```
Constructors and `destructors'
- Set{7,8}, Bag{7,8,8}, Sequence{7,8,7}, OrderedSet{8,7,7}
- Set{}, Bag{}, Sequence{}, OrderedSet{}
- Set{7..9}, Bag{7..9}, Sequence{7..9}, OrderedSet{7..9}
- Set{}->including(8)->including(7), Bag{8,9,7,8,9}->excluding(9)

Basic boolean and integer query operations
- Set{7,8}=Set{8,7,8,7}, OrderedSet{7,8}<>OrderedSet{8,7}
  Set{7,8}<>Bag{7,8}, OrderedSet{7,8}<>Sequence{8,7}
- Set{7,8}->includes(8), Set{7,8}->excludes(9),
  Set{7,8}->includesAll(Set{8,8,7,7}), Set{7,8}->excludesAll(Set{6,9})
- Set{}->isEmpty(), Set{7,8}->notEmpty(), Set{8,8,7,7}->size()=2
  Set{7,8,7}->count(7), Bag{7,8,7}->count(7)
  Sequence{7,8,7}->count(7), OrderedSet{7,8,7}->count(7)

Advanced boolean query operations
- Set{7..9}->forAll(i|i>=0), Bag{7..9}->exists(i|i.mod(2)=0)
- Sequence{7..9}->one(i|i.mod(2)=0)
- OrderedSet{-9..-8}->including(8)->including(9)->isUnique(i|i*i)=false

Advanced collection-valued query operations
- Set{21..42}->select(i|i.mod(3)=0 and i.mod(7)=0)
- Bag{21..42}->reject(i|i.mod(2)=0 or i.mod(3)=0)
- Set{21..42}->any(i|i.mod(2)=1)
- Set{7,8,8}->union(Set{9,9,8}), Bag{7,8,8}->union(Bag{9,9,8})
  Sequence{7,8,8}->union(Sequence{9,9,8})
  OrderedSet{7,8,8}->union(OrderedSet{9,9,8})
- Set{-2..2}->collect(i|i*i), Set{-2..2}->collect(i|Sequence{i,i*i})
  Set{-2..2}->collectNested(i|Sequence{i,i*i})
- Set{-2..2}->collectNested(i|Sequence{i,i*i})->flatten()
- Set{-6,-5,-4,7,8,9}->sortedBy(i|i*i)

Complex query operations
- Set{-2..2}->iterate(i:Integer;r:Set(Sequence(OclAny))=Set{}|
    r->including(Sequence{i,i*i,if i.mod(2)=0 then 'E' else 'O' endif}))
- Capitals: M[adrid], P[aris], A[msterdam], B[erlin], Z[urich], V[ienna]
  let TupleSet=
    Set{Tuple{s:'M',t:'P'},Tuple{s:'P',t:'A'},Tuple{s:'A',t:'B'},
        Tuple{s:'M',t:'Z'},Tuple{s:'Z',t:'V'},Tuple{t:'B',s:'V'}} in
  TupleSet->closure(T1|
    TupleSet->select(T2|T1.t=T2.s)->
      collect(T2|Tuple{s:T1.s,t:T2.t})->
        asSet())
                    +-------------+
                    |  select =   |
      Tuple{T1.s,T1.t}   Tuple{T2.s,T2.t}
               |        collect        |
               +-----------------------+

Coercions
- Sequence{8,7,8}->asSet()=Set{8,7}
- OrderedSet{8,7,8}->asBag()=Bag{8,7}
- Set{7,8}->asSequence()=Sequence{8,7}
  or Set{7,8}->asSequence()=Sequence{7,8}
- Bag{8,8,7,7}->asOrderedSet()=OrderedSet{7,8}
  or Bag{8,8,7,7}->asOrderedSet()=OrderedSet{8,7}
- Set{-2..2}->collect(i|i*i)->asSet()
```

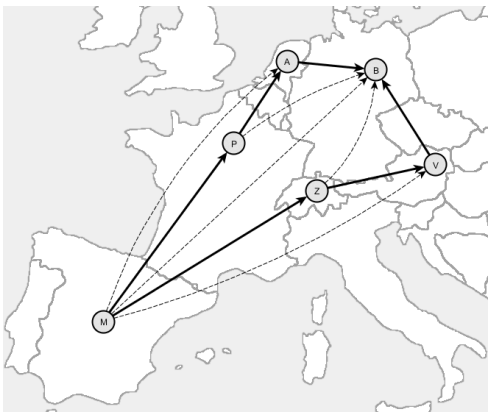

Fig. 1. Example OCL expressions showing OCL essentials avoiding objects.