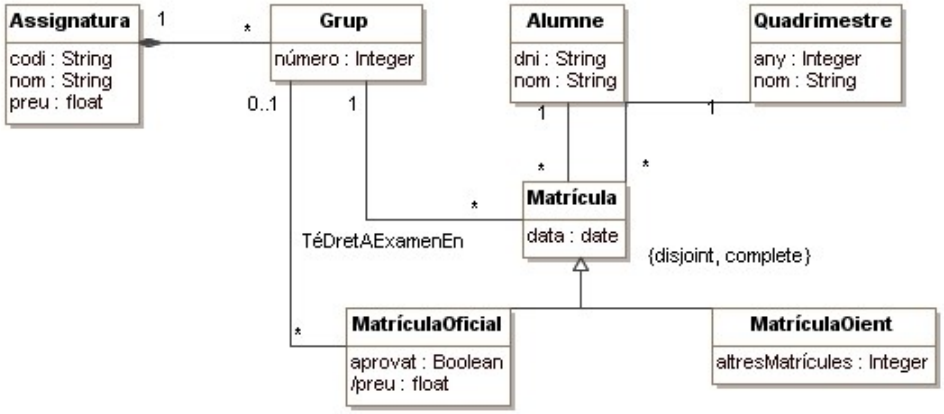
ARQUITECTURA DEL SOFTWARE

Unit 3.3: Presentation Layer Design

Enunciats Exercicis

Exercici 1

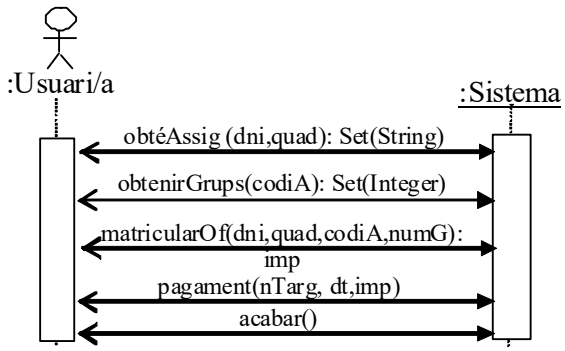
Un centre docent ens ha encarregat el disseny d'alguns casos d'ús per a l'ajuda al seu procés de matriculació. El centre oferta cada quadrimestre un conjunt d'assignatures. Cada assignatura té un codi, un nom, un preu de matrícula i un conjunt de grups. Un alumne pot tenir diverses matrícules en un quadrimestre per diferents assignatures. Les matrícules poden ser oficials o d'oients (sense dret a examen). Les matrícules oficials enregistren si l'alumne en acabar el curs ha aprovat l'assignatura i en quin grup ha fet l'examen. Les matrícules d'oient enregistren el nombre de matrícules totals que té l'alumne aquell mateix quadrimestre. A continuació disposeu de l'esquema conceptual, de la descripció del disseny extern del cas d'ús *AltaMatrículaOficial* , del diagrama de seqüència dels esdeveniments del sistema del cas d'ús concret i dels contractes de la capa de domini corresponents:



R.Integritat Textuals:
- Claus classes no associatives: (Assignatura, codi); (Alumne, dni); (Quadrimestre, nom)
RT1. Una assignatura no pot tenir dos grups amb el mateix número
RT2. No poden haver-hi dues matrícules pels mateixos grup d'assignatura, alumne i quadrimestre
RT3. Un alumne no es pot matricular en dos grups de la mateixa assignatura en un mateix quadrimestre
RT4. Un alumne en un quadrimestre no es pot matricular de més de 7 assignatures
RT5. En un quadrimestre i per un grup no poden haver-hi més de 80 matrícules
RT6. Un alumne que ha aprovat una assignatura no la pot tornar a matricular en quadrimestres posteriors
RT7. L'assignatura del grup on ha fet l'examen l'alumne amb matrícula oficial ha de ser la mateixa que l'assignatura del grup on s'ha matriculat
... altres restriccions no rellevants pel problema
Informació derivada:
ID1. El preu d'una matrícula oficial és el preu de l'assignatura matriculada

Descripció textual del disseny extern pel cas d'ús *AltaMatrículaOficial*:
1. L'usuari introdueix el dni de la persona i el quadrimestre en el que es vol matricular i prem <<OK>>.
2. A continuació, el sistema mostra un desplegable amb totes les assignatures de les que es pot matricular l'alumne. L'usuari selecciona una i prem <<OK>>.
3. El sistema mostra un missatge indicant els números de grup existents per a aquella assignatura. L'usuari introdueix en un camp de text el grup a matricular i prem <<OK>>.
4. El sistema crea la matrícula i mostra en una pantalla el seu import. A més, demana les dades del pagament (suposeu que només està permès pagar amb targeta de crèdit). L'usuari introdueix el número de la targeta i la data de caducitat i prem <<OK>>.
5. El sistema demana autorització a un sistema extern de pagament i informa a l'usuari del resultat del procés de creació de la matrícula.
6. L'usuari confirma que ha rebut el missatge.
7. El sistema acaba el cas d'ús.

8. Si en alguna de les interaccions anteriors es produeix algun dels errors possibles o es cancel·la el cas d'ús, es mostra un missatge i acaba el cas d'ús.



context CapaDomini::obtéAssig (dni: String, quad: Integer): Set(String)
exc *alumne-no-existeix:* L'alumne amb *dni* no existeix. (pre original)
exc *quadrimestre-no-existeix:* El quadrimestre *quad* no existeix. (pre original)
exc *no-hi-ha-assignatures:* L'alumne *dni* no pot matricular cap assignatura al quadrimestre *quad*. (millora de la interfície)
exc *no-pot-matricular:* L'alumne *dni* ja té 7 assignatures matriculades al quadrimestre *quad*. (RT4)
post result = conjunt de codis de les assignatures que no estan matriculades per aquest quadrimestre (RT2 i RT3) i que no estiguin aprovades per l'alumne (RT6).
post self.dniM:=dni i self.quadM:=quad.

context CapaDomini:: obtenirGrups (codiA: String): Set(Integer)
pre *assignatura-existeix*: L'assignatura amb *codiA* no existeix. (garantida per l'operació anterior)
exc *no-hi-ha-grups:* L'assignatura amb *codiA* no té grups. (millora de la interfície)
post result = conjunt de números de grup de l'assignatura *codiA*.
post self.codiAM:=codiA.

context CapaDomini::matrículaOf (númG: Int): Float (import)
exc *grup-no-existeix:* El grup *númG* no existeix. (R estructural: referencial de l'associació entre Matrícula i Grup)
exc *grup-ple:* El grup *númG* ja té 80 matrícules. (RT5)
exc *matrícula-existeix:* La matrícula existeix. (Rclau de matrícula)
post *matrículaCreada*: es crea una instància de matrícula oficial de l'alumne *dni* pel grup *númG* de l'assignatura *codiA* i pel quadrimestre *nom*.
post result= preu de la matrícula (import) i self.importM:=import.

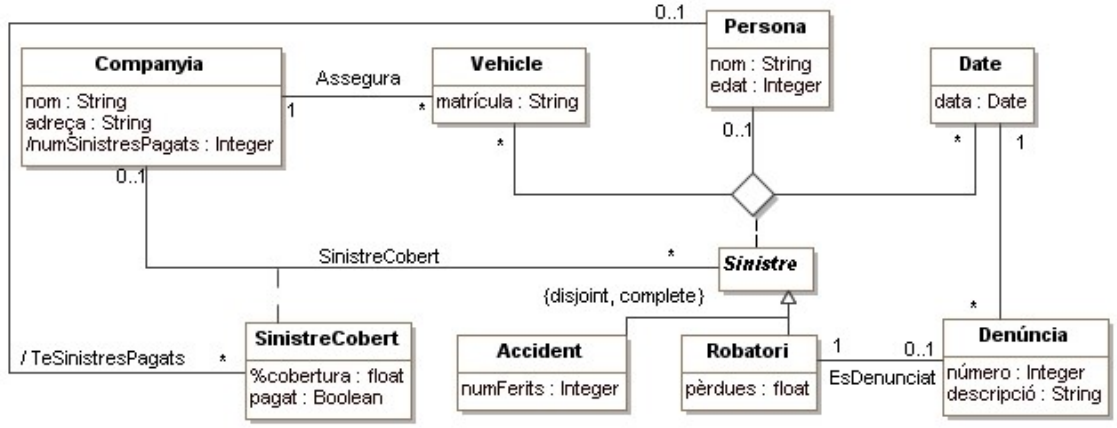
context CapaDomini::pagament (numTarg: String, dataCad: date)
exc error-servei: el servei ens retorna error
post *autorització:* el sistema invoca l'execució de l'operació *autoritzacióCàrrec* del sistema extern *SistemaAutoritzacióCàrrecs* amb el número de la targeta *numTarg*, la data de caducitat *dataCad* i l'import importM.

Es demana:
(a) Definiu el model navegacional de cas d'ús (descripció d'alt nivell).

(b) Diagrama de classes de la Capa de Presentació i de Domini (en aquesta capa només cal que definiu els controladors).
(c) Diagrames de seqüència de les operacions dels controladors de la Capa de Presentació.

Exercici 2

Un consorci de companyies d'assegurances ens ha demanat que li dissenyem una part d'un sistema software per gestionar els sinistres dels vehicles que tenen assegurats. Els vehicles s'identifiquen per matrícula i es guarda la companyia en la que estan assegurats. Una persona pot tenir un sinistre per un determinat vehicle i en una determinada data. Un sinistre pot estar cobert per una companyia asseguradora. Si ho està, se'n coneix el percentatge de cobertura i si ha estat pagat o no. A més, de tots els sinistres possibles, interessa guardar informació específica dels accidents (es vol conèixer el nombre de ferits, si n'hi ha) i dels robatoris (es vol saber l'import de les pèrdues i si han estat denunciats o no). Les denúncies s'identifiquen per un número i se'n coneix també la data en què es van fer efectives i la seva descripció. A continuació disposeu de l'esquema conceptual, de la descripció del disseny extern del cas d'ús *PagarSinistreCobert* , del diagrama de seqüència dels esdeveniments del sistema del cas d'ús concret i dels contractes de la capa de domini corresponents:

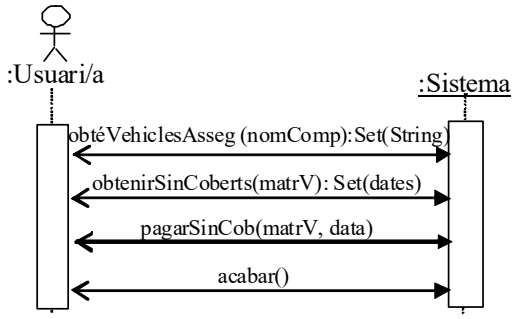


R.I. Textuals:
- Claus: (Companyia, nom); (Vehicle, matrícula); (Persona, nom); (Denúncia, número); (Date, data)
- La companyia asseguradora que cobreix el sinistre ha de ser la companyia on està assegurat el vehicle.
- Els sinistres coberts que corresponen a robatoris només poden ser pagats si tenen la denúncia corresponent.
- La data de la denúncia ha de ser posterior a la data del sinistre.
- Una persona pot tenir com a molt 100 sinistres.
- Una companyia pot pagar com a màxim 10 sinistres del mateix Vehicle.
- Una companyia no pot tenir sinistres coberts i pagats amb un % de cobertura inferior que d'altres que estiguin coberts però no pagats.
- Un vehicle no pot tenir més de 100 robatoris amb pèrdues superiors a 1000 Euros.
- Altres restriccions no rellevants pel problema

Info. derivada:
- *numSinistresPagats* de Companyia = nombre de sinistres coberts i pagats d'aquella companyia.
- *TeSinistresPagats* permet obtenir els sinistres coberts i pagats d'una persona

Descripció textual del disseny extern pel cas d'ús *PagarSinistreCobert*:
1. L'usuari introdueix el nom de la companyia i prem <<OK>>.
2. A continuació, el sistema mostra un desplegable amb totes les matrícules dels vehicles assegurats per la companyia. L'usuari selecciona un vehicle i prem <<OK>>.
3. El sistema mostra un desplegable amb les dates on hi ha sinistres coberts i no pagats pel vehicle seleccionat. L'usuari selecciona una data i prem <<OK>>.
4. El sistema enregistra que el sinistre cobert s'ha pagat i s'informa a l'usuari.

5. L'usuari confirma la recepció del missatge i acaba el cas d'ús.
6. Si en alguna de les interaccions anteriors es produeix algun dels errors possibles, es mostra un missatge i acaba el cas d'ús.



context CapaDomini::obtéVehiclesAsseg (nomComp: String): Set(String)
exc *companyia-no-existeix:* La companyia *nomComp* no existeix.
exc *no-hi-ha-vehicles:* La companyia no té vehicles assegurats.
post result = conjunt de matrícules dels vehicles assegurats per la companyia *nomComp*.

context CapaDomini::obtenirSinCoberts (matrV: String): Set(date)
pre *vehicle-existeix*: El vehicle amb matrícula *matrV* existeix.
exc *no-hi-ha-sinistresCoberts*: El vehicle amb matrícula *matrV* no té sinistres coberts.
post result = conjunt de dates on el vehicle amb matrícula *matrV* té sinistres coberts però no pagats.
post self.matr:=matrV.

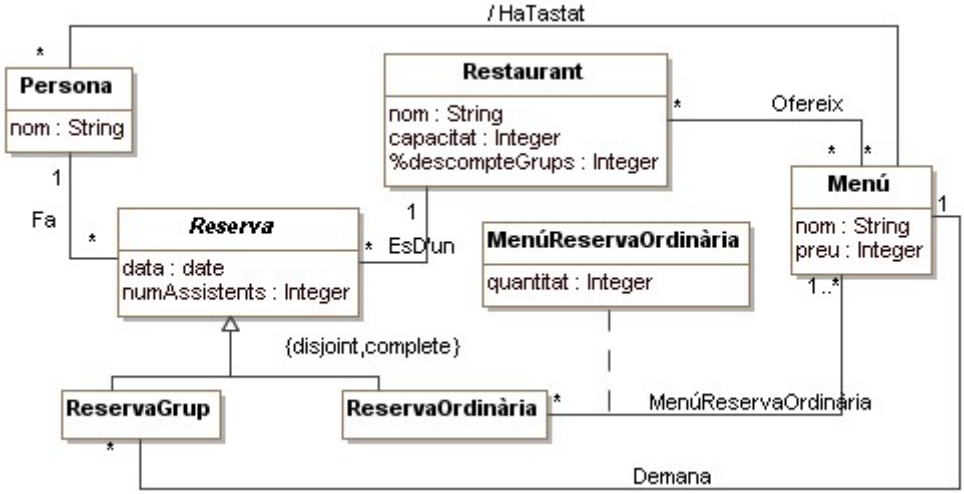
context CapaDomini::pagarSinCob (dataSin: date)
pre *sinistreCobertNoPagat*: el sinistre cobert pel vehicle *matrVehicle* i data *dataSin* existeix i no està pagat (pre original)
exc *denunciaExisteix*: si el sinistre corresponent a al sinistre cobert és un robatori i no té denuncia assignada (RT3)
exc *màx10Pagaments*: el vehicle ja té 10 sinistres coberts i pagats (RT5)
exc *coberturaInferior*: hi ha altres sinistres coberts i no pagats de la mateixa companyia amb un % de cobertura superior (RT6)
post *sinistreCobertPagat*: s'ha pagat el sinistre cobert. (post original)
post *incrementarSinPagats*: s'incrementa *numSinistresPagats* de la companyia del sinistre. (materialització de *numSinistresPagats*)
post *creacióTéSinistresPagats*: es crea una instància de l'associació entre la Persona (que ha participat en el sinistre) i el sinistre cobert. (materialització de l'associació TéSinistresPagats)

Es demana:
(a) Definiu el model navegacional de cas d'ús (descripció d'alt nivell).
(b) Diagrama de classes de la Capa de Presentació i de Domini (en aquesta capa només cal que definiu els controladors).
(c) Diagrames de seqüència de les operacions dels controladors de la Capa de Presentació.

Exercici 3

Una cadena de restaurants que ofereixen menús per sopars ens ha demanat que li dissenyem una part d'un sistema software per gestionar les seves reserves. Els clients de

la cadena fan reserves per un restaurant concret en una data determinada. Les reserves poden ser de dos tipus: les reserves de grup són reserves per a un conjunt de persones que demanen el mateix menú i ho fan en el moment de fer la reserva, i les reserves ordinàries són reserves per a una o més persones que demanen el menú que vulguin en el moment de sopar. Els restaurants ofereixen un descompte (%descompteGrups) en el preu del menú per a les reserves de grup. A continuació disposeu de l'esquema conceptual, de la descripció del disseny extern del cas d'ús *AltaReservaGrup*, del diagrama de seqüència dels esdeveniments del sistema del cas d'ús concret i dels contractes de la capa de domini corresponents:

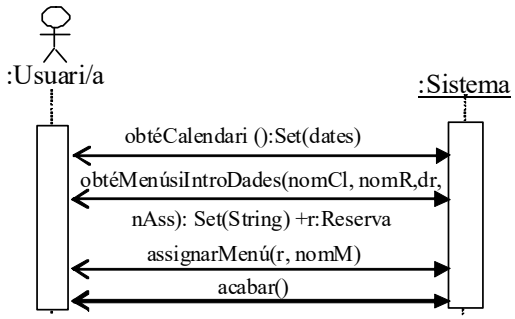


R.I. Textuals:
- Claus: (Persona, nom); (Restaurant, nom); (Menú, nom); (Reserva, nomPersona+data)
- Els menús de les reserves han de ser menús oferts pel restaurant on s'ha fet la reserva.
- La capacitat d'un restaurant ha de ser més gran que el sumatori dels assistents de les reserves d'aquell restaurant per una data determinada.
- Tots els atributs de tipus integer han de ser positius
- El numAssistents d'una reserva de grup ha de ser més gran que 5
- En una reserva ordinària el numAssistents ha de coincidir amb la quantitat de menús demanats

Informació derivada:
- HaTastat: relaciona les persones amb els menús que ha tastat o tastarà (per reserves de grup o ordinàries)

A continuació disposeu d'una descripció textual del disseny extern pel cas d'ús *AltaReservaGrup*:

1. El responsable de la central de reserves de la cadena de restaurants vol donar d'alta una nova reserva de grup. Selecciona l'opció corresponent i prem <<OK>>.
2. El sistema mostra una pantalla per introduir el nom del client, el nom del restaurant, el número de assistents i la data. La data a seleccionar es mostra en un calendari de dies a partir d'avui.
3. El responsable introdueix les dades anteriors i prem <<OK>>.
4. A continuació, el sistema enregistra la reserva de grup i mostra un desplegable amb tots els menús oferts pel restaurant.
5. El responsable selecciona un menú i prem <<OK>>.
6. El sistema enregistra que per a la reserva de grup creada es demana el menú seleccionat i informa al responsable. El responsable confirma la recepció del missatge. El cas d'ús acaba.
7. Si en alguna de les interaccions anteriors es produeix algun dels errors possibles, es mostra un missatge i acaba el cas d'ús.



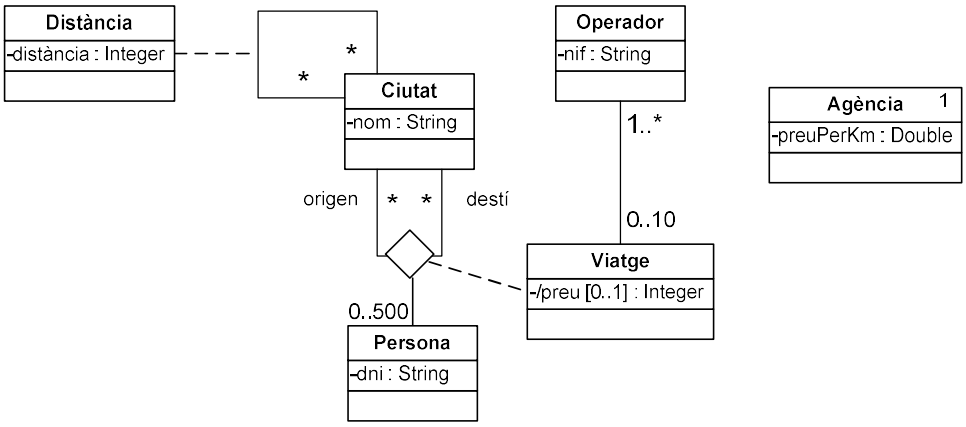
context CapaDomini::obtéMenúsiIntroDades (nomCl: String, nomR:String, dr:date, nAss:Integer): Set(String)
pre *numAssOk*: el *nAss* és més gran que 5 (RT)
exc *clientNoExisteix*: el client *nomCl* no existeix (R estructural- int referencial)
exc *restaurantNoExisteix*: el restaurant *nomR* no existeix (R estructural- int referencial)
exc *restaurantSenseCapacitat*: El restaurant no té capacitat per la reserva (RT)
exc *reservaGrupExisteix*: La reserva de grup ja existeix (RT Clau de reserva)
post *reservaGrupCreada*: s'ha creat la reserva de grup (r). (post original)
post *obtéMenús*: result= nom dels menús oferts pel restaurant nomR
post *reservaGuardada*: self.reserva:= r

context CapaDomini::assignarMenú (nomM: String)
pre *menúServit*: el menú *nomM* existeix
post *assignaMenú*: el menú s'assigna a la reserva
post *haTastat*: si la persona (de la reserva) no ha tastat el menú aleshores es dóna d'alta una instància de l'associació *HaTastat* entre la persona i el menú.

Es demana:
(a) Definiu el model navegacional de cas d'ús (descripció d'alt nivell).
(b) Diagrama de classes de la Capa de Presentació i de Domini (en aquesta capa només cal que definiu els controladors).
(c) Diagrames de seqüència de les operacions dels controladors de la Capa de Presentació.

Exercici 4

Una Agència de viatges organitza Viatges per a Persones d'una Ciutat origen a una Ciutat destí. Els viatges tenen un preu calculat en funció de la distància entre les ciutats i el preuPerKm fixat per l'agència. Cada viatge té assignats un o més Operadors potencials. A continuació disposeu de l'esquema conceptual, de la descripció del disseny extern del cas d'ús *AltaViatge*, del diagrama de seqüència dels esdeveniments del sistema del cas d'ús concret i dels contractes corresponents:

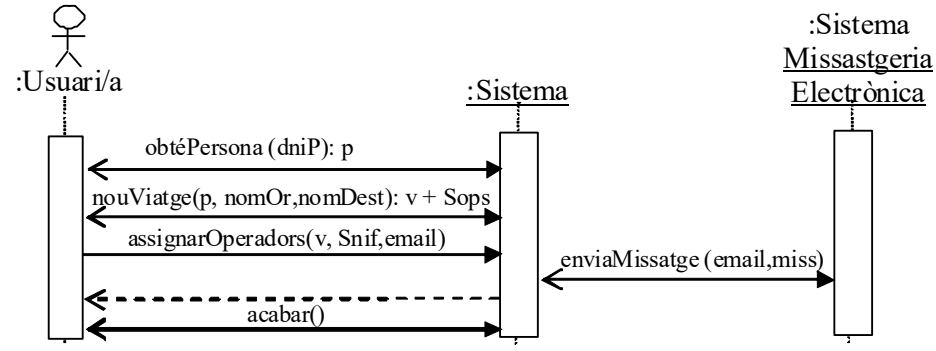


RT1. Claus: (*Ciutat*, *nom*); (*Persona*, *dni*); (*Operador*, *nif*)
RT2. Les ciutats *origen* i *destí* d'un viatge són diferents
RT3. Una persona no pot tenir més de 100 viatges
...altres restriccions que no són d'interès per al problema

ID1. El *preu* d'un viatge és igual a la *distància* establerta entre la seva ciutat origen i destí, multiplicada pel *preuPerKm* estipulat per l'agència. Si les dues ciutats no estan directament relacionades per *Distància*, l'atribut *preu* té valor nul.

Tot seguit disposeu de la descripció textual del cas d'ús concret:

1. L'usuari introdueix el dni de la persona i prem <<OK>>.
2. A continuació, l'usuari introdueix les dues ciutats i prem <<OK>>.
3. El sistema enregistra el viatge i mostra en una llista tots els operadors amb menys de 10 viatges assignats que, per tant, poden ser assignats com a operadors potencials a aquest nou viatge.
4. Finalment, l'usuari marca en aquesta llista els operadors potencials d'aquest viatge que desitja, introdueix el mail del client i prem <<OK>>.
5. El sistema enregistra els operadors del viatge, informa a l'usuari i invoca al sistema extern de missatgeria electrònica per enviar un mail amb les dades del viatge al client.
6. L'usuari confirma la recepció del missatge i s'acaba el cas d'ús.
7. Si es produeix algun dels errors possibles, es mostra un missatge i acaba el cas d'ús.



context CapaDomini::obtéPersona (dniP:String)
exc *personaNoExisteix*: la persona amb dni *dniP* existeix.
post self.persona = persona amb dni dniP.

context CapaDomini::nouViatge (nomOr: String, nomDest: String): Sops:Set(string)
pre *ciutatsDiferents*: les ciutats origen i destí són diferents. (RT1)
exc *ciutatOrigenNoExisteix*: la ciutat amb nom *nomOr* no existeix. (R. referencial)
exc *ciutatDestíNoExisteix*: la ciutat amb nom *nomDest* no existeix. (R. referencial)
exc *viatgeExisteix*: el viatge per la persona, ciutat origen i destí existeix. (R clau viatge)
exc *personaAmbMoltsViatges*: la persona ja té 100 viatges. (RT2)
exc *ciutatsAmbMoltsViatges*: les ciutats origen i destí ja tenen 500 viatges. (R gràfica)
post *viatgeCreat*: crea una instància de viatge de *p* entre *nomOr* i *nomDest* i s'assigna el preu.
post self.viatge = viatge creat.
post result = nif dels operadors que tenen assignats menys de 10 viatges.

context CapaDomini::assignarOperadors (Snif: Set(String), email:String)
pre *operadorsExisteixen*: els operadors de Snif existeixen. (R. referencial)
pre *operadorsSenseMaxViatges*: els operadors de Snif no tenen 10 viatges assignats. (R. gràfica)
pre *operadorsNoRepetits*: si hi ha operadors a Snif, no hi ha repetits. (R. implícita no repetits associacions)
pre *viatgeAmbOperadors*: hi ha operadors a Snif. (R gràfica)
post *operadorsAssociats*: per tot nif *n* dins de *Snif*, associa el viatge *v* amb l'operador de nif *n*.
post *enviaMail*: s'invoca l'execució de l'operació *enviaMissatge* del sistema de *MissatgeriaElectrònica* amb l'email i el missatge de text amb les dades del viatge.

Es demana:
(a) Definiu el model navegacional de cas d'ús (descripció d'alt nivell).
(b) Diagrama de classes de la Capa de Presentació i de Domini (en aquesta capa només cal que definiu els controladors).
(c) Diagrames de seqüència de les operacions dels controladors de la Capa de Presentació.