

UTN – FRBA – Paradigmas de Programación

Ejercicios – Objetos – guía 2

Trabajo con clases

Ejercicio 1 – micros empresarios

ACME S.A. tiene una planta modelo en una bucólica zona rural lejos del tráfico¹ urbano.

Para que la gente pueda llegar a la planta, la empresa tiene varios micros contratados. En cada micro entran n pasajeros sentados y m parados, donde el n y el m son particulares de cada micro (no son todos los micros iguales).

La gente no es toda igual, entonces para subirse a un micro se fija en distintas cosas:

- los apurados se suben siempre
- los claustrofóbicos se suben sólo si el micro tiene más de 120 m³ de volumen (se sabe el volumen de cada micro)
- los fiacas se suben sólo si entran sentados
- los moderados se suben sólo si quedan al menos x lugares libres (no importa si sentados o parados), donde el x es particular de cada persona moderada.
- los obsecuentes toman la misma decisión que tomaría su jefe (de cada empleado se sabe quién es su jefe, que es otro empleado).

Modelar a los micros y las personas de forma tal de

- poder preguntarle a un micro si se puede subir a una persona, para lo cual tienen que darse dos condiciones: que haya lugar en el micro, y que la persona acepte ir en el micro.
- hacer que se suba una persona a un micro, si no puede, que tire error, donde "error" es `self error: 'mensaje de error como String'`
- hacer que se baje una persona de un micro, si no se puede (porque está vacío), que tire error
- poder preguntarle a un micro quién fue el primero que se subió, `nil` si está vacío. OJO si en el micro hay una sola persona y se baja.

Ejercicio 2 – ¿dónde está la moneda?

Implementar el juego en el que una persona tiene tres sombreros, pone una moneda debajo de uno de ellos, los va moviendo para marearte, y después te pregunta ¿dónde está la moneda?, vos apostás, señalás un sombrero, si está ahí te llevás el doble de lo apostado, y si no ... seguí participando.

El juego debe entender el mensaje `apostar:al:`, donde el primer argumento es la cantidad de plata, y el segundo es un número del 1 al 3. P.ej. si quiero apostarle 20 pesos al sombrero 3, en el workspace pongo

```
elJuegoQueCree apostar: 20 al: 3
```

¹ sí, existe la palabra, fijate en www.rae.es ¡tu página amiga!

que devuelva la cantidad de plata que me da el juego, que en este caso es 40 o 0.

Al inicializar el juego, pasarle la cantidad de plata con la que empieza. Después quiero saber en cualquier momento para un juego: cuánta plata tiene, cuántas veces ganó (o sea, no pagó nada), cuántas veces perdió (o sea, pagó), cuál fue el valor de la apuesta más alta (no importa si ganó o perdió).

Existe la clase `Random` a cuyas instancias les puedo enviar el mensaje `next`, que me devuelve un número "al azar" entre 0 y 1; los números entienden `truncated` que te devuelve la parte entera (prueben `4.99 truncated`).

Ejercicio 3 – atención de animales

En un campo hay que atender a los animales, que tienen varias necesidades. Consideremos vacas, gallinas y cerdos, que tienen estas características.

Vaca

- Cuando come aumenta el peso en lo que comió / 3 y le da sed.
- Cuando bebe se le va la sed y pierde 500 g de peso.
- Conviene vacunarla una vez, o sea, si no se la vacunó conviene vacunarla, y si ya se la vacunó no conviene volverla a vacunar.
- Tiene hambre si pesa menos de 200 kg.
- Cada tanto se la lleva a caminar, en cada caminata pierde 3 kg.

Cerdo

- Cuando come aumenta el peso en lo que comió – 200 g (si come menos de 200 g no aumenta nada); si come más de 1 kg se le va el hambre, si no no.
- Quiero saber, para cada cerdo, cuánto comió la vez que más comió.
- Siempre conviene vacunarlo.
- Cuando bebe se le va la sed, y le da hambre.
- Si come más de tres veces sin beber le da sed.

Gallina

- Cuando come no se observa ningún cambio, siempre pesa 4 kg.
- Siempre tiene hambre, nunca tiene sed, nunca conviene vacunarla.
- Quiero saber, para una gallina, cuántas veces fue a comer.

Como se ve, importa cuánto come un animal cuando come (excepto para las gallinas), pero no cuánto bebe cuando bebe.

Hay varios dispositivos de atención automática a los animales:

1. Comederos normales: cada comedero da de comer una cantidad fija que varía para cada comedero, puede atender a los animales con hambre que pesen menos de lo que soporta el comedero, que también es un valor que depende del comedero.
Un comedero normal necesita recarga si le quedan menos de 10 raciones, cuando se lo recarga se le cargan 30 raciones.

2. Comederos inteligente: le dan de comer a un animal su peso / 100. Pueden atender a cualquier animal con hambre.
Un comedero inteligente necesita recarga si le quedan menos de 15 kg, al recargarlo se lo lleva hasta su capacidad máxima (que se indica para cada comedero).
3. Bebederos: dan de beber a un animal, pueden atender a los animales con sed.
Un bebedero necesita recarga cada 20 animales que atiende, lo que se le hace al recargarlo no se registra en el sistema (sí que se lo recarga para volver a contar desde ahí 20 animales atendidos).
4. Vacunatorios: vacunan a un animal, pueden atender a los animales que conviene vacunar.
Un vacunatorio necesita recarga si se queda sin vacunas, al atenderlo se le recargan 50 vacunas.

Una estación de servicio tiene 3 dispositivos de atención automática.

Modelar lo que se describió de forma tal de poder

- saber si un animal puede ser atendido por una estación de servicio; o sea, si se lo puede atender en alguno de sus dispositivos.
- indicar que un animal se atiende en una estación, en este caso se elige un dispositivo al azar que pueda atenderlo, y se lleva al animal a ese dispositivo para que lo atienda.
- saber para un cerdo cuánto comió la vez que más comió.
- recargar los dispositivos que necesitan recarga en una estación de servicio.

Ejercicio 4 – bolitas viajeras

ACME Playland Co. está planificando un nuevo juego para chicos de entre 6 y 11 años, que es un circuito para bolitas. En la caja vienen tramos de circuito y bolitas. Algunas bolitas son lisas, otras no; las bolitas tienen distinto peso.

Pensemos que el circuito tiene un solo sentido, entonces de cada tramo me van a interesar solamente el/las salida/s.

Los tramos pueden ser:

- tubos, de cada uno sé la longitud en cm, tienen una sola salida.
- desvío por lisa/no lisa, tiene dos salidas 1 y 2, si la bolita que entra es lisa sale por la 1, si no sale por la 2. El paso por el desvío no lleva tiempo.
- desvío por peso, tiene dos salidas 1 y 2, si la bolita pesa más de n gramos (el n es distinto para cada desvío) sale por la 1, si no sale por la 2. El paso por el desvío no lleva tiempo.
- detención, detiene a la bolita n segundos (el n es distinto para cada tramo detención), tiene una sola salida.
- acelerador, detiene a la bolita 10 segundos y aumenta su velocidad en 3 m/s, tiene una sola salida, toma 2 segundos.

ACME quiere hacer una simulación en la cual: se pueden armar circuitos, mostrar por dónde pasan bolitas, y cuánto tardan. La idea de la simulación es: vos tirás una bolita en un tramo indicando una velocidad, y tenés un botón "next" que la va haciendo avanzar de a un tramo.

P.ej. enchufó un túnel de 20 cm, un desvío por peso de 10 g, salida 1 del desvío a una detención de 5 seg. y de ahí a un túnel de 10 cm, salida 2 del desvío a un acelerador y de ahí a un túnel de 30 cm.

Tiro una bolita de 8 g al túnel inicial a 2 m/seg

- el primer "next" va al desvío, tardó 5 seg
- el segundo "next" va al acelerador,
- el tercer "next" va al túnel de 30 cm, se agregan 2 seg que es lo que se demora en el acelerador, ahora la bolita va a 5 m/seg.
- el cuarto "next" le agrega 6 seg (tiempo total: 13 seg), la bolita llega al final.
- a partir de acá, los "next" no producen ningún cambio.

Modelar el juego tal que pueda

- crear tramos y engancharlos para crear circuitos.
- lanzar una bolita por un tramo a una velocidad.
- poder preguntarle a un tramo si está bien configurado, los que tienen una sola salida siempre están bien configurados (si no tienen siguiente son fin de circuito), los que tienen dos deben tener configurados sus dos tramos siguientes.
- hacer avanzar a una bolita un paso, pasa al tramo siguiente que le corresponde.
Si el tramo donde está está mal configurado que se quede donde está.
Si el tramo no tiene siguiente debe afectar a la bolita (aumentar tiempo, cambiar velocidad) y la bolita debe darse cuenta que "llegó al final", una vez que "llegó al final" si la hago avanzar no hace nada.
- preguntarle a una bolita en cualquier momento: hace cuántos segundos que está viajando (haciendo la cuenta de cuánto tiempo estuvo en cada tramo, si llegó al final (o sea, está en un tramo fin de circuito), su velocidad actual.

En esta simulación loca sí vale mandar varias bolitas por el mismo lugar, no se traban ni nada.

Ejercicio 5 – porcentaje

Implementar la clase Porcentaje, de forma tal que al evaluar este código

```
p := Porcentaje new.  
p valor: 15.  
^p aplicarA: 2000
```

se obtenga 300, que es el 15% de 2000.

Ejercicio 6 – distancias que se suman

Implementar en Smalltalk lo que haga falta para que pueda escribir literalmente

```
10 kilometros + 45 metros
```

y que me devuelva un objeto que represente 10045 metros.

Ayuda: si agregás un método a la clase Number, todos los números van a entender ese mensaje.