# Guía 7 - Más sobre herencia
# Programación con Objetos 1
# 10/08/2018

### Ejercicio 1: Atención de animales

Agregar lo que sigue al ejercicio 4 de la guía 5, el de atención de animales.

**Nota**
Los agregados son solamente sobre el modelo de animales. O sea, no hace falta tener implementados los dispositivos de atención ni los centros de servicios, para hacer este ejercicio.

***Parte 1:*** *Variantes de animales*
A partir de los animales conocidos, o sea vacas, cerdos y gallinas, agregar estas variantes.

- *Cerdo resistente*: Desde que lo vacunan no conviene vacunarlo. Ahora, si después de recibir una vacuna come de una vez, 5 kg o más, ahí conviene vacunarlo de nuevo.
- *Vaca zen*: Una vez que se la vacuna, nunca más tiene sed.
- *Gallina turuleca*: Además de comer, nos interesa cuando pone huevos. Cada vez que pone huevos, pueden ser uno, dos o tres. Implementar esto como tres mensajes distintos, `haPuestoUnHuevo()`, `haPuestoDosHuevos()`, `haPuestoTresHuevos()`. Si el total de huevos que puso desde que nació es impar, entonces tiene sed, si es par, entonces no tiene sed.

***Parte 2:*** *Felicidad*
Agregar la capacidad, para cualquier animal, de saber si está feliz o no. Para que un animal esté feliz, tiene que pasar que no convenga vacunarlo, y además, que pese o bien menos de 10 kg, o bien más de 150 kg. Agregar también estas tres variantes nuevas de animales:

- *Vaca reflexiva*: para estar feliz, además, tuvo que haber salido a caminar al menos 2 veces. Para las vacas reflexivas (¡OJO! no para todas) hay que contar cuántas veces salió a caminar.
- *Vaca filósofa*: es como la reflexiva, salvo que para estar feliz, además de todas las condiciones que trae por ser reflexiva, tiene que no tener sed.
- *Cerdito alegre*: es como el cerdo normal (o sea, no como el cerdo resistente), y siempre está feliz. **No hay** ningún cerdo que sea ser feliz y también resistente.

**Ejercicio 2: Viajeros**
Modelar un conjunto de personas, y para cada persona, los viajes que hizo. Para cada viaje, nos interesa el país donde fue, y el año.

Hay que poder responder:

- En qué países estuvo una persona en un determinado año. Son: los países de los viajes que hizo la persona, más los países en donde residió en ese año.
  P.ej. `pepe.enQuePaisesEstuvo(2014)`.
- Dada una persona, si coincidió con otra en un determinado año, o sea, si hay al menos un país en el que los dos estuvieron en ese año.
  P.ej. `pepe.coincidioCon(ana,2014)`.

Los países donde residió una persona dependen de qué tipo de persona es. Considerar estos tipos de personas:

- *Establecido*: se sabe en qué país vive, en cualquier año que se pregunte, residió en ese país y en ningún otro.
- *Migrante*: se sabe en qué país nació, a qué país se mudó, y en qué año. Hasta el año antes de mudarse, residió en el país en el que nació. Después de mudarse, residió en el país al que se mudó. El año en que se mudó residió en los dos.
- *Doctor*: se sabe en qué país vive, en qué país hizo el doctorado, y entre qué años. P.ej. si Juan, que vive en Brasil, hizo el doctorado en Colombia entre 2008 y 2011, entonces residió en Colombia esos 4 años, y en Brasil hasta 2008 inclusive, y a partir de 2011 (para 2009 y 2010, residió todo el año en Colombia).
- *Menor*: se sabe quién es la madre. Reside, en cualquier año, en los mismos paises que la madre. OJO los viajes del menor son separados, si viajaron juntos, hay que cargar el viaje en los dos, solamente se considera que coinciden los países de residencia.

**Ejercicio 3: Naves espaciales**
Vamos a trabajar con un modelo de naves espaciales.

***Parte 1:*** *Modelo básico*
Lo que nos interesa de la nave es la velocidad y la dirección en la que se mueve respecto del Sol. La velocidad se mide en km/seg. La dirección es un número entre -10 y 10, si es 10 se acerca al Sol, si es -10 se aleja, si es 0 le está dando vueltas al Sol manteniendo siempre la misma distancia. Números entre 1 y 9 indican que rota en espiral alejándose, cuando más se acerca a 10, más se aleja. Lo mismo entre -1 y -9, pero para alejarse.

Las naves tienen que entender estos mensajes:

- `acelerar(cuanto)` y `desacelerar(cuanto)`, aumenta o disminuye su velocidad en el valor indicado. Tener en cuenta que una nave no puede ir a más de 100000 km/seg ni tampoco la velocidad puede ser negativa, controlar estos topes. Ayuda: para esto usar max y min.
- `irHaciaElSol()` y `escaparDelSol()`, cambia su dirección a 10 y -10 respectivamente.
- `alejarseUnPocoDelSol()` y `acercarseUnPocoAlSol()`, suma y resta 1 respectivamente a la dirección.

***Parte 2:*** *Reacción ante amenazas para distintos tipos de nave*
Agregar a las naves la capacidad de responder al mensaje `recibirAmenaza()`. Lo que tiene que hacer *cualquier* nave al recibir una amenaza es: primero escapar, y después avisar. Hay que implementar varios tipos de nave, para esto usar herencia. La idea es que la clase `NaveEspacial` quede abstracta.

Los tipos de naves son estos:

- *Naves-baliza*: se les agrega un color, que se puede manejar como un String. El color inicial es `"azul"`.

  Para una nave-baliza, *escapar* es ir hacia el Sol, y *avisar* es cambiar su color a `"rojo"`. Agregarle un mensaje adicional `volverALaNormalidad()` que cambie su color a `"azul"`.

- *Naves de pasajeros*: al modelo se agrega: cantidades de pasajeros, stock de raciones de comida, stock de raciones de bebida.

  Para una nave-baliza, *escapar* es duplicar su velocidad, y *avisar* es darle a cada pasajero una ración de comida y dos de bebida, para este modelo, eso se refleja en bajar el stock.

- *Naves de combate*: al modelo se agregan dos controles: si está invisible y si tiene los misiles desplegados. Agregar métodos `ponerseInvisible()`, `ponerseVisible()`, `desplegarMisiles()` y `replegarMisiles()`. También hay que mantener una lista con los mensajes que emite cada nave de combate, cada mensaje es un String. Para esto, agregar el método
  `emitirMensaje(textoDelMensaje)`. A una nave de combate se le tiene que poder preguntar: cuántos mensajes emitió, cuál fue el primer mensaje emitido, cuál fue el último mensaje emitido, si es escueta (o sea, si todos los mensajes que emitió son de menos de 30 caracteres), y si alguna vez emitió un determinado mensaje (p.ej. si alguna vez emitió `"Hola amigos"` o no.

  Para una nave de combate, *escapar* implica ejecutar dos veces la acción de acercarse un poco al Sol, y después ponerse invisible y desplegar sus misiles. Por su parte, *avisar* es emitir el mensaje `"Amenaza recibida"`.

***Parte 3:*** *Gobernación estelar*
Agregar al modelo las gobernaciones estelares. Cada gobernación maneja un conjunto de

unidades. Cada unidad puede ser, o bien una nave, o bien un depósito estelar. De cada depósito se sabe cuántos containers tiene depositados; tiene que entender el mensaje `recibirContainers(cuantos)`. Los depósitos estelares también tienen que ser capaces de `recibirAmenaza()`, en cuyo caso destruyen la mitad de los containers que tiene depositados; en este modelo sencillo, lo único que se registra es que su cantidad de containers baja a la mitad.

Una gobernación estelar también puede `recibirAmenaza()`, si pasa eso, le envía el mismo mensaje a cada una de sus unidades.

**Ejercicio 4: Empresa de transporte**
Se trata de modelar una empresa de transporte, en varias etapas.

***Parte 1:*** *Modelo inicial: vehículos, viajes, provincias, empresas*
El modelo inicial incluye:

a) *Provincias*: de cada provincia se conoce su nombre, que se maneja como un String, y si se vende o no GNC. Nada más.

b) *Rutas*: de cada ruta se conoce: cantidad total de km, de estos cuántos están en buen estado y cuántos en mal estado, y provincias por las que pasa. A una ruta le tengo que poder preguntar:

   - si está dañada, o sea, si tiene más km en mal estado que en buen estado.
   - si se puede hacer con GNC, o sea, si en todas las provincias por las que pasa se vende GNC.

c) *Vehículos*: de cada vehículo se registra para qué provincias está habilitado. Tienen que entender los mensajes `habilitar(prov)` e `inhabilitar(prov)`.

d) *Viajes*: de cada viaje se sabe: por qué ruta se hace, y la carga a transportar medida en kilos. A un viaje se le tiene que poder preguntar si es riesgoso, las condiciones son que la carga sea de 5000 kg o más, y que además la ruta esté dañada.

e) *Empresas*: Una empresa de transportes tiene varios vehículos. Dada una empresa, se quiere saber:

   - cuáles de sus vehículos pueden hacer un viaje. La condición para que un vehiculo pueda hacer un viaje es que esté habilitado para todas las provincias por las que pasa el viaje de acuerdo a su ruta.
   - si tiene cobertura total de una provincia dada, o sea, si todos sus vehículos están habilitados o no para esa provincia.

***Parte 2:*** *Distintos tipos de vehiculos*
Agregar al modelo de los vehículos: su carga máxima, el tiempo para realizar un viaje, y el consumo de combustible por kilómetro en buen y en mal estado. Para esto, manejar varios tipos de vehículos, dejando a *Vehiculo* como clase abstracta.

**Nota**:
Conviene armar esta segunda parte en un archivo separado, copiar el código de la parte 1 y modificar a gusto. Considerar

a) *Camionetas*: se comportan así

   - carga máxima: 1200 kg.
   - tiempo para realizar un viaje: (km buen estado / 120) + (km mal estado / 100).

b) *Combis*: cada combi puede tener, o no, capota. Se comportan así

   - carga máxima: 2500 kg, si tienen capota agregar 1000 kg más.
   - tiempo para realizar un viaje: km / 100 si la carga del viaje es hasta 1800 kg, km / 90 si es de más de 1800 kg.

c) *Camiones*: de cada camión se sabe cuántos ejes tiene. Se comportan así

   - carga máxima: 6000 kg por eje.
   - tiempo para realizar un viaje: 15 minutos fijo + (km buen estado / 100) + (km mal estado / n), donde n es 70 hasta 10000 kg, 60 por más de 10000 kg. Si tiene más de 2 ejes, sumar una hora más.

Agregar a los viajes un tiempo máximo, y a las condiciones para que un vehículo pueda hacer un viaje, que pueda llevar la carga del viaje, y que no tarde más que tiempo máximo indicado para el viaje. Para esto último, modificar los métodos escritos para la parte 1.

***Parte 3:*** *Peso máximo total*
Agregar a todos los vehículos la capacidad de preguntarles su peso máximo total, que es su tara más su carga máxima. La tara se indica específicamente para cada vehículo.

**Ejercicio 5: Tareas**
Se trata de modelar las tareas que debe realizar Acme S.R.L. para desarrollarse. Un objetivo del modelo es poder calcular el *costo total* de una tarea, que se calcula así:

costo total = (costo horario * cantidad de horas) + costo inicial + gastos de papeleo.

Los gastos de papeleo se establecen para cada tarea cuando se crea.

Considerar estos tipos de tarea

a) *Mantenimiento de equipos*: lo hace un técnico que se especializa en un único tipo de tarea (p.ej. mantener tornos eléctricos). De cada técnico se sabe cuánto tarda en hacer el mantenimiento de un equipo, y cuánto cobra por hora. Una misma tarea de mantenimiento de equipos puede involucrar varios equipos, todos iguales, para cada tarea se indica cuántos. Este tipo de tareas no tiene costo inicial.

   Ejemplo: mantenimiento de 10 impresoras, lo hace Juan, un técnico que tarda 2 horas por impresora, y cobra 100 pesos la hora. Esta tarea lleva 20 horas, el costo horario es de 100 pesos, y el costo inicial es 0.

b) Construcción: se calcula cuántos días se va a tardar, cuántas personas van a participar, y lo que cobra por hora cada persona (se supone que todas las personas asignadas a la misma tarea de construcción tienen el mismo valor horario). Para el total de horas que va a llevar la tarea, se calculan 8 horas por día. El costo inicial (que es lo que sale llevar los materiales hasta donde se hace la obra) se indica para cada tarea de construcción.

   Ejemplo: construcción de una pared, lleva 3 días de 4 personas, el valor horario es de 70 pesos por persona, lleva un costo inicial de 500 pesos. Esta tarea lleva 24 horas, el costo horario es de 280 pesos, y el costo inicial es el especificado, o sea 500 pesos.

c) Encuestas: consiste en encuestar a una cantidad de personas que viven en una misma ciudad. Se especifica: a cuántas personas se va a encuestar, y a qué distancia (en km) se encuentra la ciudad. Para calcular cuántas horas lleva la tarea, considerar que se hacen 3 encuestas por hora. El costo horario es de 180 pesos. El costo inicial es de 12 pesos por kilómetro, es lo que sale el remis.

   Ejemplo: hacer 21 encuestas en una ciudad que está a 200 km. Esta tarea lleva 7 horas, el costo horario es 180 pesos, y el costo inicial es 2400 pesos.

d) Rehacer una tarea que se hizo mal: se parte de una tarea. La cantidad de horas de rehacer una tarea es la misma que la tarea, más 5 horas que es lo que lleva verificar que esta vez se hizo bien; el costo horario es el mismo que el de la tarea; y el costo inicial es el mismo del de la tarea más 1500 pesos que es lo que sale la verificación.

   Ejemplo: rehacer las encuestas del punto anterior. Esta tarea lleva 12 horas, el costo horario es 180 pesos, y el costo inicial es 3900 pesos.

Además, se debe indicar para cada tarea si está hecha o no. Cuando se crea no está hecha, y todas las tareas deben entender el mensaje `registrarQueSeHizo()`.

Acme S.R.L. cuenta con un plantel de supervisores. Cada tarea se asigna a un supervisor. A cada supervisor se le asigna un presupuesto.

A partir del modelo hay que poder indicar, para un supervisor:

- el costo total de las tareas asignadas al supervisor que están hechas.
- si el supervisor necesita fondos adicionales. La condición es que el presupuesto no le alcance para el costo total de las tareas que tiene asignadas y que no están hechas.
- si el supervisor se especializa en grandes obras, o sea, si todas las tareas que tiene asignadas llevan, cada una, más de 40 horas.
- que terminó con todo, o sea, para cada tarea que tiene asignada, registrar que se hizo.
- el conjunto de las tareas que tiene asignadas, y cuyo costo horario sea mayor a un parámetro.