Trabajo práctico 2: Diseño
"Broker System"

Normativa

Límite de entrega: martes 27 de Mayo de 2014 a las 22:00 hs.
Normas de entrega: Ver "Información sobre la cursada" en el sitio Web de la materia.
(`http://www.dc.uba.ar/materias/aed2/2014/1c/informacion`)

Versión: 2.2 del 17/5/14 (ver TP2 Changelog)

Enunciado

El objetivo de este trabajo práctico consiste en realizar el diseño completo del módulo que se explica con el TAD WOLFIE (ver sección Especificación), lo que implica diseñar todos aquellos módulos necesarios para la tarea.

Con respecto al enunciado del TP1 hay algunos cambios:

- Las promesas se ejecutan únicamente ante un cambio de cotización.
- El orden para ejecutar las promesas pendientes ahora está definido y es el siguiente:
  1. Se ejecutan las de venta.
  2. En el momento de ejecutar promesas de compra se le da prioridad a los clientes que mayor cantidad total de acciones poseen.

Estos cambios están especificados como son deseados. Confíen en la axiomatización más que en el español.

Contexto de uso y complejidades requeridas

Se requiere que las operaciones exportadas de los TADs se mapeen, en lo posible, uno a uno en las interfaces de los módulos correspondientes.

Ademas, las operaciones indicadas a continuación, deberán cumplir las complejidades temporales[1] detalladas.

- **clientes**(w) es $O(1)$ y debe devolver un iterador.
- **títulos**(w) es $O(1)$ y debe devolver un iterador.
- **promesasDe**(c, w) es $O(T \cdot C \cdot |max\_nt|)$ y $O(1)$ para llamados consecutivos con el mismo c. Se puede devolver un iterador.
- **accionesPorCliente**(c, nt, w) es $O(log(C) + |nt|)$
- **inaugurarWolfie**(cs, w) es $O(\#(cs)^2)$
- **agregarTítulo**(t, w) es $O(|nombre(t)| + C)$
- **actualizarCotización**(nt, cot, w) es $O(C \cdot |nt| + C \cdot log(C))$
- **agregarPromesa**(c, p, w) es $O(|titulo(p)| + log(C))$
- **enAlza**(nt, w) es $O(|nt|))$

donde t es un título, nt es el nombre de algún título y $|nt|$ su longitud. P es la cantidad de promesas pendientes del Wolfie w, C es la cantidad de clientes de w, T es la cantidad de títulos de w, y $|max\_nt|$ la longitud máxima entre todos los nombres de los títulos de w.

Algunas consideraciones

- Hay estructuras que pueden servir para más de una finalidad, sobre todos los contenedores.
- Pueden crear módulos adicionales si así lo necesitan.
- No todas las operaciones exportadas de los TADs deben corresponder operaciones en los módulos. Diseñen sólo lo necesario.

[1] Se desestimarán los costos de eliminación de objetos, con lo cual pueden ignorarlos en el cálculo de complejidades.

- Todas las operaciones deben ser especificadas formalmente según las herramientas vistas en clase. Agregar comentarios necesarios para entender la forma en la cual deben ser utilizadas para su correcto funcionamiento.
- Todos los algoritmos deben tener su desarrollo que justifique los órdenes de complejidad. Si algún paso es no trivial pueden hacer notas a continuación del mismo.
- Cuando se formalicen los invariantes y funciones de abstracción, deben identificar cada parte de la fórmula y comentar en castellano lo que describe, para facilitar su seguimiento y corrección.
- Tener en cuenta que las complejidades son en peor caso. Soluciones más eficientes serán bien recibidas.
- Cuentan con los siguiente TADs y módulos:
  - CHAR que representan los posibles caracteres. Siendo un tipo enumerado de 256 valores. con funciones ord y ord^{-1} para la correspondencia de cada $Char$ a Nat.
  - STRING como sinónimo de $Vector(Char)$.
  - Todos los definidos en el apunte de TADs básicos.
  - Todos los definidos en el apunte de módulos básicos.

Especificación

Renombres de TADs

TAD Cliente es Nat
TAD Dinero es Nat
TAD Nombre es String
TAD TipoPromesa es Enum {compra, venta}
TAD ParPC es Tupla(Promesa,Cliente)

TAD Promesa

TAD Promesa

igualdad observacional

$$(\forall p, p' : \text{promesa})\ \left(p =_{\text{obs}} p' \iff \begin{pmatrix}\text{título(p)} = \text{título(p')} \wedge \text{tipo(p)} = \text{tipo(p')} \wedge \text{limi-}\\ \text{te(p)} = \text{limite(p')} \wedge \text{cantidad(p)} = \text{cantidad(p')}\end{pmatrix}\right)$$

géneros promesa

exporta promesa, generadores, observadores, otras operaciones

usa TipoPromesa, Dinero, Nombre, Nat

observadores básicos

título : promesa $\longrightarrow$ nombre
tipo : promesa $\longrightarrow$ tipoPromesa
límite : promesa $\longrightarrow$ dinero
cantidad : promesa $\longrightarrow$ nat

generadores

crearPromesa : nombre $\times$ tipoPromesa $\times$ dinero $\times$ nat $\longrightarrow$ promesa

otras operaciones

promesaEjecutable : promesa $\times$ dinero $\times$ nat $\longrightarrow$ bool
compraVenta : nat $\times$ conj(promesa) $\longrightarrow$ nat
promesasSobreTítulo : nombre $\times$ tipoPromesa $\times$ conj(promesa) $\longrightarrow$ conj(promesa)

axiomas $\forall t$: nombre $\forall n, m$: nat $\forall tipo$: tipoPromesa

título(crearPromesa(t, tipo, n, m)) $\equiv$ t
tipo(crearPromesa(t, tipo, n, m)) $\equiv$ tipo
límite(crearPromesa(t, tipo, n, m)) $\equiv$ n
cantidad(crearPromesa(t, tipo, n, m)) $\equiv$ m

promesaEjecutable(p,cot,disp) $\equiv$ (tipo(p) = venta $\wedge$ cot $<$ límite(p)) $\vee$
(tipo(p) = compra $\wedge$ cot $>$ límite(p) $\wedge$ disp $\geq$ cantidad(p))

```
compraVenta(disp, ps) ≡ if vacío?(ps) then
                            disp
                        else
                            if tipo(dameUno(ps)) = compra then
                                compraVenta(disp + cantidad(dameUno(ps)), sinUno(ps))
                            else
                                compraVenta(disp - cantidad(dameUno(ps)), sinUno(ps))
                            fi
                        fi
```

promesasSobreTítulo(t1,tp,ps) ≡ **if** vacío?(ps) **then**
∅
else
promesasSobreTítulo(t1, tp, sinUno(ps)) ∪
(**if** t1 = título(dameUno(ps)) ∧ tp = tipo(dameUno(ps)) **then**
{dameUno(ps)}
else
∅
fi)
fi

Fin TAD

TAD Título

TAD Título

igualdad observacional

$$(\forall t, t' : \text{título}) \left(t =_{\text{obs}} t' \iff \begin{pmatrix} \text{nombre(t)} = \text{nombre(t')} \wedge \text{cotización(t)} = \text{cotización(t')} \\ \wedge\ \#\text{máxAcciones(t)} = \#\text{máxAcciones(t')} \wedge \\ \text{enAlza?(t)} \iff \text{enAlza?(t')} \end{pmatrix} \right)$$

géneros título

exporta título, generadores, observadores, otras operaciones

usa Nombre, Dinero, Nat

observadores básicos

nombre	: título	⟶	nombre
#máxAcciones	: título	⟶	nat
cotización	: título	⟶	dinero
enAlza	: título	⟶	bool

generadores

crearTítulo	: nombre × dinero × nat	⟶	título
recotizar	: dinero × título	⟶	título

otras operaciones

cotizaciónActual	: nombre *nomTit* × conj(título) *ts*	⟶	nat
			{(∃t: título)(t ∈ ts ∧ nombre(t) = nomTit)}
cambiarCotización	: nombre *nomTit* × dinero *cot* × conj(título) *ts*	⟶	conj(título)
			{(∃t: título)(t ∈ ts ∧ nombre(t) = nomTit)}
límiteTenencia	: nombre *nomTit* × conj(título) *ts*	⟶	nat
			{(∃t: título)(t ∈ ts ∧ nombre(t) = nomTit)}
tituloEnAlza	: nombre *nomTit* × conj(título) *ts*	⟶	bool
			{(∃t: título)(t ∈ ts ∧ nombre(t) = nomTit)}

axiomas ∀s: nombre ∀n, c, c′: nat

nombre(crearTítulo(s,c,n))	≡	s
nombre(recotizar(c,t))	≡	nombre(t)
#máxAcciones(crearTítulo(s,c,n))	≡	n
#máxAcciones(recotizar(c,t))	≡	#máxAcciones(t)
cotización(crearTítulo(s,c,n))	≡	c
cotización(recotizar(c,t))	≡	c
enAlza(crearTítulo(s,c,n))	≡	true
enAlza(recotizar(c,t))	≡	c > cotización(t)

cotizaciónActual(t1, ts) ≡ **if** nombre(dameUno(ts)) = t1 **then**
cotización(dameUno(ts))
else
cotizaciónActual(t1, sinUno(ts))
fi

cambiarCotización(t1, cot, ts) ≡ **if** vacío?(ts) **then**
∅
else
if nombre(dameUno(ts)) = t1 **then**
Ag(recotizar(cot,dameUno(ts))), sinUno(ts))
else
Ag(dameUno(ts),cambiarCotización(t1, cot, sinUno(ts)))
fi
fi

límiteTenencia(t1,ts) ≡ **if** nombre(dameUno(ts)) = t1 **then**
#máxAcciones(dameUno(ts))
else
límiteTenencia(t1,sinUno(ts))
fi
títuloEnAlza(t1,ts) ≡ **if** nombre(dameUno(ts)) = t1 **then**
enAlza(dameUno(ts))
else
títuloEnAlza(t1,sinUno(ts))
fi

Fin TAD

TAD Wolfie

TAD Wolfie

géneros wolfie

exporta wolfie, generadores, observadores, enAlza

usa Bool, Nat, Dinero, Promesa, Cliente, Título, Conjunto(α), Nombre, TipoPromesa, parPC

igualdad observacional

$$(\forall w1, w2 : \text{wolfie}) \left(w1 =_{\text{obs}} w2 \iff \left(\begin{array}{l} \text{clientes(w1)} = \text{clientes(w2)} \wedge \text{títulos(w1)} = \text{títulos(w2)} \wedge_{\text{L}} (\forall\ \text{c:cliente, t:título})\ ((\text{c} \in \text{clientes(w1)} \wedge \text{t} \in \text{títulos(w1)}) \Rightarrow_{\text{L}} (\text{promesasDe(c, w1)} = \text{promesasDe(c, w2)}) \wedge \text{accionesPorCliente(c, nombre(t), w1)} = \text{accionesPorCliente(c, nombre(t), w2)}) \end{array} \right) \right)$$

observadores básicos

clientes	: wolfie	$\longrightarrow$	conj(cliente)	
títulos	: wolfie	$\longrightarrow$	conj(título)	
promesasDe	: cliente c × wolfie w	$\longrightarrow$	conj(promesa)	{c ∈ clientes(w)}
accionesPorCliente	: cliente c × nombre $nomTit$ × wolfie w	$\longrightarrow$	nat	{c ∈ clientes(w) ∧ (∃t: título)(t ∈ títulos(w) ∧ nombre(t) = nomTit)}

generadores

inaugurarWolfie	: conj(cliente) cs	$\longrightarrow$	wolfie	{¬ vacío(cs)}
agregarTítulo	: título t × wolfie w	$\longrightarrow$	wolfie	{(∀t_2: título)(t_2 ∈ títulos(w) ⇒ nombre(t) ≠ nombre(t_2))}
actualizarCotización	: nombre $nomTit$ × nat cot × wolfie w	$\longrightarrow$	wolfie	{(∃t: título)(t ∈ títulos(w) ∧ nombre(t) = nomTit)}
agregarPromesa	: cliente c × promesa p × wolfie w	$\longrightarrow$	wolfie	

$$\left\{\begin{array}{l}(\exists t\text{: título})(\text{t} \in \text{títulos(w)} \wedge \text{nombre(t)} = \text{título(p)}) \wedge \text{c} \in \text{clientes(w)} \wedge_{\text{L}} (\forall p_2\text{: promesa})(p_2 \\ \in \text{promesasDe(c, w)} \Rightarrow (\text{título(p)} \neq \text{título}(p_2) \vee \text{tipo(p)} \neq \text{tipo}(p_2))) \wedge (\text{tipo(p)} = \text{vender} \\ \Rightarrow \text{accionesPorCliente(c, título(p), w)} \geq \text{cantidad(p)}))\end{array}\right\}$$

otras operaciones

enAlza	: nombre $nomTit$ × wolfie w		⟶ bool
		{(∃t: título)(t ∈ títulos(w) ∧ nombre(t) = nomTit)}	
accDisponibles	: nombre $nomTit$ × wolfie w		⟶ nat
		{(∃t: título)(t ∈ títulos(w) ∧ nombre(t) = nomTit)}	
tenenciasDeTodos	: nombre $nomTit$ × conj(cliente) cs × wolfie w		⟶ nat
	{cs ⊆ clientes(w) ∧ (∃t: título)(t ∈ títulos(w) ∧ nombre(t) = nomTit)}		
promesasAEjecutarPorCliente	: nombre $nomTit$ × dinero cot × cliente c × wolfie w		⟶ conj(promesa)
	{c ∈ clientes(w) ∧ (∃t: título)(t ∈ títulos(w) ∧ nombre(t) = nomTit)}		
filtrarPorCliente	: cliente × conj(parPC)		⟶ conj(promesa)
ventasAEjecutar	: nombre $nomTit$ × dinero cot × wolfie w		⟶ conj(parPC)
		{(∃t: título)(t ∈ títulos(w) ∧ nombre(t) = nomTit)}	
comprasAEjecutar	: nombre $nomTit$ × dinero cot × wolfie w		⟶ conj(parPC)
		{(∃t: título)(t ∈ títulos(w) ∧ nombre(t) = nomTit)}	
promesasAEjecutar	: dinero cot × nat $disp$ × secu(parPC) $pends$		⟶ conj(parPC)
aparearPC	: nombre × tipoPromesa × conj(cliente) × wolfie		⟶ conj(parPC)
accVendidas	: conj(parPC)		⟶ nat
promesasOrdenAccClie	: conj(parPC) cs × conj(parPC) vs × wolfie w		⟶ secu(parPC)
ordenSecu	: conj(parPC) × secu(cliente))		⟶ secu(parPC)
estaClie?	: cliente c × conj(parPC) ps		⟶ bool
		{¬ vacío(ps)}	
damePromesa	: cliente c × conj(parPC) ps		⟶ promesa
		{estaClie?(c,ps)}	
ordenarPorAcciones	: dicc(cliente,nat)		⟶ secu(cliente)
máxAcc	: dicc(cliente,nat) dc		⟶ cliente
		{¬ vacío(dc)}	
máxAccAux	: conj(cliente) cs × dicc(cliente,nat) dc		⟶ tupla(cliente,nat)
		{¬ vacío(dc) ∧$_L$ cs ⊆ claves(dc)}	
totalAccClie	: conj(parPC) vs × wolfie w		⟶ dicc(cliente,nat)
totalAccClieAux	: conj(parPC) cs × conj(parPC) vs × wolfie w		⟶ dicc(cliente,nat)
accTotalClie	: cliente c × wolfie w		⟶ nat
		{c ∈ clientes(w)}	
accTotalClieAux	: cliente c × conj(título) ts × wolfie w		⟶ nat
		{c ∈ clientes(w) ∧ ts ⊆ títulos(w)}	
accVendidasClie	: conj(parPC) ps × cliente c		⟶ nat

axiomas ∀cs: conj(cliente) ∀w: wolfie ∀t: título ∀$t1, t2$: nombre ∀c, cl: cliente ∀p: promesa ∀cot: nat ∀ts: conj(título) ∀ps: conj(promesa) ∀$disp$: nat ∀tp: tipoPromesa

clientes(inaugurarWolfie(cs))	≡ cs
clientes(agregarTítulo(t, w))	≡ clientes(w)
clientes(actualizarCotización(t2, cot, w))	≡ clientes(w)
clientes(agregarPromesa(c, p, w))	≡ clientes(w)
títulos(inaugurarWolfie(cs))	≡ ∅
títulos(agregarTítulo(t, w))	≡ Ag(t, títulos(w))
títulos(actualizarCotización(t1, cot, w))	≡ cambiarCotización(t1,c,títulos(w))
títulos(agregarPromesa(c, p, w))	≡ títulos(w)
promesasDe(c, inaugurarWolfie(cs))	≡ ∅
promesasDe(c, agregarTítulo(t, w))	≡ promesasDe(c, w)
promesasDe(c, agregarPromesa(cl, p, w))	≡ promesasDe(c,w) ∪ **if** c = cl **then** {p} **else** ∅ **fi**

```
promesasDe(c, actualizarCotización(t1, cot, w))  ≡ promesasDe(c, w) -
                                                   promesasAEjecutarPorCliente(t1, cot, c, w)

enAlza(t1,w)                  ≡ títuloEnAlza(t1,títulos(w))

accDisponibles(t1,w)          ≡ límiteTenencia(t1,títulos(w)) - tenenciasDeTodos(t1,clientes(w),w)

tenenciasDeTodos(t1,cs,w)     ≡ if vacío?(cs) then
                                    0
                                else
                                    accionesPorCliente(dameUno(cs),t1,w) +
                                    tenenciasDeTodos(t1,sinUno(cs),w)
                                fi

promesasAEjecutarPorCliente(t1,cot,c,w)  ≡ filtrarPorCliente(c, (ventasAEjecutar(t1,cot,w) ∪
                                                                 comprasAEjecutar(t1,cot,w)) )

filtrarPorCliente(c, ps)      ≡ if vacío?(ps) then
                                    ∅
                                else
                                    filtrarPorCliente(c,sinUno(ps)) ∪
                                    (if Π2(dameUno(ps)) = c then {Π1(dameUno(ps))} else ∅ fi)
                                fi

ventasAEjecutar(t1,cot,w)     ≡ promesasAEjecutar(cot,0,conjASecu(aparearPC(t1,venta,clientes(w),w)))

comprasAEjecutar(t1,cot,w)    ≡ promesasAEjecutar(cot,
                                    (accDisponibles(t1,w) + accVendidas(ventasAEjecutar(t1,cot,w))),
                                    promesasOrdenAccClie(aparearPC(t1,compra,clientes(w),w),
                                                         ventasAEjecutar(t1,cot,w),w))

aparearPC(t1,tp,cs,w)         ≡ if vacío?(cs) then
                                    ∅
                                else
                                    aparearPC(t1,tp,sinUno(cs),w) ∪
                                    (if vacío?(promesasSobreTítulo(t1, tp, promesasDe(dameUno(cs),w)))
                                    then
                                        ∅
                                    else
                                        {<promesasSobreTítulo(t1,tp,promesasDe(dameUno(cs),w)),dameUno(cs)>}
                                    fi)
                                fi

promesasAEjecutar(cot,disp,pends)  ≡ if vacía?(pends) then
                                         ∅
                                     else
                                         if promesaEjecutable(Π1(prim(pends)),cot,disp) then
                                             {prim(pends)} ∪
                                             ( if tipo(Π1(prim(pends))) = compra then
                                                   promesasAEjecutar(cot,
                                                       disp - cantidad(Π1(prim(pends))),
                                                       fin(pends))
                                               else
                                                   promesasAEjecutar(cot,
                                                       disp + cantidad(Π1(dameUno(pends))),
                                                       fin(pends))
                                               fi )
                                         else
                                             promesasAEjecutar(cot, disp, fin(pends))
                                         fi
                                     fi
```

```
accVendidas(ps) ≡ if vacío?(ps) then
                    0
                  else
                    cantidad(Π1(dameUno(ps))) + accVendidas(sinUno(ps))
                  fi

promesasOrdenAccClie(cs,vs,w) ≡ ordenSecu(cs, ordenarPorAcciones(totalAccClie(vs,w)))

ordenSecu(ps,sc) ≡ if vacía?(sc) then
                     <>
                   else
                     if estaClie?(prim(sc),ps) then damePromesa(prim(sc),ps) • <> else <> fi
                     & ordenSecu(ps,fin(sc))
                   fi

estaClie?(c,ps) ≡ Π2(dameUno(ps)) = c ∨L estaClie?(c,sinUno(ps))

damePromesa(c,ps) ≡ if Π2(dameUno(ps)) = c then
                      Π1(dameUno(ps))
                    else
                      damePromesa(c,sinUno(ps))
                    fi

ordenarPorAcciones(dc) ≡ if vacío?(dc) then
                           vacío
                         else
                           máxAcc(dc) • ordenarPorAcciones(borrar(máxAcc(dc),dc))
                         fi

máxAcc(dc)          ≡ Π1(máxAccAux(claves(dc),dc))
máxAccAux(cs,dc)    ≡ if vacío(sinUno(cs)) then
                        <dameUno(cs), obtener(dameUno(cs),dc)>
                      else
                        if obtener(dameUno(cs),dc) > Π2(máxAccAux(sinUno(cs),dc)) then
                          <dameUno(cs), obtener(dameUno(cs),dc)>
                        else
                          máxAccAux(sinUno(cs),dc)
                        fi
                      fi

totalAccClie(vs,w)          ≡ totalAccClieAux(clientes(w),vs,w)
totalAccClieAux(cs,vs,w)    ≡ if vacío?(cs) then
                                vacío?
                              else
                                definir(dameUno(cs),
                                  (accTotalClie(dameUno(cs),w)+accVendidasClie(vs,dameUno(cs))),
                                  totalAccClieAux(sinUno(cs),vs,w))
                              fi

accTotalClie(c,w)           ≡ accTotalClieAux(c,títulos(w),w)
accTotalClieAux(c,ts,w)     ≡ if vacío?(ts) then
                                0
                              else
                                accionesPorCliente(c,dameUno(ts),w)+accTotalClieAux(c,sinUno(ts),w)
                              fi

accVendidasClie(ps,c) ≡ if vacío?(ps) then
                          0
                        else
                          accVendidasClie(c,sinUno(ps)) +
                          if c = Π2(dameUno(ps)) then cantidad(Π1(dameUno(ps))) else 0 fi
                        fi
```

accionesPorCliente(c, t1, agregarTítulo(t2, cot, w)) ≡ **if** t1 = t2 **then**
0
else
accionesPorCliente(c, t1, w)
fi

accionesPorCliente(c, t1, agregarPromesa(cl, p, w)) ≡ accionesPorCliente(c,t1,w)

accionesPorCliente(c,t1,actualizarCotización(t2,cot,w)) ≡ **if** t1 = t2 **then**
compraVenta(accionesPorCliente(c,t1,w),
promesasAEjecutarPorCliente(t1,cot,c,w))
else
accionesPorCliente(c,t1,w)
fi

Fin TAD