Guía 5: Clases
Programación con Objetos 1
Versión del 10/08/2018

En estos ejercicios vamos a trabajar con clases. O sea, en lugar de definir el comportamiento de cada objeto por separado, vamos a definir clases, y a crear cada objeto que necesitemos a partir de una clase.

Ejercicio 1: Muchos contadores

Transformar el objeto contador (guía 1, ejercicios 1 y 2) en clase. Probar en el REPL con dos contadores, darle a uno tres `inc()` y al otro dos `inc()` seguido de un `dec()`, y verificar que el `valorActual()` y el `ultimoComando()` de cada uno sean los esperados.

Después, hacer un test que haga estas verificaciones automáticamente. Y dibujar el grafo de objetos correspondiente a los dos contadores, en el estado en que están al final del test.

Ejercicio 2: Muchas golondrinas

Transformar a Pepita (guía 1, ejercicios 3 y 4, y guía 2, ejercicio 2), Pepón y Pipa (guía 2, ejercicio 6) en las clases Golondrina, Gorrion y Paloma. Agregar al objeto roque que puede entrenar una colección de aves, y la consulta a cada ave sobre si puede volar o no unos kilómetros (guía 4, ejercicio 1).

Aclaración: **no incluir** la dieta de las aves (guía 2, ejercicio 5), o sea, el parámetro de comer es un número que representa una cantidad de gramos de alpiste.

Armar un test que incluya:

- las golondrinas pepita y juanita,
- los gorriones pepón, león y agamenón,
- la paloma pipa.

Darle de comer 100 a cada bicho. Después, pasarle a Roque para que entrene a pepita, juanita, leon y pipa.

Hacer que roque también entrene a una "paloma sin nombre". Para esto poner en el test `roque.agregarParaEntrenar(new Paloma())`

Después de esto, correr un ciclo de entrenamiento de Roque.

Finalmente, hacer el assert sobre cada una de las seis aves "con nombre", la energía de golondrinas y gorriones, los km recorridos de pipa.

Lograr que el test dé verde, y dibujar el grafo de objetos al final del test.

Ejercicio 3: Remisería

Hacer un modelo de remiserías. Vamos a trabajar con la flota de vehículos de cada remisería.

De cada vehículo nos van a interesar: capacidad (o sea, cantidad de pasajeros que pueden transportar), velocidad máxima, color y peso.

Considerar estos vehículos:

- Chevrolet Corsa, cuya capacidad es 4 pasajeros, su velocidad máxima 150 km/h, y pesan 1300 kg. De cada Corsa hay que informar de qué color es.
- Autos standard que funcionan a gas. Cada uno de ellos puede tener puesto, o no, un tanque adicional. La capacidad de un auto a gas es de 4 pasajeros si no tiene puesto el tanque adicional, o 3 pasajeros si sí lo tiene puesto. La velocidad máxima es 120 km/h con, o 110 km/h sin, tanque adicional. El peso es 1200 kg, más 150 kg adicionales si tiene tanque adicional. Todos los autos a gas son azules.
- Una Trafic (sí, una sola), que es reconfigurable, porque se le puede cambiar el interior y el motor. La capacidad de la Trafic es la de su interior, la velocidad máxima la que indique su motor, el peso es 4000 kg más lo que pesen interior y motor, y es de color blanco.

  Prever dos interiores, uno cómodo (capacidad 5, peso 700 kg) y otro popular (capacidad 12, peso 1000 kg). Y dos motores, uno pulenta (velocidad máxima 130 km/h, peso 800 kg) y otro batatón (velocidad máxima 80 km/h, peso 500 kg).
- Autos que son todos distintos entre sí, de cada uno hay que especificar capacidad, velocidad máxima, peso y color.

Cada remisería debe aceptar los siguientes mensajes:

- `agregarAFlota(vehiculo)` y `quitarDeFlota(vehiculo)`.
- `pesoTotalFlota()`, la suma del peso de cada vehículo agregado a la flota.
- `esRecomendable()`, es verdadero si la remisería tiene al menos 3 vehículos, y además, *todos* los vehículos registrados en la remisería pueden ir al menos a 100 km/h.
- `capacidadTotalYendoA(velocidad)`, la cantidad total de personas que puede transportar la remisería, considerando solamente los autos de su flota cuya velocidad máxima sea mayor o igual a la velocidad indicada.
- `colorDelAutoMasRapido()`, eso.

Se pide

a) Implementar las clases y objetos que hagan falta para modelar autos y remiserías según lo que se describió.

b) Armar un test en el que hay dos remiserías.

   En el test, definir un auto llamado cachito, (o sea, poner `var cachito = ...`), que sea un Corsa rojo.

   Los autos de la primer remisería son: cachito; dos Corsa más, uno negro y el otro verde; un auto standard a gas con el tanque adicional puesto; y un auto "distinto" que tiene: capacidad 5, velocidad máxima 160 km/h, peso 1200 kg, color beige.

   Los autos de la segunda remisería son: cachito; tres autos standard a gas, uno con el tanque adicional puesto y dos sin, y la Trafic, configurada con el interior cómodo y el motor batatón.

   Sí, cachito está en la flota de las dos remiserías, acepta encargos de las dos.

   Hacer asserts sobre cada una, respecto del peso total, si es recomendable o no, la capacidad total yendo a 140 km/h, y el color del auto más rápido.

c) Hacer el dibujo del grafo de objetos generado por este test.

d) Agregar al modelo los viajes. De cada viaje nos interesa: los kilómetros, el tiempo máximo de viaje en horas, la cantidad de pasajeros, y también un conjunto de colores incompatibles, o sea, que los pasajeros rechazan hacer el viaje en autos de esos colores.

   Agregar la capacidad de preguntar si un auto puede hacer un viaje, enviándole un mensaje al viaje, claro, con el auto como parámetro. Para que un auto pueda hacer un viaje se tienen que dar tres condiciones: que la velocidad máxima sea al menos 10 km/h mayor a la velocidad promedio que necesita el viaje (que es kilómetros dividido tiempo máximo); que la capacidad del auto dé para la cantidad de pasajeros del viaje; y que el auto no sea de un color incompatible para el viaje.

   Usando esto, agregarle a las remiserías un método para que puedan responder qué autos pueden hacer un viaje.

e) Agregar el registro de los viajes hechos por cada remisería. Para esto, agregarle a la clase que define el comportamiento de las remiserías, el método `registrarViaje(viaje,auto)`, que agregue el viaje a una colección de viajes hechos, y le asigne el auto al viaje. Agregarle al viaje un atributo que sea el auto que lo hizo.

   Nota: no conviene que sea el auto quien se acuerde de los viajes que hizo, porque si un auto trabaja con dos remiserías, se complica distinguir qué viajes hizo con cada una.

   A partir de esto, agregar lo que haga falta para poder preguntarle a una remisería:

   - cuántos viajes hizo un auto (para esa remisería, claro).
   - cuántos viajes hizo la remisería de más de una cantidad de kilómetros.
   - cuántos lugares libres en total hubo en los viajes que hizo la remisería. P.ej. si la remisería hizo un viaje de 2 pasajeros usando un auto de capacidad 5, ese viaje tuvo 3 lugares libres.

- cuánto pagarle a un auto, de acuerdo a los viajes que hizo para esa remisería. Cada remisería establece un valor por kilómetro, y un mínimo para cada viaje. Por ejemplo, si una remisería establece 3 pesos por km y 30 de mínimo por viaje, entonces un viaje de 7 km lo paga 30 pesos (porque 3 x 7 = 21 no llega a 30), y un viaje de 25 km lo paga 75 pesos (3 x 25 = 75 supera el mínimo de 30).

Ejercicio 4: Atención de animales

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

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, le da hambre, y pierde 1 kg de peso.
- 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 ración, que es 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 una cantidad indeterminada dispositivos de atención automática.

Se pide:

a) 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, mediante una pregunta a un objeto. ¿A cuál objeto?
   - 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. Esto mediante una orden a un objeto.
   - recargar los dispositivos que necesitan recarga en una estación de servicio, enviándole un mensaje a la estación de servicio.
   - saber para una estación de servicio: si un animal fue o no atendido, el conjunto de los animales atendidos en esa estación que conviene vacunar, el animal más pesado que atendió, y el peso total de los animales atendidos.

b) Armar un test con una vaca, un cerdo y una gallina, y un dispositivo de cada tipo. Hacer al menos 5 asserts sobre si un animal se puede o no atender en un dispositivo, tiene que haber al menos dos que sí y dos que no.

c) Armar un segundo test donde haya dos vacas, dos cerdos y una gallina. Armar dos estaciones de servicios, una con dos comederos normales, que soporten animales de hasta 70 y 110 kg respectivamente, y un bebedero, el otro con un comedero inteligente y un vacunatorio. Tiene que pasar que

- haya un animal que pueda atenderse en la primer estación y no en la segunda,
- haya un animal que pueda atenderse en la segunda estación y no en la primera,
- uno de los comederos necesite recarga,
- uno de los animales pueda atenderse en las dos estaciones, en cada una en un solo dispositivo. P.ej. una vaca de 120 kg, ya vacunada y con sed.

Hacer estos asserts, después hacer que el animal que puede atenderse en las dos estaciones se atienda en ambas, probar que quede como tiene que quedar. Mandar otros dos animales a atenderse, después hacer asserts sobre lo que se pide de las estaciones de servicio. Hacer el grafo de objetos correspondiente al final de este segundo test.

Ejercicio 5: Muchas apuestas de quiniela

Realizar una resolución del enunciado de la quiniela reducida (guía 4, ejercicio 5) usando clases para representar los distintos tipos de apuesta y los sorteos. De esta forma, vamos a poder tener muchas apuestas a la cabeza, muchas a los primeros 2, etc., en lugar de una sola de cada una. También se podrían tener muchos sorteos.

Sugerencia: como ahora las apuestas se tienen que crear, es fácil hacer que cada apuesta conozca a su sorteo. De esa forma, ya no hace falta pasar un sorteo como parámetro al preguntarle a una apuesta si es ganadora.

Otra sugerencia: encararlo de esta forma:

- Primero la parte 1. O sea, implementar un sorteo que solamente se le pueda pasar el resultado, y después preguntarle qué números salieron (o qué número salió en tal lugar). Que como total apostado devuelva p.ej. siempre 1000.
- Con esto, se pueden implementar las apuestas. O sea, la parte 3.
- Finalmente, agregarle al sorteo las apuestas, y resolver la parte 2.

Ejercicio 6: Infraestructura ferroviaria

Una administradora ferroviaria necesita una aplicación que le ayude a manejar las formaciones que tiene disponibles en distintos depósitos.

Una **formación** es lo que habitualmente llamamos "un tren", tiene una o varias *locomotoras*, y uno o varios *vagones*. Hay vagones de pasajeros y vagones de carga. En cada **depósito** hay: formaciones ya armadas, y locomotoras sueltas que pueden ser agregadas a una formación.

Vamos a resolverlo de a poco.

***Parte 1:** Vagones*
De cada vagón de pasajeros se conocen largo y ancho, en metros. La cantidad de pasajeros que puede transportar un vagón de pasajeros es:

- Si el ancho es de hasta 2.5 metros: metros de largo * 8.

- Si el ancho de más de 2.5 metros: metros de largo * 10.

P.ej., si tenemos dos vagones de pasajeros, los dos de 10 metros de largo, uno de 2 y el otro de 3 metros de ancho, entonces el primero puede llevar 80 pasajeros, y el segundo puede llevar 100. Un vagón de pasajeros puede llevar hasta 100 kilos de carga.

De cada vagón de carga se conoce la carga máxima que puede llevar, en kilos. Un vagón de carga no puede llevar ningún pasajero.

El peso máximo de un vagón, medido en kilos, se calcula así:

- Para un vagón de pasajeros: cantidad de pasajeros que puede llevar * 80, más los 100 kg de carga, más el peso del vagón vacío que es 200 kilos * metro de largo.
- Para un vagón de carga: la carga máxima que puede llevar + 160 (en cada vagón de carga van dos guardas), más el peso del vagón vacío que es 1500 kilos.

Modelar los vagones de carga y de pasajeros, y las formaciones (que por ahora sólo incluyen vagones), de manera de poder saber:

1. El total de pasajeros que puede transportar una formación.
2. Cuántos vagones livianos tiene una formación; un vagón es liviano si su peso máximo es menor a 2500 kg.
3. El vagón que más pasajeros puede transportar en una formación.
4. Si es una formación carguera, o sea, si cada uno de sus vagones puede llevar al menos 1500 kg de carga.
5. Si es una formación parejita, o sea, si la diferencia entre el mayor y el menor de los pesos máximos de los vagones no supera los 200 kg.

Armar la solución de forma tal que puedan aparecer otros tipos de vagón (vagones mixtos, vagones de pasajeros de dos pisos, coche comedor, etc.).

Parte 2: *Locomotoras*
De cada locomotora se sabe: su peso, el peso máximo que puede arrastrar, y su velocidad máxima. P.ej. puedo tener una locomotora que pesa 1000 kg, puede arrastrar hasta 12000 kg, y su velocidad máxima es de 80 km/h. Obviamente se tiene que arrastrar a ella misma, entonces no le puedo cargar 12000 kg de vagones, solamente 11000.

Lograr que las formaciones puedan incorporar locomotoras, de forma tal de poder saber:

1. La velocidad máxima de una formación, que es el mínimo entre las velocidades máximas de las locomotoras.
2. Si una formación es eficiente; es eficiente si cada una de sus locomotoras puede arrastrar, al menos, 5 veces su peso (el de la locomotora misma).

3. Si una formación puede moverse. Una formación puede moverse si el arrastre total de las locomotoras es mayor o igual al peso máximo de la formación, que es: peso de las locomotoras + peso máximo de los vagones.

4. Cuántos kilos de empuje le faltan a una formación para poder moverse, que es: 0 si ya se puede mover, y (peso máximo de la formación - arrastre de las locomotoras) en caso contrario.

¡Atención! No vale agregarle empuje a los vagones ni peso máximo a las locomotoras.

Parte 3: *Depósitos*
Agregar al modelo los depósitos ferroviarios. Recordemos que en cada depósito hay: formaciones ya armadas, y locomotoras sueltas que pueden ser agregadas a una formación.

Se pide resolver lo siguiente:

1. Dado un depósito, obtener el conjunto formado por el vagón más pesado de cada formación; se espera un conjunto de vagones.

2. Saber si un depósito necesita un conductor experimentado. Un depósito necesita un conductor experimentado si alguna de sus formaciones es compleja. Una formación es compleja si: tiene más de 20 unidades (sumando locomotoras y vagones), o el peso total (sumando locomotoras y vagones) es de más de 10000 kg.

3. Agregar, dentro de un depósito, una locomotora a una formación determinada, de forma tal que la formación pueda moverse. Si la formación ya puede moverse, entonces no se hace nada. Si no, se le agrega una locomotora suelta del depósito cuyo arrastre útil sea mayor o igual a los kilos de empuje que le faltan a la formación. Si no hay ninguna locomotora suelta que cumpla esta condición, no se hace nada.

Para el punto 2, indicar en castellano: a qué objeto se le pide lo que indica el enunciado y con qué mensaje, qué otros objetos colaboran en la resolución, y qué se le pide a cada uno.