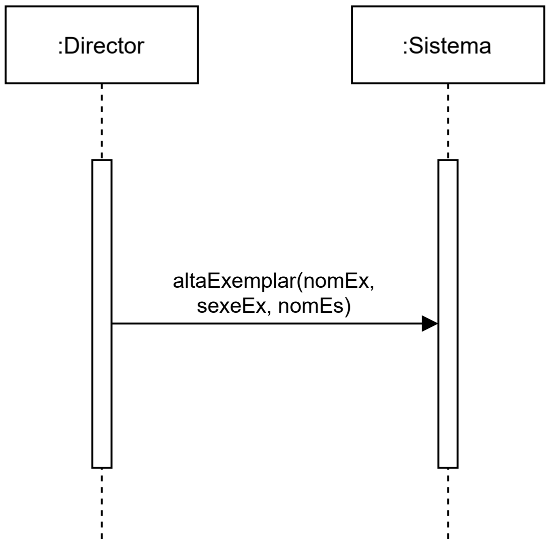
AltaExemplar



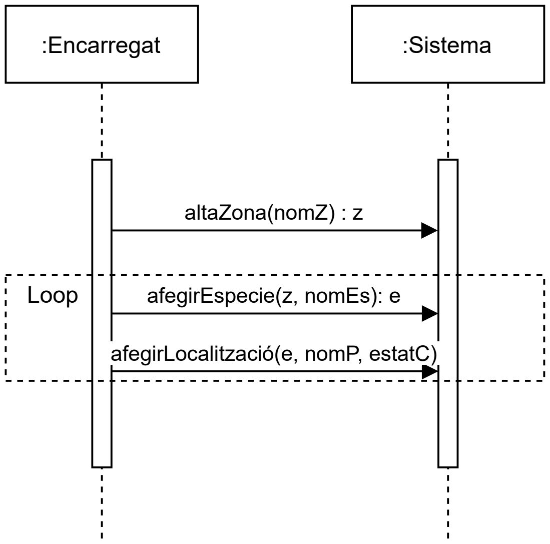
context: Sistema::altaExemplar(nomEx: String, sexeEx: TipusSexe, nomEs: String, min: Integer, max: Integer, fertil: Bool)

pre: Especie.allInstances()->exists(e | e.nom = nomEs and e.zona->exists(z | z.especie->size()>=2))

post: Exemplar.allInstances()->exists(ex | ex.oclIsNew() and ex.nom = nomEx and ex.sexe = sexeEx and ex.especie.nom = nomEs and result = ex and

```
if ex.especie.nom = "Lleó" then

    ex.oclIsTypeOf(ExemplarLleó) and ex.oclAsType(ExemplarLleó).minimKg =
    min and ex.oclAsType(ExemplarLleó).maximKg = max and

  if ex.sexe = TipusSexe.Femení then
    ex.oclIsTypeOf(ExemplarLleóFemella) and ex.oclAsType(ExemplarLleóFemella)
    .fertil = fertil
  endif
endif)
```

AltaZona


:Encarregat
:Sistema
altaZona(nomZ) : z
Loop
afegirEspecie(z, nomEs): e
afegirLocalització(e, nomP, estatC)

context: Sistema::altaZona(nomZ: String) : Zona

pre: --

post: Zona.allInstances()->exists(z | z.oclIsNew() and z.nom = nomZ and result = z)

context: Sistema::afegirEspecie(z: Zona, nomEs: String) : Especie

pre: --

post: if not Especie.allInstances()@pre->exists(e | e.nom = nomEs) then

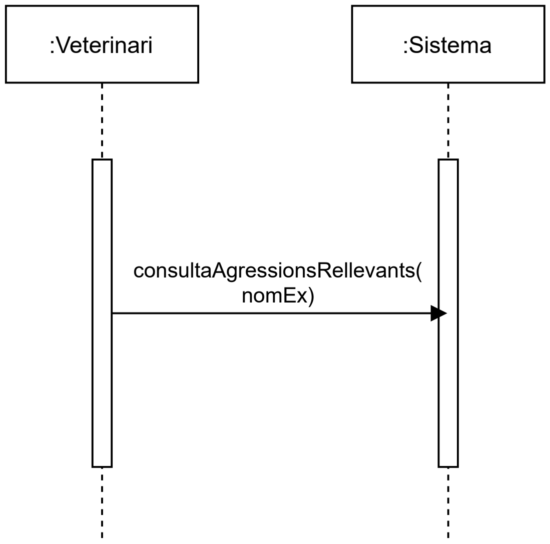
```
    Especie.allInstances()->exists(e | e.oclIsNew() and e.nom = nomEs and
    result = e)

endif and zo.especie.nom->includes(nomEs)
```

context: Sistema::afegirLocalització(e: Especie, nomP: String, estatC:TipusEstat)

pre: Pais.allInstances()->exists(p | p.nom = nomP)

post: Localitzacio.allInstances()->exists(lo | lo.oclIsNew() and lo.estatConservacio = estatC and lo.pais.nom = nomP and lo.especie = e)

ConsultaAgressionsRellevants



context: Sistema::consultaAgressionsRellevants(nomEx: String) :

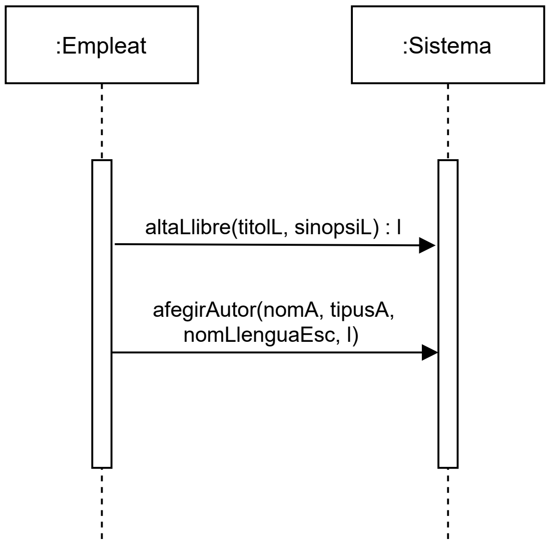
agressions: Set(TupleType(data: Data, nomExFerit: String))

pre: Exemplar.allInstances()->exists(ex | ex.nom = nomEx and not Localitzacio.allInstances()->exists(lo | ex.especie.pais.nom->includes(lo.pais.nom) and lo.especie.nom = ex.especie.nom and l.estatConservacio = TipusEstat.EnPerillGreu))

body: let agressions: Set(Agressio) = Agressio.allInstances()->select(a | a.agressor.nom = nomEx and a.agredit.especie.localitzacio->exists(lo | lo.estatConservacio = "EnPerillGreu"))

in result = agressions->collect(a | Tuple{ data = a.data.data; ferits = a.ferit.nom })

AltaLlibre



:Empleat
:Sistema
altaLlibre(titolL, sinopsiL) : l
afegirAutor(nomA, tipusA, nomLlenguaEsc, l)

context: Sistema::altaLlibre(titolL: String, sinopsiL: String) : Llibre

pre: --

post: Llibre.allInstances()->exists(l | l.oclIsNew() and l.titol = titolL and l.sinopsi = sinopsiL and result = l)

context: Sistema::afegirAutor(nomA: String, tipusA: TipusAutor, nomLlenguaEsc: String, l: Llibre)

pre: --

post: if not Persona.allInstances()@pre->exists(a | a.nom = nomA) then

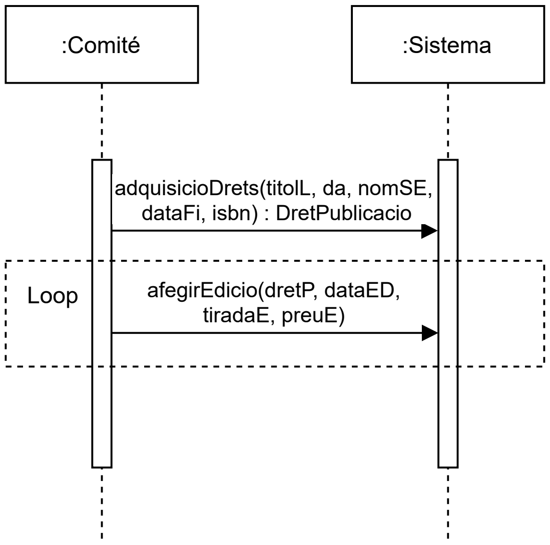
Persona.allInstances()->exists(p | p.oclIsNew() and p.nom = nomA and

p.oclIsTypeOf(Autor) and p.oclAsType(Autor).tipus = tipusA and

p.oclAsType(Autor).llengua.nom = nomLlenguaEsc and l.autor = p)

else if Persona.allInstances()->exists(p | p.nom = nomA and not
Autor.allInstances()->exists(a | a.nom = nomA) then
p.oclIsTypeOf(Autor) and p.oclAsType(Autor).llengua.nom = nomLlenguaEsc
and p.oclAsType(Autor).tipus = tipusA and l.autor = p)
endif

AdquisicióDrets



:Comité
:Sistema
adquisicioDrets(titolL, da, nomSE, dataFi, isbn) : DretPublicacio
Loop
afegirEdicio(dretP, dataED, tiradaE, preuE)

context: Sistema::adquisicioDrets(titolL: String, da: Date, nomSE: String, dataFi: Date, isbn: Integer) : DretPublicacio

pre: Llibre.allInstances() -> exists(l | l.titol = titolL) and SegellEditorial.allInstances() -> exists(se | se.nom = nomSE)

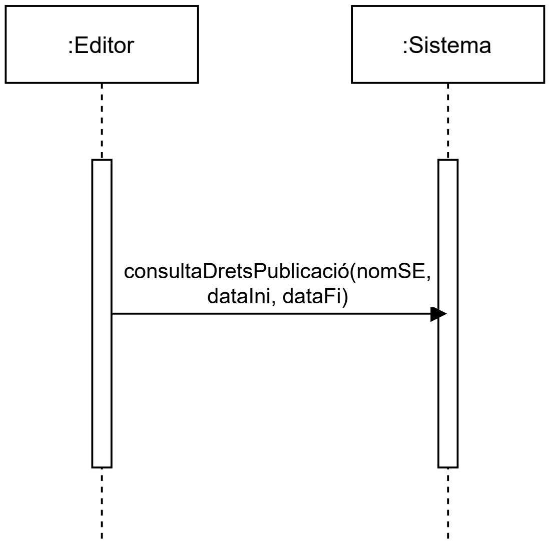
post: DretPublicacio.allInstances()->exists(dp | dp.oclIsNew() and dp.data.data = da and dp.llibre.titol = titolL and dp.segelleditorial.nom = nomSE and dp.dataFi = dataFi and dp.isbn = isbn and result = dretP)

context: Sistema::afegirEdicio(dretP: DretPublicacio, dataED: Date, tiradaE: Integer, preuE: Integer)

pre: Data.allInstances()->exists(d | d.data = dataED)

post: Edicio.allInstances()->exists(e | e.oclIsNew() and e.tirada = tiradaE and e.preu = preuE and e.dretpublicacio = dretP and e.data.data = dataED)

ConsultaDretsPublicació



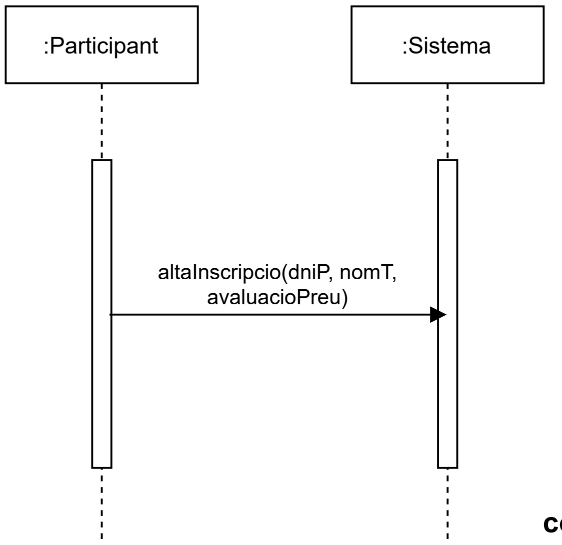
context: Sistema::consultaDretsPublicació(nomSE: String, dataIniC: Date, dataFiC: Date) : Set (TupleType(titol: String, isbn: Integer, dates: Set(Date))

pre: SegellEditorial.allInstances() -> exists(se | se.nom = nomSE) and DretPublicacio.allInstances()->exists(dp | dp.segelleditorial.nom = nomSE and dp.data.data > dataIniC and dp.data.data < dataFiC and dp.llibre->size()>10)

body: let drets: Set(DretPublicacio) = DretPublicacio.allInstances() -> select(dp | dp.data.data > dataIniC and dp.data.data < dataFiC and dp.edicio->size()>5)

in drets->collect(dp | Tuple{titol=dp.llibre.titol; isbn=dp.isbn; dates=dp.datesEdicions.data})

AltaInscripcio



context: Sistema::altaInscripcio(dniP: String, nomT: String, avaluacioPreu: TPreu)

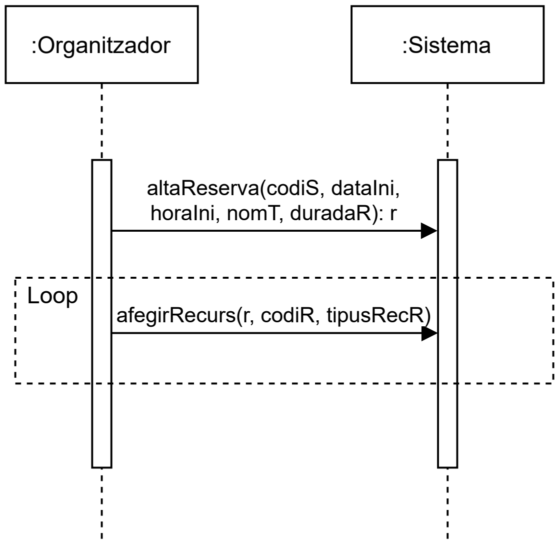
pre: Taller.allInstances()->exists(t | t.nom = nomT) and Persona.allInstances()->exists(p | p.dni = dniP)

post: Inscripcio.allInstances()->exists(i | i.oclIsNew() and i.taller.nom = nomT and i.participant.dni and

```
if i.taller.tipusTaller = TallerDePagament then
    i.oclIsTypeOf(InscripcioDePagament) and
    i.oclAsType(InscripcioDePagament).pagada = false and
    i.oclAsType(InscripcioDePagament).avaluacioPreu = avaluacioPreu
endif)
```

context: Sistema::altaReserva(codiS: String, dataIni: Date, horaIni: Hora, nomT: String, duradaR: Integer): Reserva

AltaReserva


:Organitzador
:Sistema
altaReserva(codiS, dataIni, horaIni, nomT, duradaR): r
Loop
afegirRecurs(r, codiR, tipusRecR)

pre: Sala.allInstances()->exists(s | s.codi = codiS) and

Taller.allInstances()->exists(t | t.nom = nomT) and

DataHora.allInstances()->exists(d | d.data = dataIni and d.hora = horaIni)

post: Reserva.allInstances()->exists(r | r.oclIsNew() and r.sala.codi = codiS and

r.datahora.data = dataIni and r.datahora.hora = horaIni and

r.taller.nom = nomT and r.durada = duradaR and result = r)

context: Sistema::afegirRecurs(r: Reserva, codiR: String, tipusRecR: String)

post: if not Recurs.allInstances()@pre->exists(rec | rec.codi = codiR) then

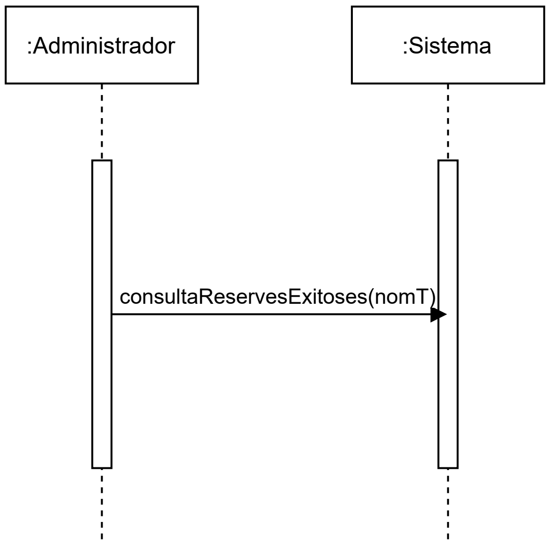
Recurs.allInstances()->exists(rec | rec.oclIsNew() and rec.codi = codiR and

rec.tipusRec = tipusRecR and r.recurs = rec)

else Recurs.allInstances()->exists(rec | r.recurs = rec)

endif)

ConsultaReservesExitoses



context: Sistema::consultaReservesExitoses(nomT: String) : Set (TupleType(codi: String, data: Date, hora: Hora, correuEPartic: Set(String))

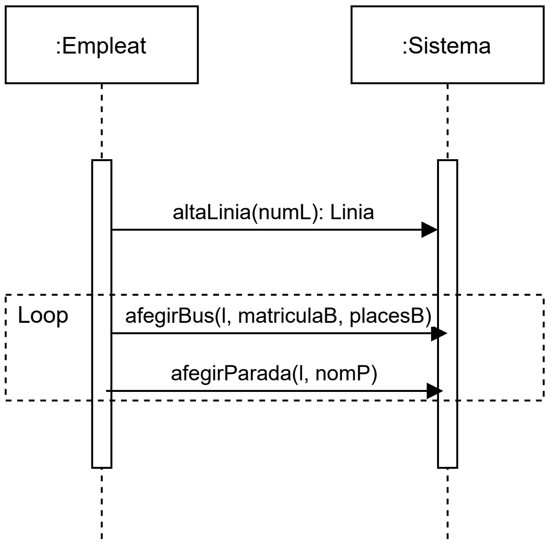
pre: taller.allInstances() -> exists(t | t.nom = nomT and
t.oclIsTypeOf(TalleDePagament)

body: let reserves: Set(Reserva) = Reserva.allInstances()->

select(r | r.valoracio-> select(v | v.puntuacio = 5)->size() > 5)

in result = reserves->collect(r | Tuple{codi = r.sala.codi; data = r.datahora.data;
hora = r.datahora.hora; correuEPartic = r.taller.participant.correuE})

AltaLínia



context: Sistema::altaLinia(numL: String): Linia

pre: --

post: Linia.allInstances()->exists(l | l.oclIsNew() and l.numero = numL and resultat = l)

context: Sistema::afegirBus(l: Linia, matriculaB: String, placesB: Integer)

pre: Bus.allInstances()->exists(b | b.matricula = matriculaB and b.places = placesB)

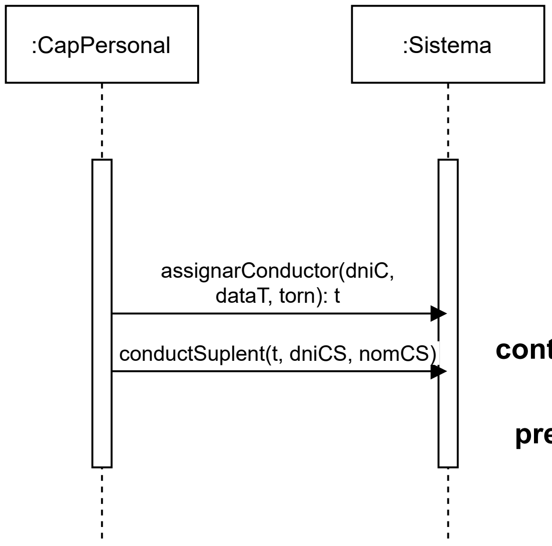
post: Bus.allInstances()->exists(b | b.matricula = matriculaB and b.places = placesB and l.bus = b)

context: Sistema::afegirParada(l: Linia, nomP: String)

pre: Parada.allInstances()->exists(p | p.nom = nomP)

post: Parada.allInstances()->exists(p | p.nom = nomP and l.parada = p)

AssignarConductor



:CapPersonal
:Sistema
assignarConductor(dniC, dataT, torn): t
conductSuplent(t, dniCS, nomCS)

context: Sistema::assignarConductor(dniC: String, dataT: Data, torn: TipusTorn) : TornTreballat

pre: Conductor.allInstances()->exists(c | c.dni = dniC)

post: TornTreballat.allInstances()->exists(tt | tt.oclIsNew() and tt.data.data = dataT and tt.conductor.dni = dniC and tt.torn.torn = TipusTorn and

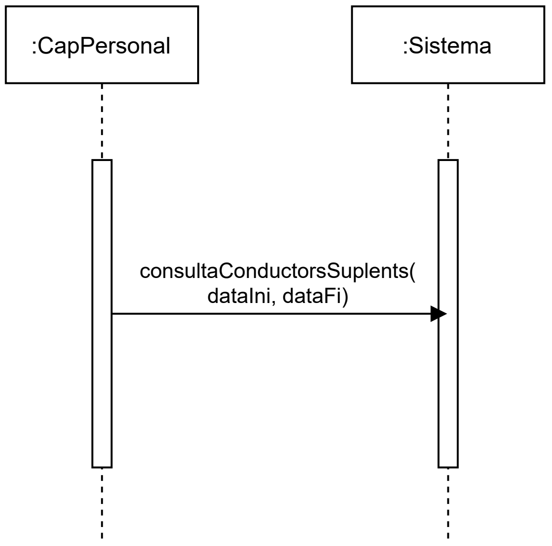
if tt.torn.torn = "TornNocturn" then

context: Sistema::conductSuplent(t: TornTreballat, dniCS: String, nomCS: String)

pre: TornTreballat.allInstances()->exists(tt | tt.data = t.data and tt.conductor = t.conductor and tt.torn = t.torn and tt.IsTypeOf(TornNocturn))

post: if not Conductor.allInstances()->exists(c | c.suplent.tornnocturn= t.oclAsType(TornNocturn) then Conductor.allInstances()->exists(cs | cs.oclIsNew() and cs.dni = dniCS and cs.nom = nomCS and cs.suplent.tornnocturn= t)

ConsultaConductorsSuplents



context: Sistema::consultaConductorsSuplents(dataIni: Date, dataFi: Date) : Set (TupleType(dni: String, linies: Set(Integer))

pre: Conductor.allInstances()->exists(c | c.suplent.tornnocturn->size()>0 and not TornTreballar.allInstances()->exists(tt | tt.conductor.dni = c.dni and tt.oclIsTypeOf(TornNocturn))

body: let conductors: Set(Conductor) = Conductor.allInstances()->select(c | TornTreballat.allInstances()->select(tt | tt.torn.torn != "TornNocturn" and tt.data.data > dataIni and tt.data.data < dataFi and tt.conductor.dni = c.dni)->size()>1)

in conductors->collect(c | Tuple{dni = c.dni; linies=c.linia.numero})