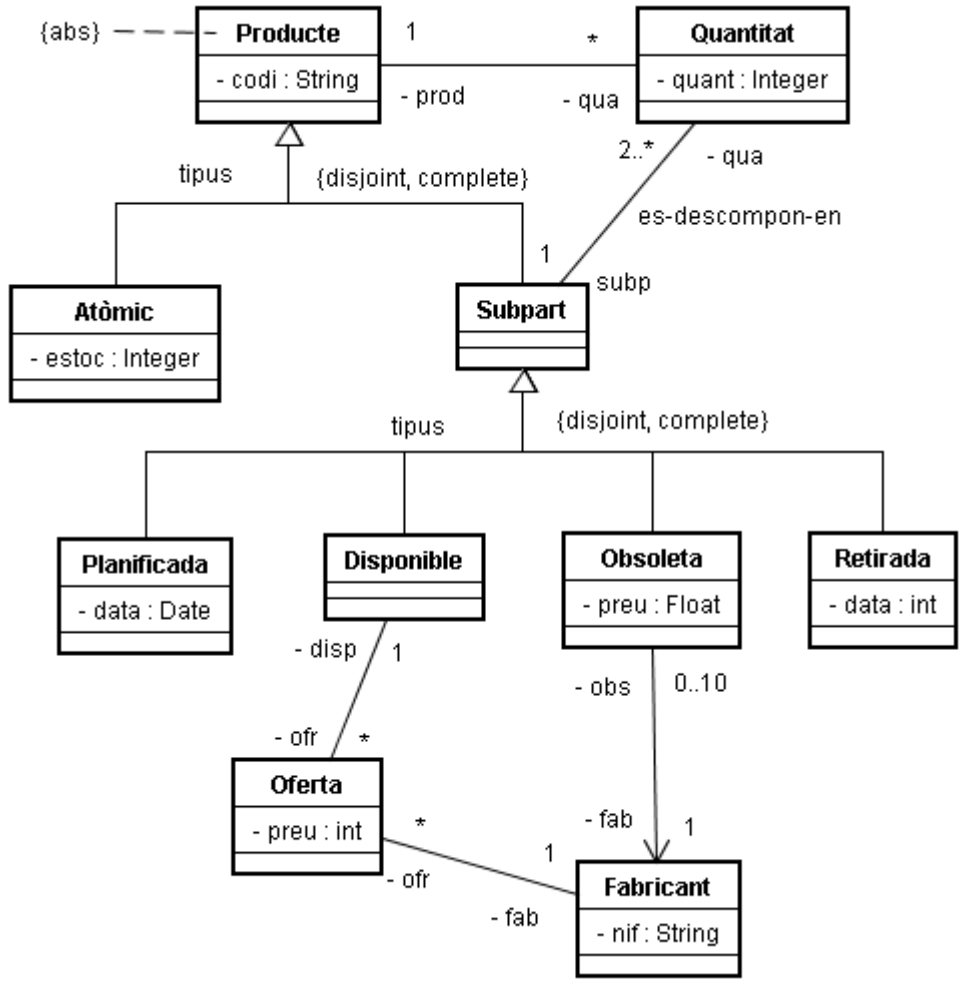
*Modificacions: preu de la classe Oferta ha de ser de tipus Float, data de la classe Retirada ha de ser Date i l'operació calculaQuantitatNecessària ha de retornar un Integer

Tercer acte avaluatori ES2 **26-nov-08**

Entregueu en el munt del vostre professor
Resolgueu les dues preguntes de teoria en una única fulla

Problema (8 punts). Els grans magatzems *El Mueble Alegre* porten un registre dels productes (és a dir, mobles) que comercialitza. Un Producte (p.e., un moble-escritori) es descompon en diverses Subparts (p.e., una taula, dues calaixeres, etc.), assemblades amb components Atòmics (p.e., 4 cargols, 4 famelles, etc.). Les subparts, al seu torn, són també productes i com a tals es poden descompondre més (així, una calaixera està formada per tres calaixos, rodes, etc., assemblades per 3 guies, 2 cargols, etc.). Pel que fa a les subparts, passen per un cicle de vida: primer es Planifiquen (amb una data d'entrada al sistema), en algun moment passen a estar Disponibles, després es consideren Obsoletes (però encara no es retiren del catàleg) i finalment es Retiren (en una data determinada). Per les Subparts Disponibles, pot haver-hi Ofertes de diversos Fabricants cadascuna a un preu de muntatge determinat. Per les Subparts Obsoletes, un d'aquests fabricants és elegit per al muntatge fins que la subpart es retiri. L'especificació següent formalitza aquest enunciat (les classes Quantitat i Oferta han sorgit de la normalització de sengles classes associatives):



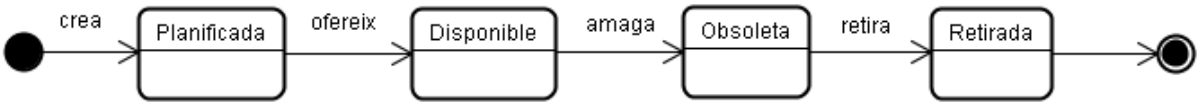
Restriccions textuals rellevants per al problema (n'hi ha d'altres que no ens afecten):

RT1. Claus: (Producte, codi), (Fabricant, nif)

RT2. No pot haver-hi dues Quantitats entre les mateixes Subpart i Producte

RT3. No pot haver-hi dues Ofertes entre els mateixos Fabricant i Subpart Disponible
RT4. Un Producte Subpart no pot formar part de la seva pròpia descomposició en Productes (ni directament ni transitivament)

Diagrama d'estats:



Considereu les operacions següents de la capa de domini:

context calculaQuantitatNecessària(codiP: String, codiAtom: String)
pre *és-subpart*: existeix una Subpart amb codi = codiP
pre *és-component-atòmic*: existeix un component Atòmic amb codi = codiAtom
post *calcula-quantitat*: retorna la quantitat necessària del component Atòmic codiAtom per construir una unitat de la subpart codiP

context obtéSubparts(nifF: String, codiAtom: String): Set(codi: String)
pre *és-component-atòmic*: existeix un component Atòmic amb codi = codiAtom
exc *fabricant-no-existeix*: no existeix Fabricant amb nif = nifF
post *obté-subparts*: retorna el conjunt de codis de Subparts Disponibles ofertades pel Fabricant nifF, que fan ús del component Atòmic codiAtom

context amaga(codiP: String, nifF: String)
exc *producte-no-existeix*: no existeix Producte amb codi = codiP
exc *és-atòmic*: el Producte codiP és un component Atòmic
exc *subpart-no-disponible*: la Subpart codiP no està Disponible
exc *fabricant-no-existeix*: no existeix Fabricant amb nif = nifF
exc *fabricant-no-oferta*: el Fabricant nifF no té Oferta per a la Subpart Disponible codiP
exc *fabricant-sobrecarregat*: el Fabricant nifF ja té encarregat el muntatge de 10 Subparts Obsoletes
post *canvia-estat*: canvia codiP de Disponible a Obsoleta, enregistrant com a preu, el preu de l'Oferta que el Fabricant nifF tenia per a codiP
post *desassocia-ofertes*: elimina les Ofertes existents per a codiP
post *associa-fabricant*: enregistra nifF com a Fabricant per al muntatge de codiP

En les operacions consultores, cal tenir en compte la relació de descomposició d'una subpart. Així, en l'exemple anterior, si preguntem per la quantitat necessària de cargols per construir un moble escritori, el resultat és 8 (4 cargols directament pel moble escritori, i com el moble escritori té 2 calaixeres i cada calaixera necessita 2 cargols, doncs en surten 8). Suposeu que inicialment, l'única navegabilitat prefixada va d'Obsoleta a Fabricant. Useu controlador cas d'ús (cada operació està a un cas d'ús diferent).

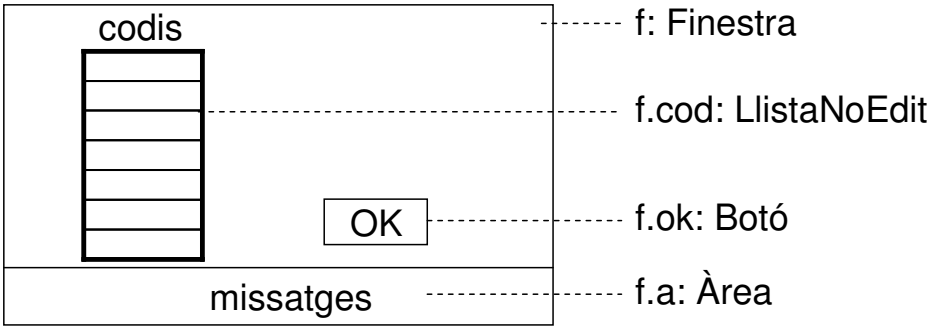
Es demana:

- diagrama de classes de disseny, que ha d'incloure totes les classes de la capa de domini, les navegabilitats finals i acoblament de les classes controladores a la resta de classes de la capa de domini. No cal incloure-hi les operacions.
- diagrames de seqüència de les tres operacions i de totes les auxiliars que hi apareguin (tret de les getter i setter).

(PREGUNTES DE TEORIA A LA PART DE DARRERA.)

TEORIA (1 punt cada pregunta)

Problema 1. Es vol dissenyar un cas d'ús per mostrar codis de productes. Suposeu com a disseny extern del cas d'ús la finestra següent, en la què també mostrem els noms dels objectes que s'usen en la capa de presentació per representar aquesta interfície:



El disseny de les capes de presentació i domini ha resultat en les dues operacions següents:

context CapaDePresentació::prémerOK()
post si hi ha productes en el sistema, els mostra usant l'espai de la llista que forma part de la finestra; altrament, mostra el missatge d'error "No hi ha productes" a l'àrea de missatges de la finestra

context CapaDeDomini::totsProductes(): Set(codi: String)
exc *no-hi-ha-productes*: no hi ha cap producte en el sistema
post retorna el conjunt de codis de productes del sistema

Es demana que dissenyeu el diagrama de seqüència de l'operació *prémerOK()* suposant controlador transacció *TxTotsProductes* en la capa de domini. NO cal que dissenyeu *totsProductes()*.

Problema 2. Sigui la classe *B* amb mètode *f* que és invocat per la classe *A* tal i com es mostra a la figura. Suposeu que es vol afegir un representant (*proxy*) de *B*. Modifiqueu la figura (tant diagrama de classes com de seqüència) aplicant-hi el patró Representant. Ha de quedar clar quines operacions hi ha a la vostra solució, si són abstractes o concretes i, en cas de ser concretes, quin és el seu diagrama de seqüència.