

# 1. Especificación

## 1.1. observadores

TAD Berretacoin  
obs blockchain: seq(Bloque)  
obs saldos: dict(idUsuario:  $\mathbb{Z}$ , dinero:  $\mathbb{Z}$ )  
obs totalCreado:  $\mathbb{Z}$

## 1.2. agregarBloque

```
proc agregarBloque (inout cadena: Berretacoin, in bloque: Bloque)
  requiere {bloque.id = ultimoBloque(cadena.blockchain).id + 1}
  requiere {bloqueValido(bloque)}
  requiere {tieneTransaccionDeCreacion(bloque)  $\rightarrow_L$  totalCreado < 3000}
  requiere {( $\forall i : \mathbb{Z}$ ) ((0  $\leq$  i < |bloque.transacciones|)  $\wedge$   $\neg$ esCreativa(bloque.transacciones[i])
 $\rightarrow_L$  (montoValido(bloque.transacciones[i]))}}
  requiere {cadena = C0}
  asegura {cadena.blockchain = C0.blockchain + bloque}
  asegura {( $\forall i : \mathbb{Z}$ ) ((0  $\leq$  i < |bloque.transacciones|)  $\wedge$  esCreativa(bloque.transacciones[i])
 $\rightarrow_L$  cadena.saldos[getIdVendedor(bloque.transacciones[i])] =
C0.saldos[getIdVendedor(bloque.transacciones[i])] + getMonto(bloque.transacciones[i]))}}
  asegura {( $\forall i : \mathbb{Z}$ ) ((0  $\leq$  i < |bloque.transacciones|)  $\wedge$  esCreativa(bloque.transacciones[i])
 $\rightarrow_L$ 
cadena.saldos[getIdVendedor(bloque.transacciones[i])] =
C0.saldos[getIdVendedor(bloque.transacciones[i])] + getMonto(bloque.transacciones[i])
 $\wedge$ 
cadena.saldos[getIdComprador(bloque.transacciones[i])] =
C0.saldos[getIdComprador(bloque.transacciones[i])] - getMonto(bloque.transacciones[i]))}}
```

### auxiliares

```
aux ultimoBloque (blockchain: seq(Bloque)) : Bloque = blockchain[|blockchain| - 1];
pred montoValido (transaccion: Transaccion, cadena: Berretacoin) {
  transaccion.monto  $\leq$  cadena.saldos[transaccion.idComprador]
}
aux getIdVendedor (transaccion: Transaccion) :  $\mathbb{Z}$  = transaccion.idVendedor;
aux getIdComprador (transaccion: Transaccion) :  $\mathbb{Z}$  = transaccion.idComprador;
aux getMonto (transaccion: Transaccion) :  $\mathbb{Z}$  = transaccion.monto;
```

## 1.3. maximosTenedores

```
proc maximosTenedores (in cadena: Berretacoin) : seq( $\mathbb{Z}$ )
  requiere {True}
  asegura {( $\forall u : \mathbb{Z}$ ) ((0  $\leq$  u < |cadena.saldos|)  $\wedge$ 
(cadena.saldos[u].dinero = valorMaximo(cadena.saldos))  $\rightarrow_L$ 
concat(res, {u}))}}
```

### auxiliares

```
aux valorMaximo (diccionario: dict(k:  $\mathbb{Z}$ , v:  $\mathbb{Z}$ )) :  $\mathbb{Z}$  = ( $\forall i : \mathbb{Z}$ ) ((1  $\leq$  i < |diccionario|)(
if diccionario[i].v > diccionario[i - 1].v then res = diccionario[i].v else res = diccionario[i - 1].v fi));
```

## 1.4. montoMedio

```
proc montoMedio (in cadena: Berretacoin) :  $\mathbb{Z}$ 
  requiere {True}
  asegura {res =  $\frac{totalTransacciones(cadena)}{cantidadTransacciones(cadena)}$ }
```

### auxiliares

```
aux totalTransacciones (cadena: Berretacoin) :  $\mathbb{Z}$  =
 $\sum_{i=0}^{|cadena.blockchain|-1} \sum_{j=0}^{|cadena.blockchain[i].transacciones|-1} (getMonto(cadena.blockchain[i].transacciones[j]))$ ;
aux cantidadTransacciones (cadena: Berretacoin) :  $\mathbb{Z}$  =
 $\sum_{i=0}^{|cadena.blockchain|-1} \sum_{j=0}^{|cadena.blockchain[i].transacciones|-1} (1)$ ;
```

## 1.5. cotizacionAPesos

```
proc cotizacionAPesos (in cotizaciones: seq< $\mathbb{Z}$ >, in cadena: Berretacoin) : seq< $\mathbb{Z}$ >
  requiere {True}
  asegura {( $\forall i : \mathbb{Z}$ ) ((0  $\leq$  i < |cotizaciones|)  $\longrightarrow_L$  res[i] = cotizaciones[i] * totalTransaccion(cadena.blockchain[i]))}
```

### auxiliares

```
aux totalTransaccion (bloq: Bloque) :  $\mathbb{Z}$  =  $\sum_{i=0}^{|bloq.transacciones|-1} (getMonto(bloq.transacciones[i]))$ ;
```