Notas TP

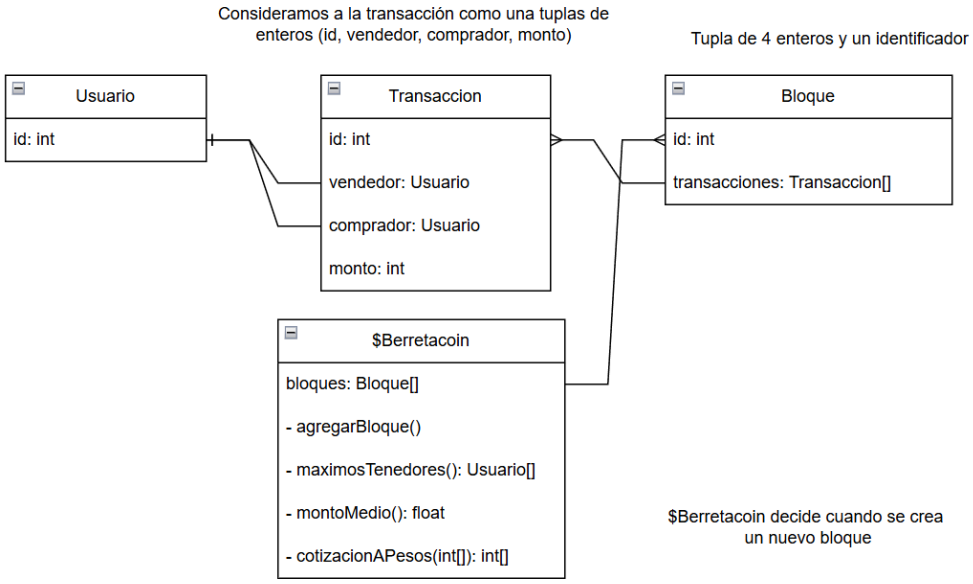
`usuario` : $\mathbb{Z}$

`transacción` : $(\mathbb{Z}, \mathbb{Z}, \mathbb{Z}, \mathbb{Z})$

Con transacción[1] y transacción[2] siendo ids de usuarios

`bloque` : $(\mathbb{Z}, \texttt{seq}\langle\texttt{transacción}\rangle)$

`berretaCoin` : `$BerretaCoin`



Usuarios

lu: (1, 5)

pato: (2, 10)

paiput: (3, 2)

bruno: (4, 7)

Transacciones

⟨id transaccion : Z, id comprador : Z, id vendedor : Z, monto : Z⟩

[

(1, 1, 2, 5) → lu le paga $5 a pato

(2, 3, 1, 1) → paiput le paga $1 a lu

(3, 1, 4, 2) → lu le paga $2 a bruno (pero esto rompe)

]

requiere {

$(\forall i : \mathbb{Z})$(0≤ i < |usuarios| $\rightarrow_L$ (

$(\forall k : \mathbb{Z})$(0≤ k < |transacciones| $\wedge$ (usuarios[i].id = transacciones[k].comprador $\vee$ usuarios[i].id = transacciones[k].vendedor) $\rightarrow_L$ (

usuarios[i].monto + *montoRecibido*(transacciones, usuarios[i].id, k) - *montoGastado*(transacciones, usuarios[i].id, k) ≥ 0

))

))

}

— auxiliar para calcular, para cada transacción (mirando todas las anteriores), el monto total recibido para un determinardo usuario

aux montoRecibido (in transacciones: seq, usuario: Z, indice: Z): Z =

$\sum_{j=0}^{indice} ifThenElse(transacciones[j].vendedor = usuario,\ transacciones[j].monto,\ 0)$

— auxiliar para calcular, para cada transacción (mirando todas las anteriores), el monto total gastado para un determinardo usuario

aux montoGastado (in transacciones: seq, usuario: Z, indice: Z): Z =

$\sum_{j=0}^{indice} ifThenElse(transacciones[j].comprador = usuario,\ transacciones[j].monto,\ 0)$

Cositas que faltan

- [x] ~~Validar que solo haya una transaccion con id_comprador 0 <-> la cantidad de B_0.bloques ≤ 3000 y que sea la primer transacción~~
- [x] ~~Validar que no haya ningun transaccion con id_comprador 0 si la cantidad de B_0.bloques > 3000~~
- [ ] Terminar de confirmar si faltan requieres (Luana)
- [ ] Los asegura
- [ ] corregir maximos tenedores (está mal el “para todo u” perteneciente a una secuencia. (Hay que indexar)

```
proc maximosTenedores (in berretacoin: Berretacoin) : seq⟨usuario⟩ {
        asegura {(∀u : berretacoin.usuarios) ((u ∈ res) ⟺ (esMaximoTenedor(berretacoin.usuarios, u)))}
}
```

proc agregarBloque(inout b: BerretaCoin, in transacciones: seq<transaccion>) {

requiere { b = B_0 }

requiere { |transacciones| ≤ 50 }

requiere {

($\forall i : \mathbb{Z}$)(0 ≤ i < |transacciones| $\rightarrow_L$ transacciones[i].id_vendedor ≠ transacciones[i].id_comprador)

}

requiere {

transacciones[0].id = 0 $\wedge_L$

($\forall i : \mathbb{Z}$)(0 ≤ i < |transacciones| - 1 $\rightarrow_L$ transacciones[i].id + 1 = transacciones[i+1].id)

}

requiere {

($\forall i : \mathbb{Z}$)(0≤ i < |B_0.usuarios| $\rightarrow_L$ (

($\forall k : \mathbb{Z}$)(0≤ k < |transacciones| $\wedge$ (B_0.usuarios[i].id = transacciones[k].comprador $\vee$ B_0.usuarios[i].id = transacciones[k].vendedor) $\rightarrow_L$ (

B_0.usuarios[i].monto + *montoRecibido*(transacciones, B_0.usuarios[i].id, k) - *montoGastado*(transacciones, B_0.usuarios[i].id, k) ≥ 0

))

))

}

— necesitamos validar que que si aparece un comprador que no estaba en la lista de usuarios, apareció antes como vendedor en las transacciones

requiere {

($\forall i : \mathbb{Z}$)(0≤ i < |transacciones| $\rightarrow_L$ (

¬(transacciones[i].comprador ∈ B_0.usuarios) $\rightarrow_L$ (

($\exists j : \mathbb{Z}$)(0≤ j < i $\rightarrow_L$ transacciones[i].comprador = transacciones[j].vendedor)

)

))

}

— Si la cantidad de bloques es menor a 3000, pedimos que |transacciones| sea por lo menos 1 (para asegurar que tiene la transacción de “agregado” de una coin en circulación

requiere {

((|B_0.bloques| ≤ 3000 ∧ |transacciones| > 0 ∧ transacciones[0].id = 0) ∧ (∀i : $\mathbb{Z}$)(1 ≤ i < |transacciones| $\rightarrow_L$ transacciones[i].id ≠ 0)) ∨

(|B_0.bloques| > 3000 ∧ (∀i : $\mathbb{Z}$)(0 ≤ i < |transacciones| $\rightarrow_L$ transacciones[i].id ≠ 0))

}

aux montoRecibido (in transacciones: seq, usuario: Z, indice: Z): Z =

$\sum_{j=0}^{indice} ifThenElse(transacciones[j].vendedor = usuario,\ transacciones[j].monto,\ 0)$

aux montoGastado (in transacciones: seq, usuario: Z, indice: Z): Z =

$\sum_{j=0}^{indice} ifThenElse(transacciones[j].comprador = usuario,\ transacciones[j].monto,\ 0)$

}