Enunciados

Vivero

Queremos modelar el funcionamiento de un vivero. El vivero arranca su actividad sin ninguna planta y con un monto inicial de dinero.

Las plantas las compramos en un mayorista que nos vende la cantidad que deseemos pero solamente de a una especie por vez. Como vivimos en un país con inflación, cada vez que vamos a comprar tenemos un precio diferente para la misma planta. Para poder comprar plantas tenemos que tener suficiente dinero disponible, ya que el mayorista no acepta fiarnos.

A cada planta le ponemos un precio de venta por unidad. Ese precio tenemos que poder cambiarlo todas las veces que necesitemos. Para simplificar el problema, asumimos que las plantas las vendemos de a un ejemplar (cada venta involucra un solo ejemplar de una única especie). Por supuesto que para poder hacer una venta tenemos que tener *stock* de esa planta y tenemos que haberle fijado un precio previamente. Además, se quiere saber en todo momento cuál es el balance de caja, es decir, el dinero que tiene disponible el vivero.

a) Indique las operaciones (procs) del TAD con todos sus parámetros.

b) Describa el TAD en forma completa, indicando sus observadores, los requiere y asegura de las operaciones. Puede agregar los predicados y funciones auxiliares que necesite, con su correspondiente definición.

c) ¿Qué cambiaría si supiéramos a priori que cada vez que compramos en el mayorista pagamos exactamente el 10 % más que la vez anterior? Describa los cambios en palabras.

Centro Ecuestre

Queremos modelar el funcionamiento de un centro ecuestre. El centro puede comprar caballos de diferentes razas. Al comprar un caballo, el vendedor nos indica cuántos "puntos de vigor" posee el caballo.

Es posible aumentar el vigor de los caballos entrenándolos. Cada entrenamiento dura un cierto tiempo y el caballo gana un punto de vigor por cada hora entrenada.

Luego, a los caballos se los hace competir en el Hipódromo de Palermo. En cada competencia el caballo puede ganar o perder. Si gana, aumenta 10 puntos de vigor. Si pierde, por la deshonra de la derrota, pierde 10 puntos.

Se quiere conocer en todo momento cuántos puntos de vigor tiene cada caballo y cuál es el mejor caballo de cada raza, siendo el mejor aquel que tiene más vigor. Si en alguna raza hay más de un caballo con el vigor máximo, alcanza con concer a cualquiera de ellos.

a) Indique las operaciones (procs) del TAD con todos sus parámetros.

b) Describa el TAD en forma completa, indicando sus observadores, los requiere y asegura de las operaciones. Puede agregar los predicados y funciones auxiliares que necesite, con su correspondiente definición.

c) Supongamos ahora que, luego del primer entrenamiento, los entrenamientos sucesivos para un mismo caballo van perdiendo efectividad: si a un caballo se lo entrena muchas veces, con cada entrenamiento el caballo recibe un 10 % menos de puntos de vigor que en el entrenamiento anterior. ¿qué cambios debería realizar en su TAD? Descríbalos con palabras.

Soluciones

Vivero

Planta ES $\mathbb{Z}$, Dinero ES $\mathbb{R}$

TAD Vivero {

obs balance: $Dinero$

obs stock: dict$\langle Planta, \text{struct}\langle cantidad : \mathbb{Z}, precio : Dinero\rangle\rangle$

proc abrirVivero (in $montoInicial$: Dinero) : Vivero {

requiere $\{montoInicial > 0\}$

asegura $\{res.balance = montoIncial\}$

asegura $\{res.stock = \{\}\}$

}

proc compraMayorista (inout v: Vivero, in p: Planta, in $cantidad$: $\mathbb{Z}$, in $precio$: Dinero) {

requiere $\{v = V_0\}$

requiere $\{cantidad > 0\}$

requiere $\{precio > 0\}$

requiere $\{v.balance >= cantidad \times precio\}$

asegura $\{v.balance = V_0.balance - cantidad \times precio\}$

asegura {

$\big(p \notin V_0.stock \land v.stock = setKey(V_0.stock, p, \langle cantidad, precio\rangle)\big) \lor$

$\big(p \in V_0.stock \land v.stock = setKey(V_0.stock, p, \langle V_0.stock[p].cantidad + cantidad, V_0.stock[p].precio\rangle)\big)$[1]

}

}

proc ponerPrecio (inout v: Vivero, in p: Planta, in $precio$: Dinero) {

requiere $\{v = V_0\}$

requiere $\{p \in v.stock\}$

requiere $\{precio > 0\}$

asegura $\{v.balance = V_0.balance\}$

asegura $\{v.stock = setKey(V_0.stock, p, \langle V_0.stock[p].cantidad, precio\rangle)\}$

}

proc vender (inout v: Vivero, in p: Planta) {

requiere $\{v = V_0\}$

requiere $\{p \in v.stock \land_L v.stock[p] > 0\}$

requiere $\{v.stock[p].cantidad > 0\}$

asegura $\{v.balance = V_0.balance + V_0.stock[p].precio\}$

asegura $\{v.stock = setKey(V_0.stock, p, \langle V_0.stock[p].cantidad - 1, V_0.stock[p].precio\rangle)\}$

}

proc balance (in v: Vivero) : Dinero {

requiere $\{True\}$

asegura $\{res = v.balance\}$

}

}

[1] Alternativa equivalente que hicimos en clase a la noche:

asegura {

$v.stock = setKey\big(V_0.stock, p,$

$\text{IfThenElse}(p \in V_0.stock,$

$\langle V_0.stock[p].cantidad + cantidad, V_0.stock[p].precio\rangle,$

$\langle cantidad, precio\rangle)$

$)$

}

Centro Ecuestre

Caballo, Raza, Vigor `ES` $\mathbb{Z}$

`TAD CentroEcuestre` {

`obs vigorCaballos:` dict$\langle Caballo, Vigor\rangle$
`obs caballosPorRaza:` dict$\langle Raza,$ conj$\langle Caballo\rangle\rangle$

`proc abrirCentro` () : CentroEcuestre {
`requiere` $\{True\}$
`asegura` $\{res.vigorCaballos = \{\} \land res.caballosPorRaza = \{\}\}$
}

`proc comprarCaballo` (inout ce: CentroEcuestre, in nc: Caballo, in r: Raza, in v: Vigor) {
`// Decido que el vigor puede empezar negativo, no hay restricciones sobre` v
`requiere` $\{nc \notin ce.vigorCaballos\}$
`requiere` $\{ce = CE_0\}$
`asegura` $\{ce.vigorCaballos = setKey(CE_0.vigorCaballos, nc, v)\}$
`asegura` $\{($
$(r \in CE_0.caballosPorRaza \land ce.caballosPorRaza =$
$setKey(CE_0.caballosPorRaza, r, CE_0.caballosPorRaza[r] \cup \{nc\})) \lor$
$(r \notin CE_0.caballosPorRaza \land ce.caballosPorRaza = setKey(CE_0.caballosPorRaza, r, \{nc\}))$
$)\}$
}

`proc entrenarCaballo` (inout ce: CentroEcuestre, in c: Caballo, in $horas$: $\mathbb{Z}$) {
`requiere` $\{c \in ce.vigorCaballos \land horas \geq 1\}$
`requiere` $\{ce = CE_0\}$
`asegura` $\{ce.caballosPorRaza = CE_0.caballosPorRaza\}$
`asegura` $\{ce.vigorCaballos = setKey(CE_0.vigorCaballos, c, CE_0.vigorCaballos[c] + horas)\}$
}

`proc competir` (inout ce: CentroEcuestre, in c: Caballo, in $gano$: Bool) {
`requiere` $\{c \in ce.vigorCaballos\}$
`requiere` $\{ce = CE_0\}$
`asegura` $\{ce.caballosPorRaza = CE_0.caballosPorRaza\}$
`asegura` $\{ce.vigorCaballos = setKey\big(CE_0.vigorCaballos, c, CE_0.vigorCaballos[c] + \mathsf{IfThenElse}(gano, 10, -10)\big)\}$
}

`proc puntosDeVigor` (in ce: CentroEcuestre, in c: Caballo) : Vigor {
`requiere` $\{c \in ce.vigorCaballos\}$
`asegura` $\{res = ce.vigorCaballos[c]\}$
}

`proc masVigorosoRaza` (in ce: CentroEcuestre, in r: Raza) : Caballo {
`requiere` $\{r \in ce.caballosPorRaza\}$
`asegura` $\{res \in ce.caballosPorRaza[r] \land$
$(\forall c : Caballo)(c \in ce.caballosPorRaza[r] \rightarrow_L ce.vigorCaballos[c] \leq ce.vigorCaballos[res])$
}
}

}