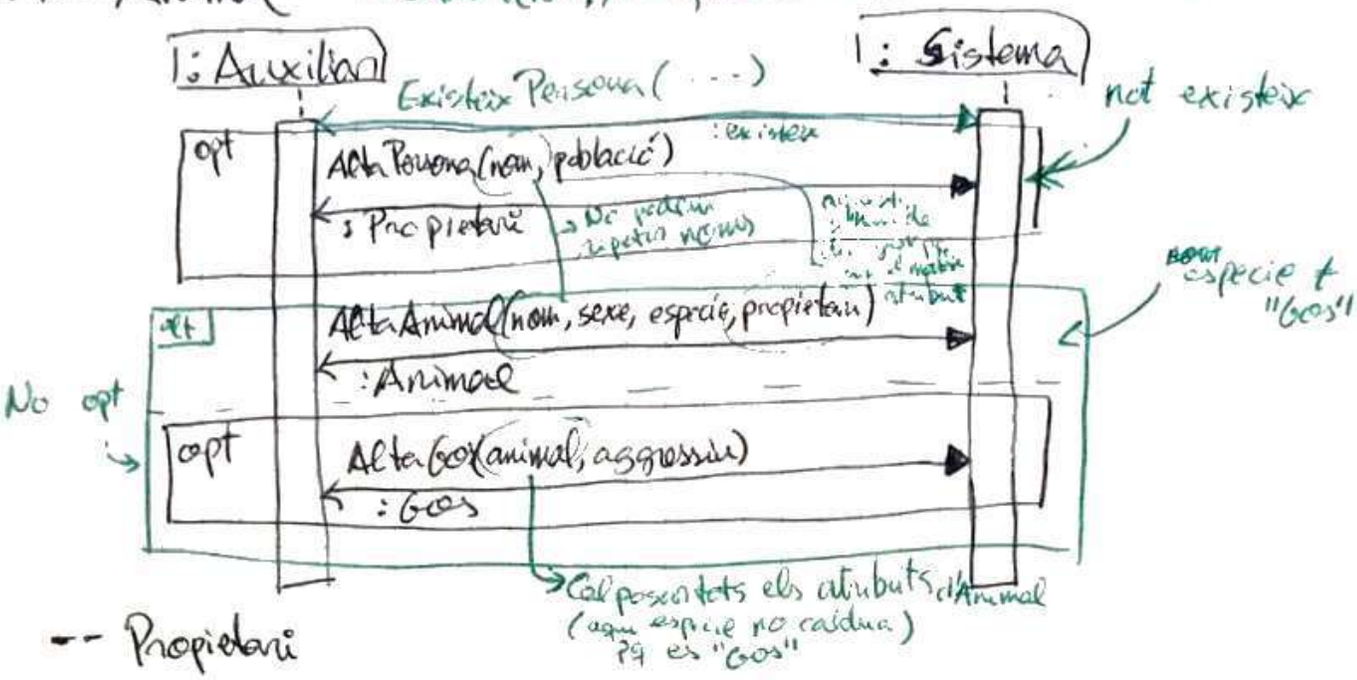
Shengye IES Problema 6

Alta Animal

Es podria fer amb una sola transacció tot això amb altaAnimal(nomP, nomA, nomE, sexe, població, agressiu)



-- Propietari

context: Sistema::altaPersona(nom:String, poblacio:String): Població
pre:
post: Propietari.allInstances()->Exists(p | p.oclIsNew() and
p.nom = nom and
p.poblacio = poblacio and
result=p)

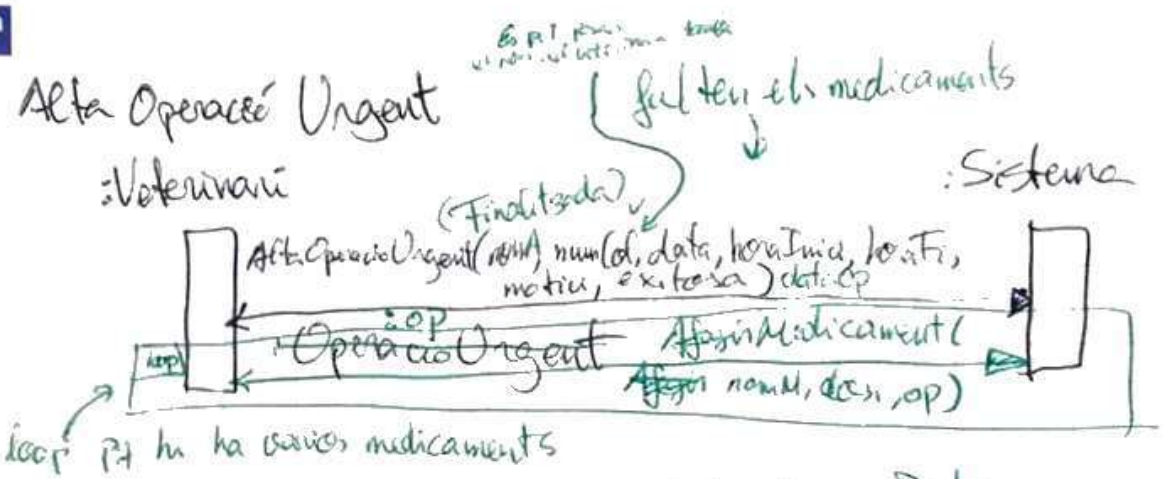
-- Animal

context: Sistema::altaAnimal(nom: String, sexe: Tsexe, especie: String, propietari: String): Animal
pre: Propietari.allInstances()->Exists(p | p=propietari)
post: Animal.allInstances()->Exists(a | a.oclIsNew() and
a.nom = nom and
a.sexe = sexe and
a.especie = especie and
a.propietari = propietari and
result = a)

-- Gos

context: Sistema::altaGos(animal: String, agressiu: Boolean): Gos
pre: Especie.allInstances()->Exists(e | e.nom = "gos" and
Animal.allInstances()->Exists(a | a = animal)
p.nom = animal.especie.nom)
post: Gos.allInstances()->Exists(g | g.animal = animal and
g.agressiu = agressiu and
g.oclIsNew() and
result=g)

Alta Operació Urgent



dates tipus Date
temps tipus Time

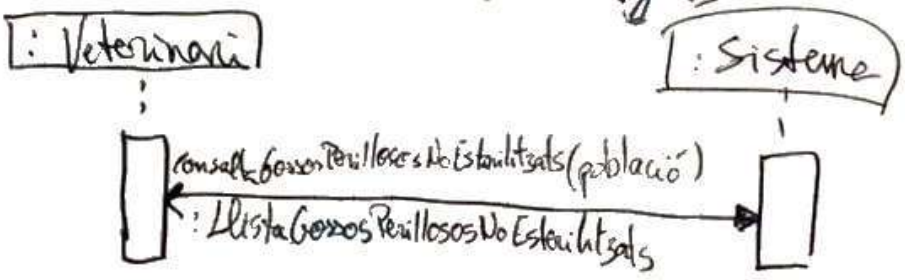
-- OperacióUrgent

context Sistema::altaOperacioUrgent(nomA:String, numCol: Integer, data:Date, horaInici: String, horaFi: String, motiu: String, exitosa: Bolean, horaFi. String): Operacio

pre: Veterinari.allInstances() -> Exists(o | o.numCol = numCol and o.especie.animal.nom -> includes(nomA))

post: Operacio.allInstances() -> Exists(o | o.oclIsNew() and o.motiu = motiu and o.numCol = numCol and o.nom = nom and exitosa = exitosa and o.horaInici = horaInici and o.horaFi = horaFi and result = o)

Consulta Gossos Perillosos no Esterilitzats



a per-t	rep	ord
Set	x	x
Seq	✓	✓
Bag	✓	x

-- LlistaGossosPerillososNoEsterilitzats

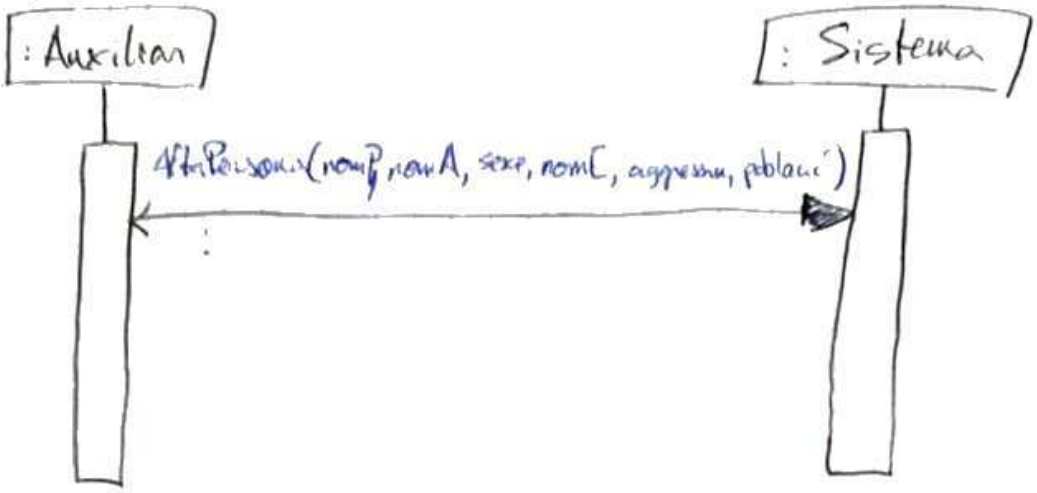
context Sistema:: ConsultaGossosPerillososNoEsterilitzats(poblacio: String): Set(TupleType(nom: String, propietari: String, veterinari: String))

pre: Persona.allInstances() -> Exists(p | p.animal -> notEmpty())

body: let gossos: Set(Gos) = Gos.allInstances() ->
select(g | g.sexe = mascle and g.agressiu = true and g.Propietari.poblacio = poblacio and g.Operacio.motiu = "Esterilització")
in result = gossos -> collect(gos | Tuple{
nom = gos.nom,
propietari = gos.Propietari.nom,
veterinari = gos.Propietari.oclIsTypeOf(Veterinari)
})

IES Problema 6 – Solució proposada

Alta Animal

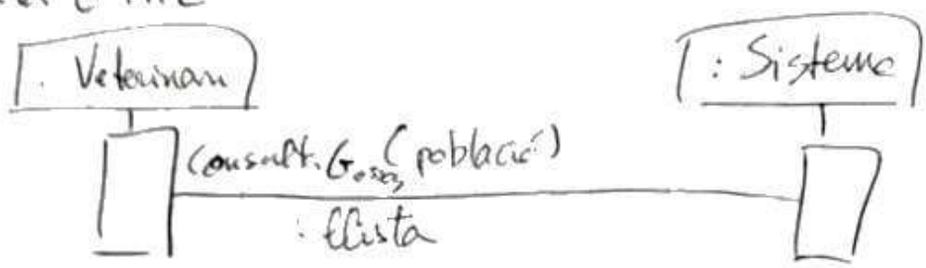


context: Sistema:: AltaAnimal(
nomP: string, poblacio: string
nomA: string, sexe: TSexe, nomE: string, agressiu: bool
)

pre: Especie.AllInstances() → exists(e | e.nom = nomE)

post: Animal.AllInstances() → exists(a |
a.nom = nomA
AND a.sexe = sexe
AND a.especie.nom = nomE
AND a.propietari.nom = nomP
AND a.oclIsNew()
AND if nomEspecie = "Gos" then
a.oclIsTypeOf(Gos)
AND a.oclAsType(Gos).agressiu = agressiu
end)
AND
if not Persona.AllInstances()@pre → exists(p1 | p1.nom = nomP) then
Persona.AllInstances() → exists(p2 |
p2.nom = nomP
AND p2.poblacio = població
AND p2.oclIsNew()
)

Consulta 6 PnE



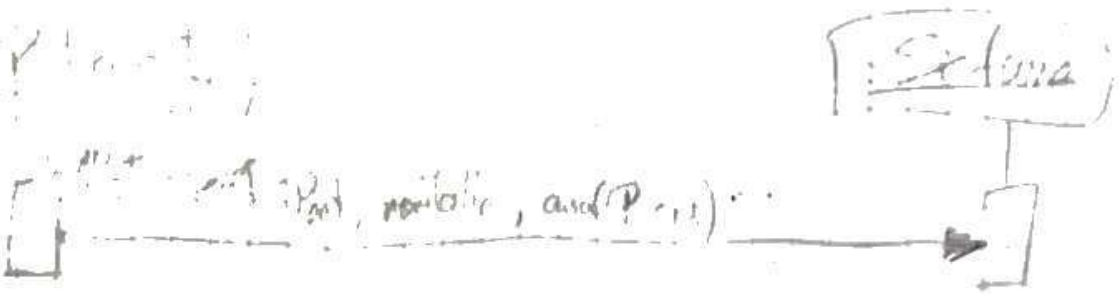

```
context: Sistema::ConsultaGossos(
    població: String)
    : Set(
        TupleType(
            nomGos: String,
            nomP: String,
            esVet: Boolean)
    )
pre: Persona.AllInstances() -> exists(p|
    p.població = població
    AND p.animal -> notEmpty()
)
body: result = Gos.AllInstances()
    -> select(g|
        g.agressiu
        AND g.sexe = TSexe::mascle
        AND g.propietari.població = població
        AND g.operacio.motiu -> excludes("esterilitzacio")
    )
    -> collect(g2| Tuple{
        nomGos = g2.nom,
        nomP = g2.propietari.nom,
        esVet = g2.propietari.oclIsTypeOf(Veterinari)
    })
```

Alta Operació Urgent

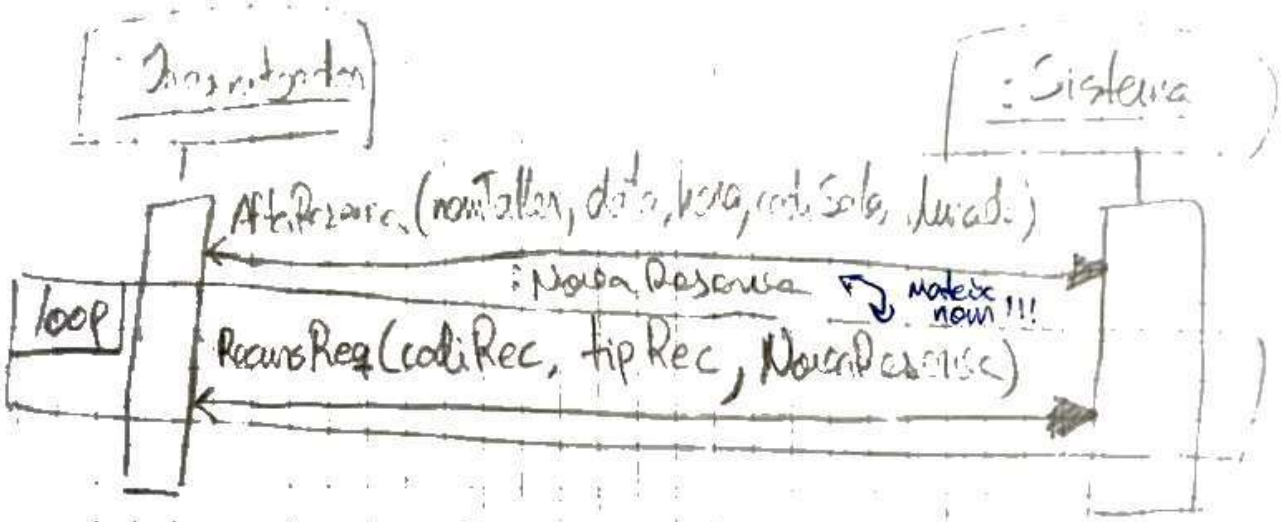
```
context: Sistema::AltaOp(...): Op
pre: Veterinari.AllInstances() -> exists(o|o.numCol = numCol
        AND o.especie.animal.nom -> includes(nomA))
    Animal.AllInstances() -> exists(a|a.nom = nomA)
post: Operacio.AllInstances() -> exists(...)
```

Meds

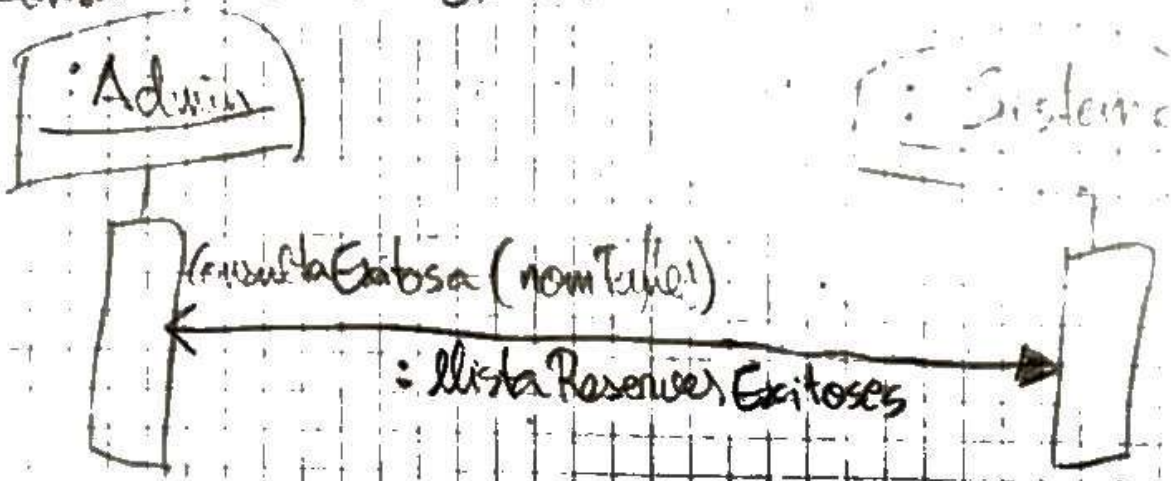
```
context: ..
pre: Medicament.AllInstances() -> exists(...)
post: ...
```



Alta Reserva



Consulta Reserves Exitoses



- Recomanacions

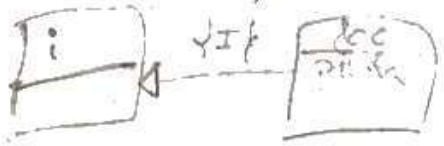
  Considerar → entrada
  → sortida
  → pre
  → condició body

- No usar count() ~~mai~~ (no ~~es~~ cal en ~~cap~~ els problemes de IES; fer servir select() → size())
- Fer servir noms de rol en navegació
  → Si no en té, és el nom de la classe en minúscula
- Si poden tenir valors repetits (no retornem una clau)
  fem →asSet()
- body:

  let ___

  in
  result= can → collect (. . .)

- Tipus d'herència:



  if i.oclIsTypeOf(foo)
  i.asType(foo). pakata // accedir a pakata

- En el context, si cal retornar posem el tipus de dada.

  Sistema::AltaReserva(___) : Reserva (si una funció)

- En una consulta no cal comprovar si el valor existeix. (Si no existeix, donaria buit i ja)
- Comprovar claus dèbils de les classes en la pre !!
  RTs, multiplicitats amb min, etc.
- No obligatori comprovar connectesa/existència de dates, hores, etc
- En crear un recurs, usar @pre
- Es poden fer servir today() i now() per comprovar dates i hores
- Herències encadenades: A → B → C . es pot accedir de A a C directament per oclIsType() o oclAsType()