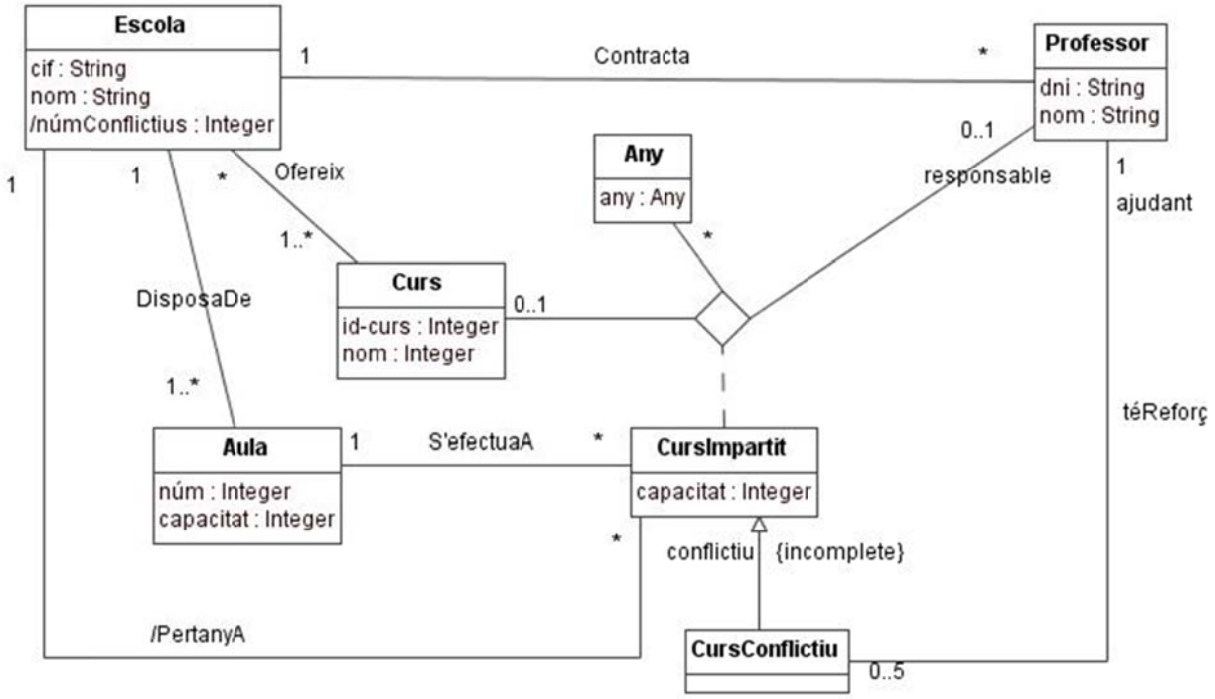
Introducció a l'Enginyeria del Software (IES)

Exercicis resolts de Domain Model: Traducció de l'Esquema d'Especificació al de Disseny

1. L'ajuntament d'una població necessita un sistema d'informació per gestionar les seves escoles i els cursos que aquestes ofereixen. Un curs ofert per una escola s'imparteix com a màxim una vegada a l'any, per part d'un professor responsable. Un professor imparteix com a màxim un curs anualment. Si un curs és conflictiu, aleshores té el reforç d'un altre professor diferent al que l'imparteix. La part rellevant de l'esquema conceptual de dades d'aquest sistema es mostra a la figura següent:



Restriccions textuals

1. Claus externes: (Escola, cif), (Professor, dni), (Curs, id-curs) , (Any, any).
2. Una escola no pot disposar de dues aules amb el mateix número.
3. En un curs impartit, l'escola de la seva aula i la del seu professor coincideixen.
4. La capacitat d'un curs impartit no és superior a la capacitat de l'aula a la que s'efectua el curs.
5. El professor responsable d'un curs conflictiu no pot ser alhora l'ajudant d'aquest curs.
6. Un professor només pot ser ajudant d'un curs conflictiu si no és responsable de cap curs impartit durant aquell any.

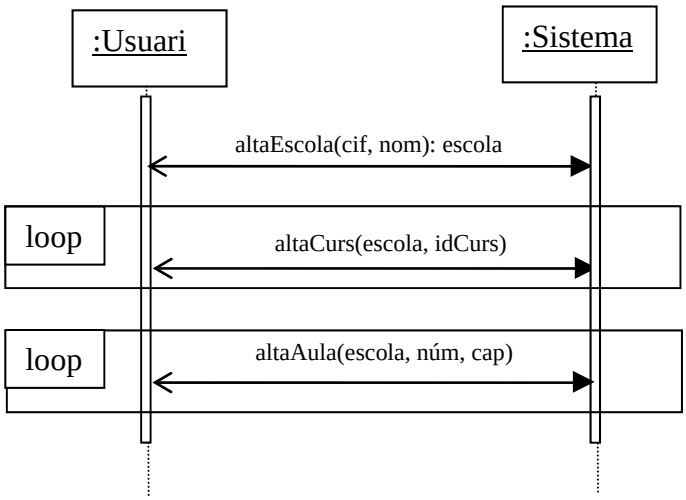
Informació derivada

1. /númConflictius d'una escola és el nombre de cursos conflictius impartits per aquella escola.
2. /PertanyA associa un curs impartit amb l'escola del seu professor responsable.

Especificació de l'esquema del comportament:

Cas d'ús: crear nova escola

Quan el regidor d'ensenyament decideix crear una nova escola li comunica a un empleat d'administració que introdueixi al sistema la nova escola amb totes les dades necessàries (és a dir, la informació de l'escola, dels cursos que ofereix i de les aules de les que disposa).



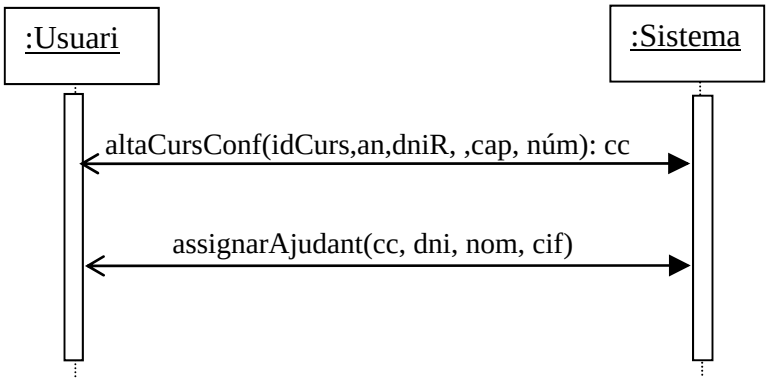
Operació:	`altaEscola(cif: String, nom:String): Escola`
Postcondició:	`Escola.allInstances()->exists(e|e.oclIsNew()and e.cif=cif and e.nom=nom and result = e)`

Operació:	`assignaCurs(e:Escola, idcurs:Integer)`
Precondició:	`Curs.allInstances()->exists(c|c.id-curs=idcurs)`
Postcondició:	`e.curs.id-curs->includes(idcurs)`

Operació:	`altaAula (e:Escola, núm:Integer, cap:Integer)`
Postcondició:	`Aula.allInstances()->exists(a|a.oclIsNew()and a.núm=núm and a.capacitat=cap and a.escola=e)`

Cas d'ús: alta de curs conflictiu

Quan el director acadèmic de l'escola vol donar d'alta un curs conflictiu, ell mateix comunica al sistema totes les dades necessàries per a fer-ho. Cal tenir en compte que el professor responsable del curs conflictiu ha d'existir al sistema, però el professor ajudant pot ser creat en aquest mateix moment (si no existia), amb tota la informació que es requereix en aquest cas. Aquest cas d'ús només es pot efectuar si en aquell moment a l'escola hi ha com a mínim tres professors que no imparteixen cap curs aquell any.



Operació:	`altaCursConflictiu(idcurs:Integer, any:Any, dnir:String, naula: Integer, cap:Integer): CursConflictiu`

Precondició:	`Curs.allInstances()->exists(c|c.id-curs=idcurs)`
	`Any.allInstances()->exists(a|a.any=any)`
	`Professor.allInstances()->exists(p|p.dni=dnir)`
	`Aula.allInstances()->exists(a|c.núm=naula)`
	`Escola.allInstances()->select(e|e.professor.dni->includes(dnir)).professor->(select(p|p.cursImpartit->isEmpty())->size()>=3`

Postcondició:	`CursConflictiu.allInstances()->exists(cc|cc.oclIsNew() and cc.curs.id-curs=idcurs and cc.responsable.dni=dnir and cc.any.any=any and cc.capacitat=cap and cc.aula.núm=naula and cc.aula.escola=cc.professor.escola and result = cc)`

Operació:	`assignarAjudant(cc:CursConflictiu, dni:String, nom:String, cif:String)`
Precondició:	`Escola.allInstances()-> exists(e|e.cif=cif)`
Postcondició:	`if not Professor.allInstances()@pre->exists(p|p.dni=dni) then Professor.allInstances()->exists(p| p.oclIsNew() and p.dni=dni and p.nom=nom and p.escola.cif=cif endif and cc.ajudant.dni=dni`

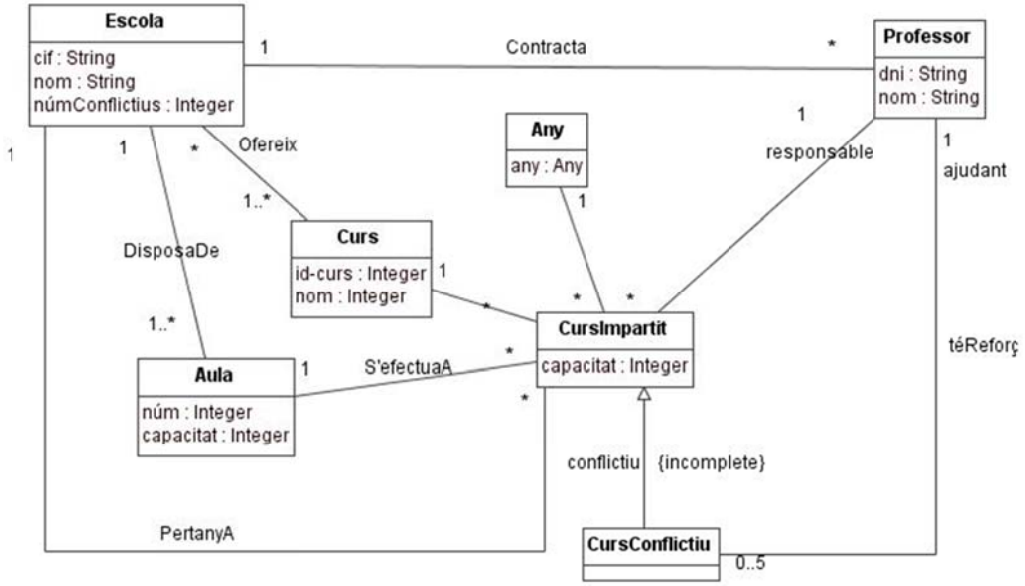
Es demana, suposant que tota la informació derivada és materialitzada :

a) Diagrama de classes de disseny obtingut a partir de l'esquema conceptual de les dades que inclogui la definició de les operacions necessàries per a tractar la informació derivada (assignades a les classes corresponents)..

b) Modificació dels contractes de les operacions, si s'escau, com a conseqüència de la traducció de l'esquema d'especificació al de disseny.

Solució:

Diagrama de classes de disseny:



RI Textuals afegides:

- No hi pot haver dos cursos impartits amb els mateixos curs i any
- No hi pot haver dos cursos impartits amb els mateixos professor i any

També caldria eliminar la classe Any i fer que fos un atribut univaluat i obligatori de la classe CursImpartit.

Modificació dels contractes de les operacions:

Operació:	`altaEscola(cif: String, nom:String): Escola`
Precondició:	
Excepcions:	`escolaJaExisteix: ja existeix una escola amb cif (clau externa)`
Postcondició:	`Escola.allInstances()->exists(e|e.oclIsNew()and e.cif=cif and e.nom=nom and result = e)`

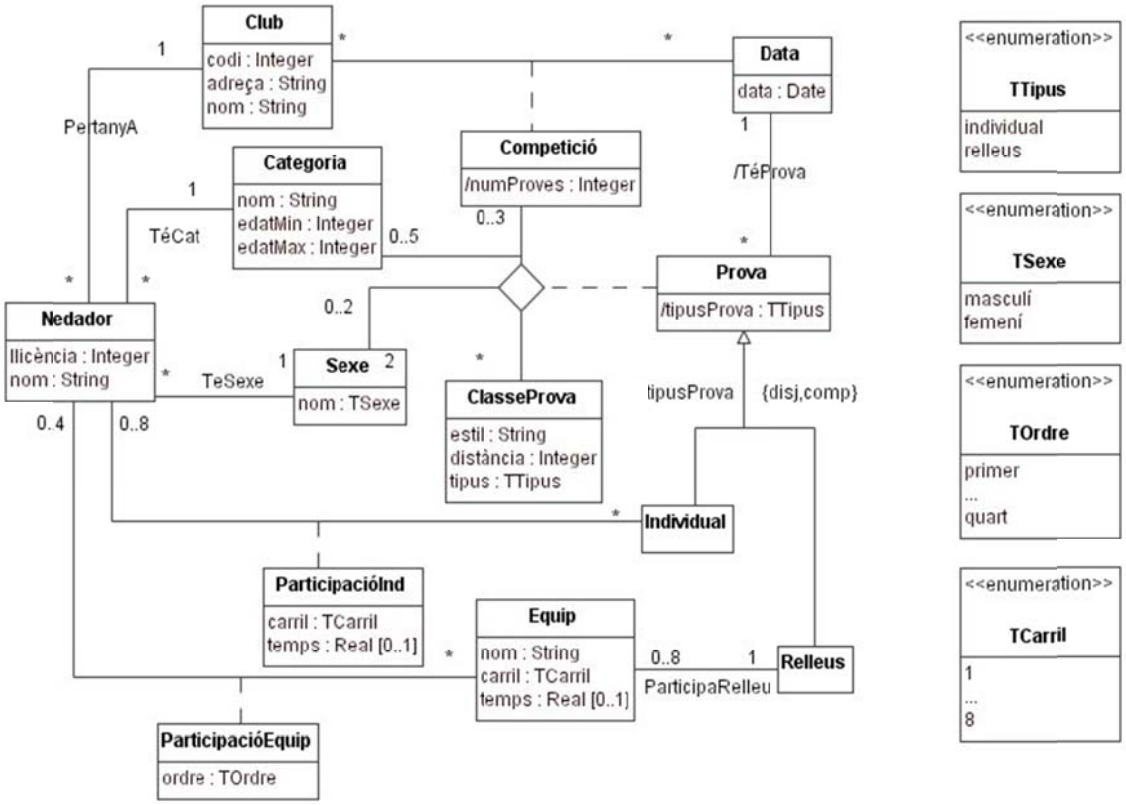
Operació:	`assignaCurs(e:Escola, idcurs:Integer)`
Precondició:	`l'escola e ja existeix`
Excepcions:	`noExisteixCurs: el curs idcurs no existeix (pre original)` `jaOfereixCurs: l'escola e ja ofereix idcurs (estructural)`
Postcondició:	`e.curs.id-curs->includes(idcurs)`

Operació:	`altaAula (e:Escola, núm:Integer, cap:Integer)`
Precondició:	`l'escola e ja existeix`
Excepcions:	`escolaJaTél'Aula: l'escola e ja té una aula núm (RI 2)`
Postcondició:	`Aula.allInstances()->exists(a|a.oclIsNew()and a.núm=núm and a.capacitat=cap and a.escola=e)`

Operació:	altaCursConflictiu(idcurs:Integer, any:Any, dnir:String, naula: Integer, cap:Integer): CursConflictiu
Precondició:	
Excepcions:	noHiHa3Profs: excepció corresponent a la pre del contracte original noExisteixCurs: el curs idcurs no existeix (pre original) noExisteixAny: l'any any no existeix (pre original) noExisteixProfessor: el professor dnir no existeix (pre original) noExisteixAula: l'aula naula no existeix (pre original) jaExisteixCursConflictiu: ja existeix un curs conflictiu identificat pel professor dnir i l'any any (estructural) jaRespEnAny: dnir ja és responsable d'algun curs a l'any (mult. 0..1) anyJaTéCurs: el curs idcurs ja s'imparteix a l'any any (mult. 0..1) escNoCoincideix: l'escola de l'aula i la de dnir no són iguals (RI 3) massaCapacitat: la capacitat d'idcurs és superior a la de l'aula (RI4)
Postcondició:	CursConflictiu.allInstances()->exists(cc|cc.oclIsNew() and cc.curs.id-curs=idcurs and cc.responsable.dni=dnir and cc.any.any=any and cc.capacitat=cap and cc.aula.núm=naula and cc.aula.escola=cc.professor.escola and result = cc) - El curs impartit s'associa amb l'escola (PertanyA) - S'incrementa en una unitat el valor de númConflictius

Operació:	assignarAjudant(cc:CursConflictiu, dni:String, nom:String, cif:String)
Precondició:	el curs conflictiu cc ja existeix
Excepcions:	noExisteixEscola: l'escola cif no existeix (pre original) ja5Conf: dni ja és ajudant de cinc cursos conflictius (mult. 0..5) ajudÉsResp: dni és el responsable del curs cc (RI 5) jaRespEnAny: dni ja és responsable d'algun curs a l'any de cc (RI6)
Postcondició:	if not Professor.allInstances()@pre->exists(p|p.dni=dni) then Professor.allInstances()->exists(p| p.oclIsNew() and p.dni=dni and p.nom=nom and p.escola.cif=cif endif and cc.ajudant.dni=dni

2. La federació catalana de natació necessita un sistema d'informació per gestionar les competicions i proves de natació que organitzen els clubs de la federació. Una competició és organitzada per un club en una data determinada. Una prova correspon a una competició, categoria, sexe i classe de prova la qual es defineix com una combinació de: estil, distància i tipus (el tipus pot ser individual o relleus). A les proves de tipus individual hi participen nedadors i a les de relleus hi participen equips de nedadors. La part rellevant de l'esquema conceptual de dades d'aquest sistema es mostra a la figura següent:



Restriccions textuals

1- Claus externes: (Club, codi), (Data, data), (Categoria, nom), (Sexe, nom), (ClasseProva, estil+distància+tipus), (Nedador, llicència).
2- L'edat mínima d'una categoria és menor o igual que la màxima.
3- En una prova de relleus, no hi poden participar dos equips amb el mateix nom.
4- En una prova individual, cada nedador que hi participa té un carril diferent.
5- En una prova de relleus, cada equip que hi participa té un carril diferent.
6- Una prova és de relleus si el tipus de la seva ClasseHora també ho és
7- Una prova és individual si el tipus de la seva ClasseHora també ho és
8- Tots els nedadors d'un equip pertanyen al mateix club.
9- En un equip, cada nedador té un ordre diferent
10- Un nedador no pot formar part de dos equips que participin a una mateixa prova
11- Els nedadors només poden nedar proves individuals o de relleus de la seva categoria i sexe.
12- Els intervals d'edats definits per les categories no es poden sobreposar.

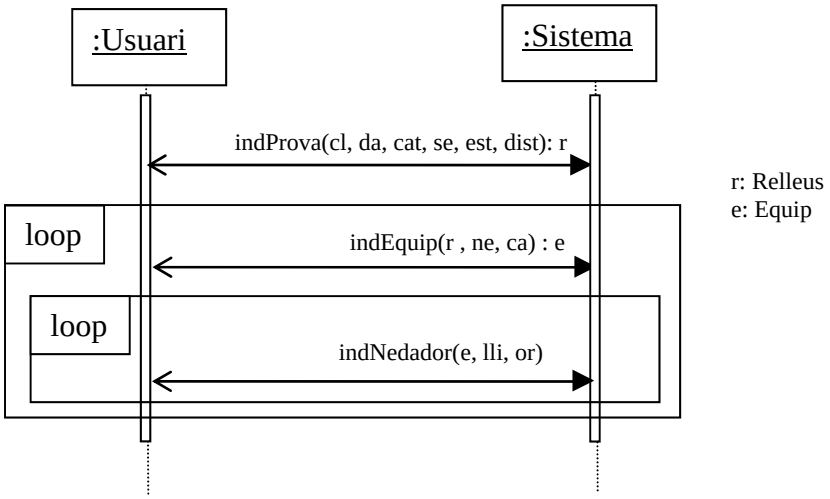
Regles de derivació

1- /tipusProva d'una prova pren el valor del tipus de la seva ClasseProva.
2- /numProves d'una competició és el nombre de proves que corresponen a la competició.
3- /TeProva associa una prova amb la data de la competició de la prova.

Especificació de l'esquema del comportament:

Cas d'ús: Crear Prova de Relleus

Quan un usuari del sistema decideix crear una nova prova de relleus indica totes les dades necessàries per identificar la prova. A continuació, per cada equip que participarà a la prova, indica el nom de l'equip, el carril, i dades de cadascun dels nedadors de l'equip, concretament, la seva llicència i el seu ordre de participació dins l'equip. Només es pot executar aquesta funcionalitat de crear prova de relleus per competicions amb una data posterior a la data actual i que ja tinguin com a mínim 10 proves individuals.



Operació:	`indProva(cl: Integer, da: Date, cat: String, se: TSexe, est: String, dist: Integer): Relleus`

Precondició:	`- Categoria.allInstances()->exists(c|c.nom = cat)`
	`- ClasseProva.allInstances()->exists(cp|cp.estil = est)`
	`- Competicio.allInstances()->exists(c|c.club.codi = cl and c.data.data = da and c.prova -> select(p|p.oclIsTypeOf(Individual)->size()<10))`
	`- da > today()`

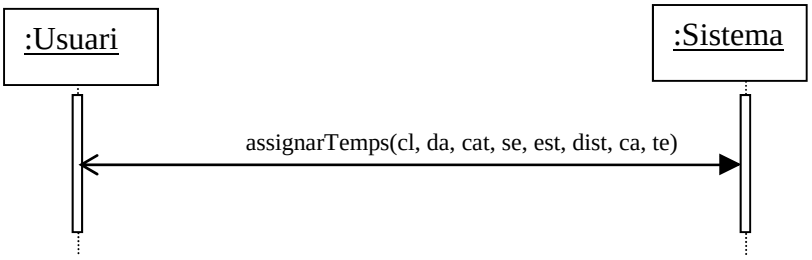
Postcondició:	`Relleus.allInstances()->exists(r|r.oclIsNew()and r.competicio.club.codi = cl and r.competicio.data.data = da and r.categoria.nom = cat and r.sexe.nom = se and r.classeProva.estil = est and r.classeProva.distancia = dist and r.classeProva.tipus = #relleus and result = r)`

Operació:	`indEquip(r: Relleus, ne: String, ca: TCarril): Equip`
Postcondició:	`Equip.allInstances()->exists(e|e.oclIsNew()and e.nom = ne and e.carril = ca and e.relleus = r and result = e)`

Operació:	`indNedador(e: Equip, lli: Integer, or: TOrdre)`
Precondició:	`Nedador.allInstances()->exists(n|n.llicència = lli)`
Postcondició:	`ParticipacióEquip.allInstances()->exists(pe|pe.oclIsNew() and pe.equip = e and pe.nedador.llicencia = lli and pe.ordre = or)`

Cas d'ús: Assignar Temps

Quan un usuari del sistema vol assignar un temps individual ha d'indicar les dades necessàries per identificar la prova, ha d'indicar el carril al qual s'ha d'assignar el temps i, finalment, el temps a assignar. Cal considerar que aquesta funcionalitat no es pot executar si la prova és de relleus (ha de ser individual), si la prova i carril indicats ja tenien temps assignat o si la data de la competició de la prova és posterior a la data actual.



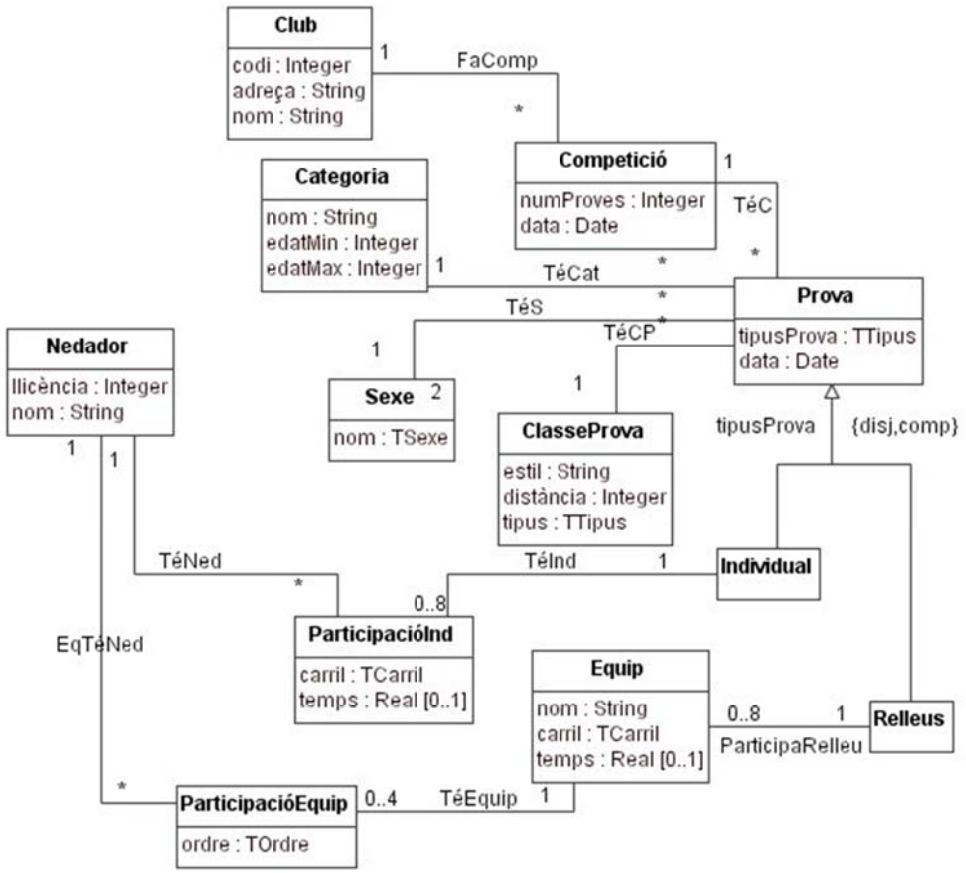
Operació:	`assignarTemps(cl: Integer, da: Date, cat: String, se: TSexe, est: String, dist: Integer, ca: TCarril, te: Real)`
Precondició:	`- ParticipacioInd.allInstances()-> exists(pi| pi.individual.competicio.club.codi = cl and pi.individual.competicio.data.data = da and pi.individual.categoria.nom = cat and pi.individual.sexe.nom = se and pi.individual.classeProva.estil = est and pi.individual.classeProva.distancia = dist and pi.carril = ca and pi.temps-> isEmpty())` `- da <= today()`
Postcondició:	`ParticipacioInd.allInstances()-> select(pi|pi.individual.competicio.club.codi = cl and pi.individual.competicio.data.data = da and pi.individual.categoria.nom = cat and pi.individual.sexe.nom = se and pi.individual.classeProva.estil = est and pi.individual.classeProva.distancia = dist and pi.carril = ca).temps = te`

Es demana, suposant que tota la informació derivada és materialitzada :

a) Diagrama de classes de disseny obtingut a partir de l'esquema conceptual de les dades que inclogui la definició de les operacions necessàries per a tractar la informació derivada (assignades a les classes corresponents).

b) Modificació dels contractes de les operacions, si s'escau, com a conseqüència de la traducció de l'esquema d'especificació al de disseny. Es pot fer en llenguatge natural.

Solució: Diagrama de classes de disseny (part modificada, la resta queda igual):



RI Textuals (afegides):

- No hi pot haver dues competicions del mateix club i data.
- No hi pot haver dues proves de la mateixa competició, categoria, sexe i classe de prova.
- No hi pot haver més de tres proves de la mateixa categoria, sexe i classe de prova.
- No hi pot haver més de cinc proves de la mateixa competició, sexe i classe de prova.
- No hi pot haver dues participacions individuals de la mateixa prova individual i nedador.
- No hi pot haver dues participacions d'equip del mateix equip i nedador.

Modificació dels contractes de les operacions:

Operació:	`indProva(cl: Integer, da: Date, cat: String, se: TSexe, est: String, dist: Integer): Relleus`

Excepcions:	`noExisteixCategoria: la categoria cat no existeix (pre original)`
	`noExisteixClasseProva: la classe de prova est no existeix (pre original)`
	`jaExisteixProva: ja existeix una prova definida per (cl, da, cat, se, est)`
	`ja10ProvesInd: Competicio.allInstances()->exists(c|c.club.codi = cl and c.data.data = da and c.prova -> select(p|p.oclIsTypeOf(Individual) ->size()>10))`
	`massaTard: da <= today()`
	`ja3ProvesCaSC: ja hi ha tres proves amb la mateixa categoria, sexe i classe`
	`ja5ProvesCoSC: ja hi ha cinc proves amb competició, sexe i classe`

Postcondició:	`Relleus.allInstances()->exists(r|r.oclIsNew()and r.competicio.club.codi = cl and r.competicio.data = da and r.categoria.nom = cat and r.sexe.nom = se and r.classeProva.estil = est and r.classeProva.distancia = dist and r.classeProva.tipus = #relleus and result = r and data=da and tipusProva=r.classeProva.tipus)` `S'incrementa en 1 el valor de l'atribut numProves de la competició del club amb codi cl i la data da.`

Observacions:

- Al contracte anterior, el sexe *se* segur que existeix perquè TSexe és una enumeració. Per tant, no cal tenir una excepció noExisteixSexe.

- Pel mateix motiu, i tenint en compte que TSexe és una enumeració de dos valors, tampoc no cal tenir una excepció per garantir la multiplicitat 0..2 que hi ha a l'extrem Sexe a l'esquema conceptual de les dades d'especificació.

Operació:	`indEquip(r: Relleus, ne: String, ca: TCarril): Equip`
Precondició:	`la prova de relleus r ja existeix`

Excepcions:	`equipJaParticipa: l'equip ne ja participa a la prova r (RI 3)`
	`carrilOcupat: el carril ca ja està ocupat per algun equip (RI 5)`
	`jaHiHa8Equips: la prova r ja té 8 equips participants (multiplicitat 0..8)`

Postcondició:	`Equip.allInstances()->exists(e|e.oclIsNew()and e.nom = ne and e.carril = ca and e.relleus = r and result = e)`

Operació:	`indNedador(e: Equip, lli: Integer, or: TOrdre)`
Precondició:	`l'equip e ja existeix`

Excepcions:	`noExisteixNedador: el nedador lli no existeix (pre original)`
	`nedadorJaÉsdeL'Equip: lli ja és de l'equip e (estructural)`
	`ordJaOcupat: l'equip e ja té un nedador que ocupi l'ordre or (RI 7)`
	`nedadorJaInscrit: el nedador lli ja forma part d'algun altre equip de la mateixa prova (RI 8)`
	`sexeOCategoriaNoCoincidents: lli no té la mateixa categoria o sexe que la prova de l'equip e (RI 9)`
	`jaHiHa4Nedadors: a l'equip e ja hi ha 4 nedadors (multiplicitat 0..4)`

Postcondició:	`ParticipacióEquip.allInstances()->exists(pe|pe.oclIsNew() and pe.equip = e and pe.nedador.llicencia = lli and pe.ordre = or)`

Operació:	`assignarTemps(cl: Integer, da: Date, cat: String, se: TSexe, est: String, dist: Integer, ca: TCarril, te: Real)`
Precondició:	
Excepcions:	`participacióNoAdequada: la que ve de la 1a pre del contracte original` `massaAviat: da > today() (2a precondició del contracte original)`
Postcondició:	`ParticipacioInd.allInstances()->` `select(pi|pi.individual.competicio.club.codi = cl and pi.individual.competicio.data = da and pi.individual.categoria.nom = cat and pi.individual.sexe.nom = se and pi.individual.classeProva.estil = est and pi.individual.classeProva.distancia = dist and pi.carril = ca).temps = te`