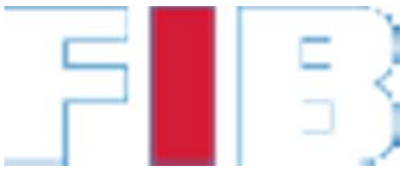
Introducció a l'Enginyeria del Software (IES)

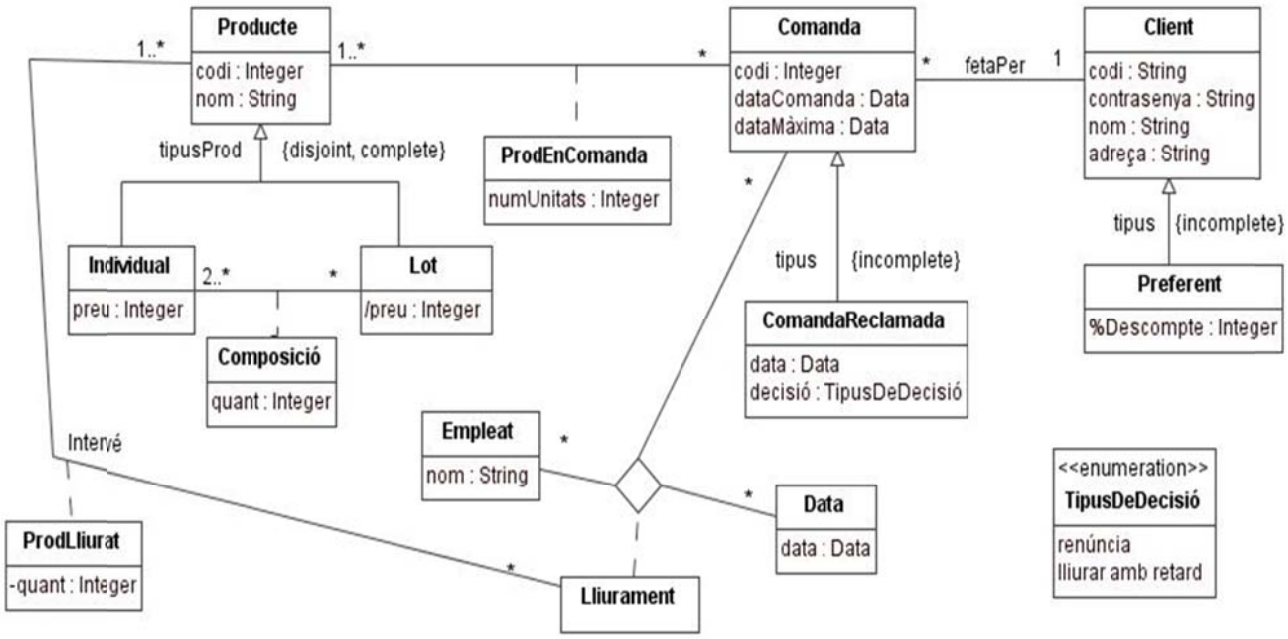
Curs 2013/14

Exercicis resolts de l'Esquema del Comportament



Esquema del Comportament en UML

1. Una empresa de distribució necessita un sistema d'informació per gestionar les comandes que serveix als seus clients. Una comanda la fa un determinat client i està composada per una sèrie de productes. Un producte pot ser individual o bé un lot de productes individuals. De cada comanda se'n poden fer diversos lliuraments d'una part dels productes que hi són continguts. La part rellevant de l'esquema conceptual de dades d'aquest sistema es mostra a la figura següent:



Restriccions Textuals:

1. Claus externes de classes no associatives:
   (Producte, codi); (Comanda, codi); (Client, codi i contrasenya); (Data, data); (Empleat, nom)
2. La data màxima d'una comanda ha de ser posterior o igual a la data de la comanda.
3. Un client no pot fer més de 5 comandes en una mateixa data.
4. La data de lliurament d'una comanda ha de ser posterior o igual a la data de la comanda.
5. Els productes d'un lliurament d'una comanda pertanyen a la seva comanda.
6. La suma de les quantitats dels productes lliurats d'una comanda no és superior a la suma de les quantitats dels productes demanats en la comanda.
7. La data de reclamació d'una comanda ha de ser posterior a la data màxima de la comanda.

Informació derivada:

1. /preu d'un lot és la suma de: preu × quantitat dels productes individuals que composen el lot.

El sistema a desenvolupar no ha de donar d'alta instàncies de Client, Empleat ni Data atès que hi ha un altre sistema encarregat de fer-ho. Ha de proporcionar les funcionalitats següents: crear lliurament (CL), crear producte (CP) i consultar comandes reclamades (CC).

CL. Quan un usuari del sistema decideix fer un nou lliurament indica totes les dades necessàries per identificar el lliurament i tota la informació necessària dels productes que conté. Un lliurament només es pot fer si tots els productes que conté ja existeixen en el moment de fer el lliurament. No es pot fer el lliurament si la comanda a la qual fa referència ja ha estat reclamada i el client hi ha

renunciat. Feu que la interacció necessària per a portar a terme aquesta funcionalitat requereixi més d'un esdeveniment.

CP. Quan un usuari del sistema vol crear un producte ha d'indicar el codi del producte i tota la informació necessària per a donar-lo d'alta. Si el producte que es crea és un lot, aleshores també caldrà indicar tota la informació dels productes individuals que el composen. Si un producte individual d'un lot no existeix en el moment de crear el lot, aleshores també caldrà crear aquest producte individual.

CCR. Quan un usuari del sistema vol consultar les comandes reclamades d'un client indica al sistema el codi del client. Aleshores el sistema mostra per cada comanda reclamada del client que té algun producte del qual no se n'ha fet cap lliurament el codi de la comanda i la data de la reclamació.

Es demana:
– *Esquema del Comportament del Sistema*: Diagrames de seqüència del sistema de totes les funcionalitats especificades i contractes en OCL de totes les operacions que hi apareixen.

Solució:

Crear lliurament

context: `Sistema::crearLliurament(nomE: String, da: Data, codiC: Integer): Lliurament`

pre:[1] `Comanda.allInstances()->exists(c|c.codi = codiC and not c.oclAsType(ComandaReclamada).decisió = "renúncia") and Empleat.allInstances()->exists(e|e.nom = nomE)`

post: `Lliurament.allInstances()->exists(l|l.oclIsNew() and l.data.data = da and l.empleat.nom = nomE and l.comanda.codi = codiC and result=l)`

context: `Sistema::afegirProd(l: Lliurament, codiP: Integer, q:Integer)`

pre: `Producte.allInstances()->exists(p|p.codi = codiP)`

post: `ProdLliurat.allInstances()->exists(pl| pl.oclIsNew()and pl.quant=q and pl.lliurament=l and pl.producte.codi=codiP)`

Crear producte

context: `Sistema::crearProd(codi: Integer, nom:String, tipus:String, preu:Integer): Producte`

post: `Producte.allInstances()->exists(p| p.oclIsNew() and p.codi=codi and p.nom=nom and if tipus="Lot" then p.oclIsTypeOf(Lot) else p.oclIsTypeOf(Individual) and p.oclAsType(Individual).preu=preu endif and result=p)`

[1] No cal comprovar que la data da existeix perquè suposem que les dates existeixen totes.

context:
```
Sistema::afegirComposició(p: Producte, codi:Integer,
q:Integer, nom:String, preu:Integer)
```

pre:
```
p.oclIsTypeOf(Lot)
```

post:
```
if not Individual.allInstances()@pre -> exists(i|i.codi =
codi) then
Individual.allInstances()->exists(i| i.oclIsNew()and
i.nom=nom and i.preu=preu) endif
and Composició.allInstances()->exists(c| c.oclIsNew() and
c.quant=q and c.individual.codi=codi and c.lot=p)
```

Consultar comandes reclamades

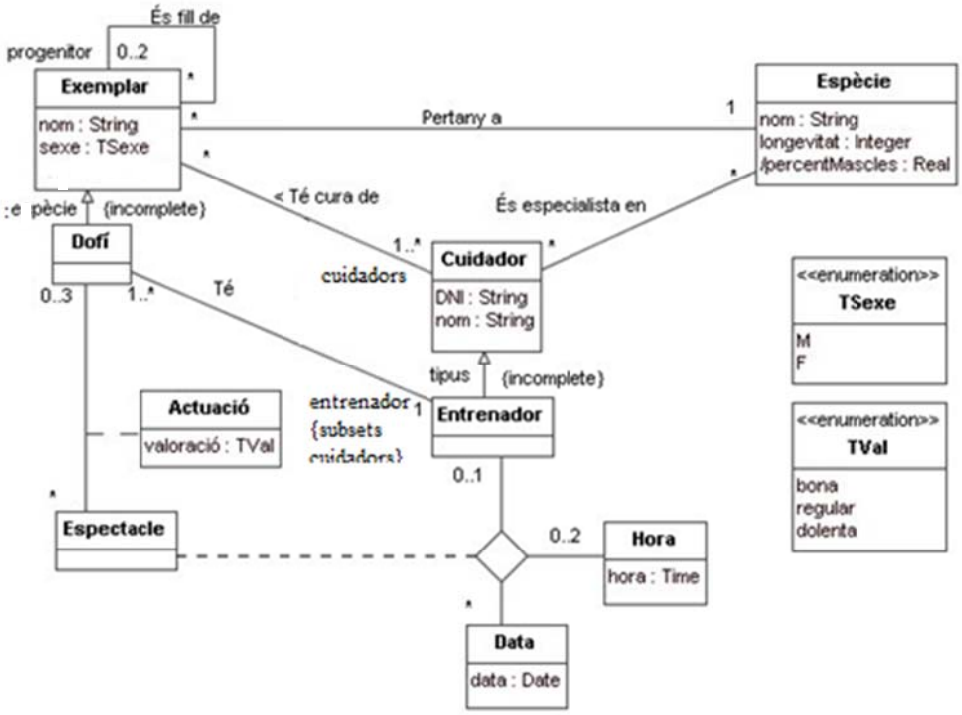
context:
```
Sistema::consultarComanda(codi: String)
:Set(TupleType(codi: Integer, da: Data))
```

body:
```
let comandes: Set(ComandaReclamada) =
ComandaReclamada.allInstances() -> select (c |
c.client.codi=codi and c.producte -> exists(p |
p.lliurament ->isEmpty()))
in
result = comandes-> collect(c | Tuple{codi = c.codi, data
= c.da})
```

2. Un zoològic ens ha demanat que li dissenyem una part del seu sistema d'informació. Els exemplars s'identifiquen pel seu nom i se'n coneix el sexe i l'espècie a la que pertanyen. De cada espècie s'emmagatzema el nom i la longevitat. Els cuidadors s'identifiquen pel seu DNI i en sabem també el nom. Cada cuidador és especialista en un conjunt d'espècies i té cura d'un conjunt d'exemplars. Els dofins tenen, a més, un entrenador que és un dels seus cuidadors. Els dofins participen en espectacles, que s'organitzen en una data i una hora, i són dirigits per un entrenador, que no ha de ser necessàriament el seu. Els dofins obtenen una valoració (bona, regular o dolenta) per cada actuació que fan. Un fragment de l'esquema conceptual en UML d'aquest sistema és:

<u>Restriccions d'integritat textuals</u>
- Claus classes no associatives: (Espècie, nom), (Exemplar, nom), (Cuidador, dni), (Hora, hora), (Data, data)
- L'associació 'És fill de' no té cicles
- Si un exemplar té dos progenitors, aquests són de sexe diferent
- Els progenitors d'un exemplar són de la mateixa espècie que l'exemplar fill
- No pot ser que un exemplar de tipus Dofí no pertanyi a l'espècie "dofí"
- Un cuidador no pot tenir cura d'exemplars d'una espècie en la que no és especialista
- Un dofí no pot actuar en dos espectacles en la mateixa data i hora

<u>Informació derivada</u>
- */espècie* correspon al nom de l'espècie a la qual pertany l'exemplar
- */percentMascles* és el percentatge d'exemplars mascles que té l'espècie al zoo



El sistema a desenvolupar no ha de donar d'alta Espècies, Cuidadors, Dates, Hores ni Espectacles, atès que hi ha un altre sistema encarregat de fer-ho. No obstant, ha de proporcionar les funcionalitats següents: *Introduir un exemplar*, *Valorar els dofins d'un espectacle* i *Llistar els dofins amb valoracions bones*.

Quan un usuari vol introduir un exemplar, indica al sistema el seu nom, el sexe, l'espècie, els DNIs dels seus cuidadors i els noms dels seus progenitors. A més, si és un dofí indicarà el DNI del seu entrenador. Feu que la interacció necessària per a portar a terme aquesta funcionalitat requereixi un únic esdeveniment.

Quan l'entrenador que ha dirigit un espectacle vol valorar tots els dofins que hi han participat, indica al sistema totes les dades necessàries per introduir l'espectacle i les valoracions dels dofins (tingueu en compte que en aquest cas d'ús no s'han de crear ni els espectacles ni els dofins). Feu que la interacció necessària per a aquesta funcionalitat requereixi més d'un esdeveniment.

Quan un usuari demana el llistat de dofins amb valoracions bones, el sistema li retorna, per cada dofí amb més de 3 valoracions bones en espectacles diferents, el seu nom i el nom dels seus progenitors. Aquest llistat només s'emetrà si tots els dofins han estat dirigits com a mínim per 2 entrenadors diferents en els seus espectacles.

Es demana:

– *Esquema del Comportament del Sistema*: Diagrames de seqüència del sistema de totes les funcionalitats especificades i contractes en OCL de totes les operacions que hi apareixen.

Solució:

Introduir exemplar

context: `Sistema::introduirExemplar(nom: String, sexe: TSexe, espècie: String, cuidadors: Set(String), pares: Set(String), entrenador: String)`

pre: `Espècie.allInstances()->exists(e|e.nom=espècie) and Cuidador.allInstances().dni -> includesAll(cuidadors) and Exemplar.allInstances().nom -> includesAll(pares) and if espècie = 'Dofí' then Entrenador.allInstances()-> exists(e|e.dni=entrenador) endif`

post: `Exemplar.allInstances()->exists(e| e.oclIsNew() and if espècie = 'Dofí' then e.oclIsTypeOf(Dofí) and e.entrenador.dni = entrenador endif and e.nom = nom and e.sexe = sexe and e.espècie.nom = espècie and e.cuidador.dni-> includesAll(cuidadors) and e.progenitor.nom-> includesAll(pares))`

Valorar dofins

context: `Sistema:obtenirEspectacle(data: Date, hora: Time): Espectacle`

pre: `Espectacle.allInstances()->exists(e | e.data.data = data and e.hora.hora = hora)`

body: `result=Espectacle.allInstances()->select(e | e.data.data = data and e.hora.hora = hora)`

context: `valorarDofí(e: Espectacle, nomDofí: String, valoració: TVal)`

pre: `Dofí.allInstances()->exists(d|d.nom = nomDofí)`

post: `Actuació.allInstances()->exists(a| a.oclIsNew() and a.espectacle = e and a.dofí.nom = nomDofí and a.valoració = valoració)`

Llistar dofins

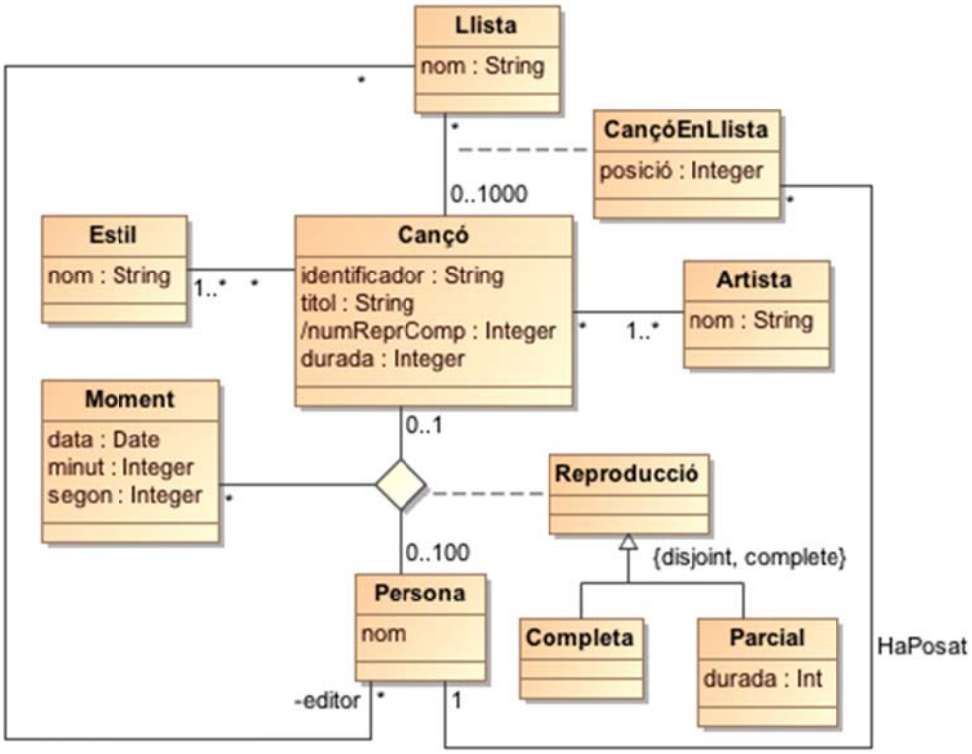
context: `Sistema::llistarDofins(): Set(TupleType(nom: String, progenitors: Set(String))`

pre: `Dofí.allInstances()->forall(d|d.espectacle.entrenador-> asSet()-> size()>=2)`

body:

```
let dofins: Set(Dofí) =
Dofí.allInstances()->select(d | d.actuació-> select(a | a.valoració = 'bona')->size() >3)
in
result = dofins-> collect(d |  Tuple {nom = d.nom, progenitors = d.progenitor.nom })
```

3. Una empresa que es dedica a la reproducció de música per Internet ens ha demanat que els desenvolupem un sistema que emmagatzemi les llistes de reproducció dels seus clients i que porti un seguiment de la música que escolten per tal de poder fer recomanacions. L'esquema conceptual de partida és el següent:



Restriccions textuals

1- Claus externes: (Persona, nom), (Llista, nom), (Artista, nom), (Estil, nom), (Cançó, identificador), (Moment, data+minut+segon).
2- Una mateixa persona no pot afegir més de 100 cançons a una llista.
3- Només els editors de la llista poden afegir cançons a la llista.
4- Donada una llista, no hi poden haver dues instàncies de CançoEnLlista amb la mateixa posició.
5- La durada d'una reproducció parcial ha de ser inferior a la durada de la cançó reproduïda.

Regles de derivació

1- /numReprComp d'una cançó és el nombre de vegades que una cançó ha estat reproduïda de manera completa.

El sistema a desenvolupar no ha de donar d'alta Cançons, ni Moments, ni Estils, ni Artistes, ni Persones. Ha de proporcionar les funcionalitats següents: Alta de reproducció (AR), Crear llista (CL) i Recomanar cançons (RC).

AR: Quan una persona ha escoltat una cançó ha d'indicar al sistema les dades necessàries per identificar la persona, el moment i la cançó així com si ha estat una reproducció completa o parcial. En aquest últim cas, haurà d'indicar també la durada de la reproducció.

CL: Quan una persona vol crear una llista indica el seu nom (serà el primer editor de la llista) així com el nom de la llista. A més, per cada cançó que vulgui afegir a la llista, indicarà el seu

identificador així com la posició dins la llista (cal que el sistema registri qui ha posat la cançó a la llista mitjançant l'associació HaPosat). Feu que la interacció necessària per a portar a terme aquesta funcionalitat requereixi més d'un esdeveniment.

RC: Quan una persona vol que el sistema li recomani cançons, indicarà el seu nom i el sistema li retornarà, per cada artista del qual hagi reproduït almenys una cançó de manera completa, l'identificador i el títol de totes les cançons de l'artista.

Es demana:

– *Esquema del Comportament del Sistema*: Diagrames de seqüència del sistema de totes les funcionalitats especificades i contractes en OCL de totes les operacions que hi apareixen.

Solució:

Alta de reproducció

context: `Sistema::altaReproducció(nomP:String, idC:String, data:Date, minut:Integer, segon:Integer, completa:boolean, durada:int)`

pre:
```
Persona.allInstances()->exists(p|p.nom = nomP)
and Cançó.allInstances()->exists(c|c.identificador = nomC)
and Moment.allInstances()->exists(m|m.data=data and
                          m.minut=minut and m.segon=segon)
```

post:
```
Reproducció.allInstances()-> exists(r|r.oclIsNew() and if
completa then r.oclIsTypeOf(Completa) else
r.oclIsTypeOf(Parcial) and
r.oclAsType(Parcial).durada=durada endif
and r.persona.nom = nomP and r.cançó.identificador=idC and
r.moment.data=data and r.moment.minut=minut and
r.moment.segon=segon)
```

Crear llista

context: `Sistema::crearLlista(nomP:String, nomL:String) :Tuple(persona:Persona, llista:Llista)`

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

post:
```
Llista.allInstances()->exists(l| l.oclIsNew() and l.nom =
nomL and and l.editor.nom=nomP and result.llista=l and
result.persona = Persona.allInstances()->select(p |
p.nom=nomP))
```

context: `Sistema::afegirCançóALlista(p:Persona, l:Llista, idC:String, pos:Integer)`

pre:
```
Cançó.allInstances()->exists(c|c.identificador = idC)
```

post:
```
CançóEnLlista.allInstances()->exists(c|c.oclIsNew() and
c.llista=l and c.cançó.identificador=idC and c.posicio=pos
and c.persona=p)
```

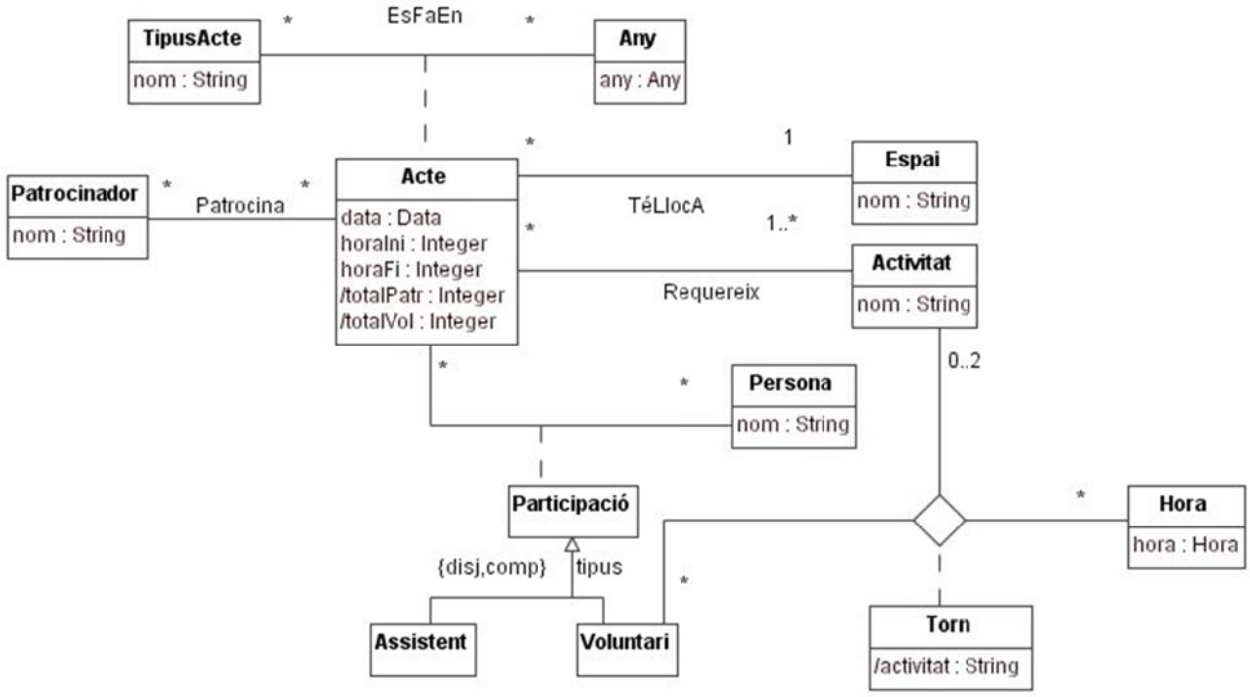
Recomanar cançons

context: `Sistema::recomanarCançons(nomP:String): Set(TupleType(id:String, titol:String)`

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

body:
```
let cançons:Set(Cançó)= Persona.allInstances()->
select(p|p.nom=nomP).reproducció->select (r|
r.oclIsTypeOf(completa)).cançó.artista.cançó
in result=cançons->collect(c|Tuple{id= c.identificador,
titol = c.titol})
```

4. L'ajuntament d'una localitat necessita un sistema d'informació per gestionar els actes de la seva festa major. Els actes són d'un determinat tipus, d'un determinat any, es fan en un espai i una data, tenen una hora d'inici i de final, poden tenir diversos patrocinadors i requereixen una o més activitats. Les persones poden participar-hi com a voluntaris o com a assistents. Els voluntaris d'un acte fan torns en hores i activitats de l'acte.



Restriccions textuals

1. Claus externes: (TipusActe, nom), (Any, any), (Patrocinador, nom), (Espai, nom), (Activitat, nom), (Hora, hora), (Persona, nom).
2. L'hora d'inici d'un acte és anterior a la seva hora de fi.
3. La data d'un acte pertany a l'any de l'acte.
4. L'activitat corresponent a un torn d'un voluntari és una de les que requereix l'acte on participa el voluntari.
5. En un espai no hi pot haver actes simultanis.
6. Una persona no pot participar en dos actes alhora.
7. L'hora d'un torn està entre l'hora d'inici i de fi de l'acte corresponent al torn.

Informació derivada

1. /totalPatr d'un acte és el nombre de patrocinadors de l'acte.

2. /totalVol d’un acte és el nombre de voluntaris de l’acte.
3. /activitat d’un torn és el nom de l’activitat del torn.

El sistema a desenvolupar no ha de donar d'alta instàncies de TipusActe, Any, Patrocinador, Espai, Activitat i Hora atès que hi ha un altre sistema encarregat de fer-ho. Ha de proporcionar les funcionalitats següents: crear un nou acte, afegir participant i consultar persones.

Quan un usuari vol crear un acte, indica al sistema el tipus d’acte, l’any, l’espai, la data, l’hora d’inici, l’hora de fi, els noms de totes les activitats que requereix i els noms de tots els patrocinadors. Feu que la interacció necessària per portar a terme aquesta funcionalitat requereixi un sol esdeveniment. Considerarem que només es poden crear actes d’anys posteriors a l’actual.

Per afegir una persona com a participant a un acte de l’any actual, l’usuari indica al sistema el nom de la persona, el tipus d’acte i si la participació serà com a assistent o com a voluntari. Si la persona no existia al sistema, també s’haurà de donar d’alta. Si la participació és com a voluntari, indica també, per a cada torn que la persona farà, l’hora i el nom de l’activitat. Feu que la interacció necessària per portar a terme aquesta funcionalitat requereixi més d'un esdeveniment. Cal tenir en compte que el sistema ha d’afegir la participació a l’acte que correspongui de l’any actual i que aquesta funcionalitat només es pot executar si l’acte té una data posterior a la data actual.

Quan un usuari vol consultar persones, indica al sistema un any i el sistema mostra totes les persones que han estat voluntàries en algun acte de l’any. Concretament, el sistema mostra el nom de la persona i, per cada acte en que ha participat, el tipus d’acte i el nom de l’espai on s’ha fet.

Es demana:
– *Esquema del Comportament del Sistema*: Diagrames de seqüència del sistema de totes les funcionalitats especificades i contractes en OCL de totes les operacions que hi apareixen.

Solució:

Crear acte

context:
```
Sistema::nouActe(ta:String, an:Any, ne:String,
da:Data, hi:Integer, hf:Integer, acts:Set(String),
patrs:Set(String))
```

pre:
```
an > currentyear()
and Espai.allInstances()->exists(e|e.nom=ne) and
Activitat.allInstances().nom->includesAll(acts) and
Patrocinador.allInstances().nom->includesAll(patrs)
and TipusActe.allInstances()->exists(ta|ta.nom=ta)
```

post:
```
Acte.allInstances()->exists(a|a.oclIsNew() and and
a.tipusActe.nom=ta and a.any.any=an and a.espai.nom=ne
and a.data=da and a.horaIni=hi and a.horaFi=hf and
a.activitat.nom->includesAll(acts) and
a.patrociandor.nom->includesAll(patrs))
```

Afegir participant

context:
```
Sistema::afegirPart(np:String, ta:String,
vol:Boolean):Participació
```
pre:
```
Acte.allInstances()->exists(a|a.tipusActe.nom=ta and
a.any.any=currentyear() and a.data > today())
```
post:
```
if not Persona.allInstances()@pre ->
exists(pr|pr.nom=np) then Persona.allInstances()->
exists(per|per.oclIsNew()and per.nom=np) endif
and Participació.allInstances()->exists(p|
p.oclIsNew() and p.persona.nom=np and
p.acte.tipusActe.nom=ta and p.acte.any.any
=currentyear() and
if vol then p.oclIsTypeOf(Voluntari) else
p.oclIsTypeOf(Assistent) endif and result=p)
```

context:
```
afegirTorn(p:Participació, ho:Hora, na:String)
```
pre:[2]
```
Activitat.allInstances()->exists(a|a.nom=na)
and p.oclIsTypeOf(Voluntari)
```
post:
```
Torn.allInstances()->exists(t|t.oclIsNew() and
t.voluntari=p and t.hora.hora=ho and
t.activitat.nom=na
```

Consultar persones

context:
```
Sistema::consultaPers(an:Any): Set(TupleType
(np:String, infActes:Set(TupleType(ta:String,
ne:String))))
```
pre:
```
Any.allInstances()->exists(a|a.any.any=an)
```
body:
```
let pers:Set(Persona) = Persona.allInstances()->
select(p|p.participació.oclAsType(Voluntari)->
exists(v|v.acte.any.any=an))
in
result = pers->collect(p| Tuple{np=p.nom,
infActes=p.acte-> collect(a|
  Tuple{ta=a.tipusActe.nom, ne=a.espai.nom})})
```

[2] No cal comprovar que l'hora *ho* existeix perquè suposem que les hores existeixen totes