:25000)'.
```

Notice that postcondition **correctSal** is true even though tim has not been properly hired. Why?

Try !openter again but this time update some objects and associations.

```
use> !openter ms hire(tim, 'Secretary', 25000)
precondition `hirePre1' is true
precondition `minSal' is true
precondition `notAlreayAnEmployee' is true
use> ! create c3 : Contract
use> ! c3.title := 'Secretary'
use> !insert (c3, ms) into IsAt
use> !insert (tim, c3) into WorksOn
```

Object diagram will be update accordingly.



Next try !opexit

```
use> !opexit
postcondition `nowAnEmployee' is true
postcondition `correctSal' is false
```

Why is **correctSal** failing?

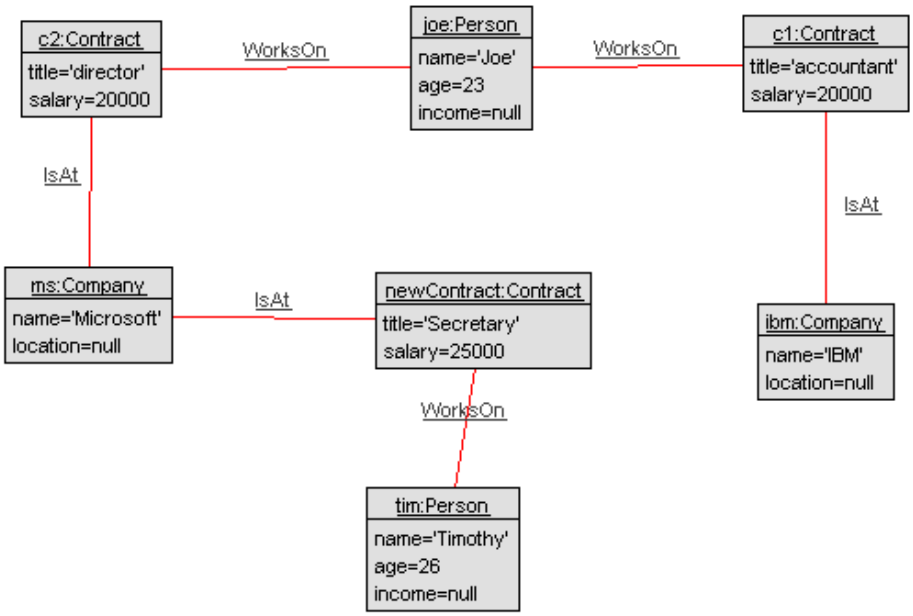
To get things right, undo the above with

```
use> !delete (tim,c3) from WorksOn
use> !delete (c3,ms) from IsAt
use> !destroy c3
use>
```

And try once more (last time) with:

```
use> !openter ms hire(tim, 'Secretary', 25000)
precondition `hirePre1' is true
precondition `minSal' is true
precondition `notAlreayAnEmployee' is true
use> !create newContract : Contract
use> !newContract.salary := 25000
use> !newContract.title := 'Secretary'
use> !insert (tim, newContract) into WorksOn
use> !insert (newContract, ms) into IsAt
use> !opexit
postcondition `nowAnEmployee' is true
postcondition `correctSal' is true
use>
```

All went well. Check you object diagram.

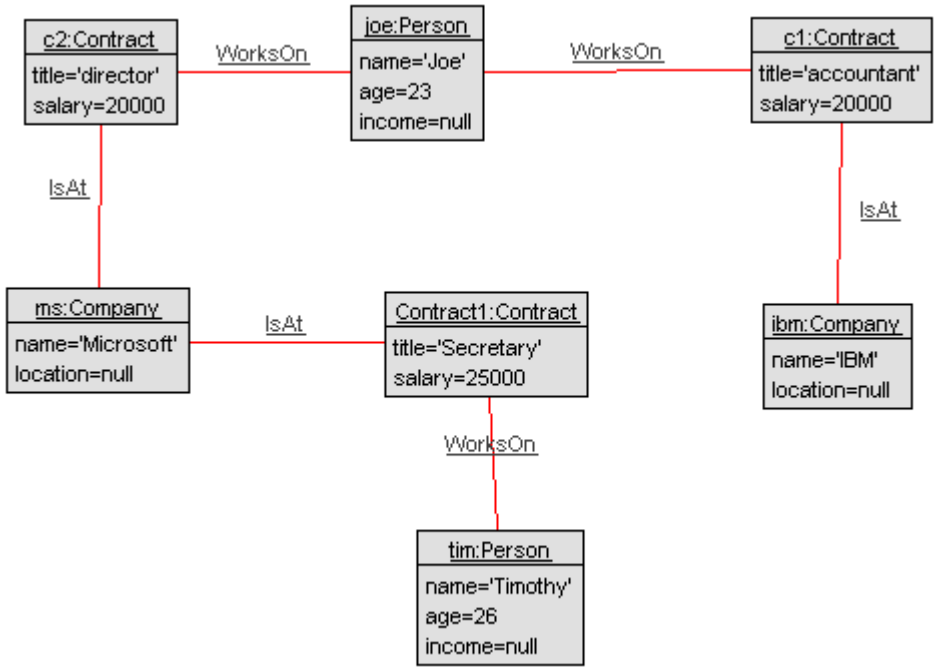


Implement hire(p,t,s) in SOIL

It would be better if **hire(p,t,s)** was implemented and did all this work for you. So do this and test it with:

```
use> ! ms.hire(tim, 'Secretary', 25000)
use>
```

to yield



Exercises

Write an invariant that says a person can only work 1 job at a given company (the class diagram does not show this constraint).

Try !openter/!opexit on fire(p) and when you are satisfied with what is going on, implement it.