Guía 4: Ejercicios usando Colecciones
Programación con Objetos 1
Versión del 10/08/2018

Ejercicio 1: Entrenadores

Extender el ejercicio de pepita y su entrenamiento de la guía 3.

Parte 1: *Consultas sobre las aves*

Realizar las modificaciones necesarias para que Pepita, Pepon y Pipa entiendan el mensaje `puedeVolar(unosKms)`. El cual determina si un ave puede volar una cantidad determinada de kilómetros.

- Pepita: si su energía es $\geq 10 + unosKms$
- Pepon: si su energía es $\geq 1 + unosKms \times 0{,}5$
- Pipa: pipa siempre puede volar, no importa cuanto se le pida

¡Ojo con esto! No duplicar código

Parte 2: *Entrenar a más de un ave*

Modificar a Roque para que sea capaz de entrenar a varias aves. Para eso se debe cambiar su interfaz (conjunto de mensajes que entiende) para soportar:

- `agregarParaEntrenar(unAve)`: Roque agrega el ave a su colección interna de aves para entrenar.
- `dejarDeEntrenar(unAve)`: Elimina el ave de su colección
- Modificar el método `entrenar()` para que Roque entrene con la misma rutina a todas sus aves. **Nota:** Si bien Roque puede entrenar a más de un ave a la vez, Susana sigue entrenando de a un ave.
- `avesCapacesDeVolar(unosKms)`: Es la colección de aves entrenadas por Roque que son capaces de volar la cantidad de `unosKms`

Escribir los tests correspondientes en el cual roque entrena a pepita, pepon y pipa.

Parte 3: *Instituto de Entrenamiento*
Modelar un instituto que posee entrenadores. El instituto puede contratar o desvincular a uno o varios entrenadores.

Hacer que el instituto entienda los siguientes mensajes:

- `contratar(entrenador)`: Agrega un entrenador al instrituto.
- `prescindir(entrenador)`: Remueve un entrenador del instituto.
- `entrenamientoGeneral()`: Todos los entrenadores entrean a su/s ave/s.
- `buenAmbiente()`: El instituto tiene buen ambiente si todos sus entrenadores están contentos. Susana está contenta si su ave puede volar más de 5 kilómetros. Roque está contento si entrena a más de un ave y a menos de 8.
- `mejoresEstudiantes()`: Es el conjunto de aves recomendados por los entrenadores. Cada entrenador elige su mejor estudiante. El mejor estudiante de Susana es la única ave que entrena. Roque elige como mejor estudiante a cualquiera que pueda volar más de 10 kilometros.

Escribir los tests necesarios probando al instituto con Roque y Susana como entrenadores. Susana entrena a Pepita y Roque a Pipa y Pipón. Dibujar el grafo de referencias.

Ejercicio 2: Casa de Pepe y Julián (A.K.A. Cuentas Bancarias

Modificar el objeto *casa* del ejercicio de las cuentas bancarias de las guías anteriores para agregar las siguientes funcionalidades.

Parte 1: *Recordar todas las cosas compradas*

La casa debe entender el mensaje `compras()` que devuelve una colección con todas las cosas compradas.

Revisar y refactorizar[1] en caso de ser necesario la implementación de la casa para que no quede información duplicada y no realice *precálculos*. El estado de la casa debe tener sólo dos atributos: La cuenta para los gastos y la colección de cosas compradas. Prestar atención a:

- Dónde y cuándo se calcula la sumatoria de los gastos de la casa.
- Cómo obtener la últicа cosa comprada al contestar si viene de equiparse.
- Cómo saber si un elemento comprado es comida o no.

Hacer un test para cada una de esas funcionalidades. Incluir también el test sobre las preguntas `esDerrochona()` y `esBacan()`

Parte 2: *Más preguntas a la casa*
Hacer que la casa pueda responder:

- Compra más cara: Devuelve la cosa comprada de mayor valor.

[1] refactorizar: cambiar el código que cumple una funcionalidad para que sea de mejor calidad

- Electrodomésticos comprados: Todas las cosas compradas que son electrodomésticos.
- Mala Época: Si todas las compras son comida.
- Qué falta comprar: Recibe una lista de cosas que se desa comprar y devuelve las cosas de esa lista que aún no se han comprado. **¡Ojo con esto!** Es una pregunta. No se pide que se compren.
- Falta comida: Si se compraron menos de 2 cosas que son comida.

Agregar los tests para cada caso.

Parte 3: *Deseos*
Romina, la novia de Pepe, tiene ciertas expectativas sobre la casa de su novio. Ella espera que Pepe y Julián hayan comprado las cosas que desea.

Modelar a Romina para que entienda mensajes que cumplan la siguiente funcionalidad:

- Desear: Agregar una cosa a su colección de deseos.
- Olvidar: Deja de desear algo.
- Pedir una cosa: Cuando Romina pide una cosa, lo que sucede es que la casa la compra si es que no la tenía. En caso contrario no hace nada. **¡Ojo con esto!** Es una orden. La cosa queda en la lista de las cosas compradas de la casa. No hay que devolverla
- Deseos no satisfechos: Son las cosas que ella desea y que la casa aún no ha comprado. **¡Ojo con esto!** No repetir código.
- Deseos satisfechos: Son las cosas que ella desea y que la casa ya compró.
- Cosas para donar: Son las cosas que tiene la casa y que ella no desea.

Caso de prueba: La casa de Julián y Pepe compra la heladera y el paquete de fideos, Romina desea el paquete de fideos, la plancha y la cama. Pide la cama y la tira de asado. Pensar bien todos los *asserts* necesarios y armar el grafo de referencias.

Ejercicio 3: Golosinas
A Mariano le gusta mucho las golosinas. Juliana opina que exagera en el consumo de las mismas. Por eso Mariano decidió programar un modelo en objetos que le permitiera analizar su conducta.

Parte 1: *Golosinas*

De cada golosina interesa el precio, el sabor, su peso en gramos y si contiene gluten. Además, cada vez que una golosina recibe un mordisco se reduce la cantidad de gramos que posee.

- Bombón: Vale 5 pesos y pesa inicialmente 15 gramos. Su gusto es frutilla. Es libre de gluten. Cuando recibe un mordisco, pierde un 20 % de su cantidad en gramos + 1 gramo extra (debido a que parte del relleno cremoso se pierde al caer al suelo).
- Alfajor: Vale 12 pesos y pesa inicialmente 300 gramos. Su gusto es chocolate. No es libre de gluten. Cuando recibe un mordisco, pierde el 20 % de su cantidad de gramos.
- Caramelo: Vale 1 peso y pesa inicialmente 5 gramos. Su gusto es frutilla. Es libre de gluten. Cuando recibe un mordisco, pierde el 1 gramo.
- Chupetín: Vale 2 pesos y pesa inicialmente 7 gramos. Su gusto es naranja. Es libre de gluten. Cuando recibe un mordisco, pierde el 10 % de su peso. A excepción de que su peso actual sea menor a 2 gramos. En ese caso, no pierde nada.
- Oblea: Vale 5 pesos y pesa inicialmente 250 gramos. Su gusto es vainilla. No es libre de gluten. Al recibir un mordisco pierde una cantidad de gramos dependiendo de su peso actual. Si es mayor a 70 gramos pierde el 50 % de su peso, si no, el 25 %.
- Chocolatín, el peso inicial es desconocido, lo asigna el usuario. El precio es de $0,50 por cada gramo de peso inicial. Pierde 2 gramos por mordisco. No es libre de gluten. **¡Ojo con esto!** El precio es según el peso inicial, no debe cambiar con los mordiscos.
- Golosina bañada: Se arma a partir de una golosina de base. El peso inicial es el de la golosina de base más 4 gramos que es lo que pesa el bañado. El precio es el de la golosina de base más 2 pesos. El gusto es el de la golosina de base. De la misma manera, es "libre de gluten"si lo es su golosina base. Con cada mordisco se da un mordisco a la golosina de base. Además, en el primer mordisco pierde 2 gramos de bañado, y en el segundo los otros dos.
- Pastilla tutti-frutti. Pesa inicialmente 5 gramos. En cada mordisco cambia el sabor, pasa de frutilla a chocolate, de ahí a naranja, de ahí vuelve a frutilla. La pastilla puede ser libre de gluten o no (se configura). Si es libre de gluten el precio es $7 o $10 si no lo es.

Parte 2: *Mariano*

Crear el objeto mariano con la capacidad de comprar golosinas:

- Entender el mensaje `comprar(unaGolosina)`, que guarda una golosina en la bolsa de golosinas compradas.
- Entender el mensaje `desechar(unaGolosina)`, que desecha la golosina escogida de la bolsa de golosinas.

- Entender el mensaje `probarGolosinas()`, que le da un mordisco a todas las golosinas dentro de la bolsa de golosinas compradas.
- Entender el mensaje `hayGolosinaSinTACC()`, que indica si hay al menos una golosina sin gluten en la bolsa de golosinas compradas.
- Entender el mensaje `preciosCuidados()`, que indica si todas las golosinas compradas tienen un precio menor o igual a 10 pesos.
- Entender el mensaje `golosinaDeSabor(unSabor)`, que devuelve la primer golosina que encuentra en la bolsa del sabor escogido.
- Entender el mensaje `golosinasDeSabor(unSabor)`, que devuelve las golosinas que encuentre dentro de la bolsa del sabor escogido.
- Entender el mensaje `sabores()`, que devuelve los sabores de las golosinas de la bolsa.
- Entender el mensaje `golosinaMasCara()`, que delvuelve la golosina mas cara en la bolsa de golosinas compradas.
- Juliana suele pedirle golosinas a Mariano y a menudo se enoja cuando él no tiene alguna de las que le gustan. Crear el mensaje en Mariano `golosinasFaltantes(golosinasDeseadas)`, que devuelve de entre las golosinas recibidas por parámetro, aquellas que Mariano no compró.
- Marcelo, en cambio, tiene antojo por comer golosinas de algunos sabores específicos. Crear el mensaje en mariano `saboresFaltantes(sabores)`, que devuelve aquellos sabores recibidos por parámetros que no están cubiertos por ninguna golosina que compró Mariano.

Ejercicio 4: Camión de transporte

Una empresa de transporte quiere administrar mejor las cargas que lleva un camión. Para eso requiere un sistema que le permita planificar que cosas debe llevar el camión sin sobrepasar su capacidad. Por otro lado, las cosas que transporta tienen un nivel de peligrosidad. Este nivel es usado para impedir que cosas que superen cierto nivel de peligrosidad circulen en determinadas rutas.

Parte 1: *Camión*
Se pide que el camión entienda los siguientes mensajes:

- `cargar(cosa)`.
- `descargar(cosa)`.
- `pesoTotal()`: Es la suma del peso del camión vacío (tara) y su carga. la tara del camión es de 1 tonelada.

- `excedidoDePeso()`: Si el peso total es superior al peso máximo. El cual es de 2.5 toneladas.
- `objetosPeligrosos(n)`: Todos los objetos cargados que superan el nivel de peligrosidad `n`.
- `objetosMasPeligrososQue(cosa)`: Todos los objetos cargados que son más peligrosos que la `cosa`.
- `puedeCircularEnRuta(nivelMaximoPeligrosidad)` Puede circular si ningúna cosa que transporta supera el `nivelMaximoPeligrosidad`.

Parte 2: *Cosas*
De las cosas que puede transportar el camión nos interesa el peso y la peligrosidad:

- *Knight Rider*: pesa 500 kilos y su nivel de peligrosidad es 10
- *Bumblebee*: pesa 800 kilos y su nivel de peligrosidad es 15 si está transformado en auto o 30 si está como robot.
- *Paquete de ladrillos*: cada ladrillo pesa 2 kilos, la cantidad de ladrillos que tiene puede variar. La peligrosidad es 2
- *Arena a granel*: El peso es variable. La peligrosidad es 1
- *Batería antiaérea* : El peso es 300 kilos si está con los misiles o 200 en otro caso. En cuanto a la peligrosidad es 100 si está con los misiles y 0 en otro caso.
- *Contenedor portuario*: Un contenedor puede tener otras cosas adentro. El peso es 100 + la suma de todas las cosas que esté adentro. Es tan peligroso como el objeto más peligroso que contiene. Si está vacío es 0.
- *Residuos radioactivos*: El peso es variable y su peligrosidad es 200
- *Embalaje de seguridad*: Es una cobertura que envuelve a cualquier otra cosa. El peso es el peso de la cosa que tenga adentro. El nivel de peligrosidad es la mitad del nivel de peligrosidad de lo que envuelve.

Incorporar las cosas al camión. Testear y probar generando distintas combinaciones. Dibujar el grafo de referencias para un camión que lleva un paquete de 30 ladrillos, un contenedor portuario que en su interior están knigth rider, bumblebee y una batería antiaérea cargada con misiles. También lleva residuos radiactivos pero dentro de un embalaje de seguridad.

Ejercicio 5: Quiniela reducida
En un sistema para casas de apuestas, se quiere resolver el pago de las apuestas de una versión reducida de la Quiniela.

Parte 1: *Sorteo*
En esta versión se sortean 5 números. Cada uno de estos números ocupa una posición en el resultado final, (del 1ro. al 5to.). El sistema no debe realizar el sorteo, simplemente se va a ingresar el resultado del mismo.

Modelar el sorteo y determinar una manera de asignar los números ganadores.

Parte 2: *Apuestas y sorteo*
Hay distintas formas de apostar. De cada apuesta nos va a interesar el valor apostado por el jugador, si es ganadora o no del sorteo, el monto del premio en caso de ser ganadora y el nombre del apostador.

Para saber cuanto es el premio, muchas apuestas requieren de saber el total recaudado por el sorteo, que se calcula sumando todos los valores apostados.

Nota: No hay dos jugadores que apuesten de la misma forma. Un jugador sí puede apostar más de una vez

Hacer que a un sorteo se le puedan agregar apuestas y que responda los siguientes mensajes:

- `apuestasGanadoras()`
- `apuestasPerdedoras()`
- `totalEnPremios()`: Es la suma de todos los premios que se tienen que pagar.
- `ganadores()`: Son los nombres de los ganadores.
- `beneficio()`: Es la diferencia entre el total recaudado y el total en premios.

Probar el sorteo con un objeto apuesta para test, el cual entiende los mensajes que necesita el sorteo para funcionar.

Parte 3: *Apuestas*

Salvo que se diga lo contrario, el dinero apostado es a elección del jugador.

A continuación se detallan todas las apuestas soportadas por el sistema. Programarlas e ir testéandolas con el sorteo.

- A la cabeza: El jugador apuesta que su número quedará en la primera posición. El premio es de 70 veces el valor de lo apostado con un máximo del 10 % del total recaudado por el torneo.
- A los primeros 2: El jugador apuesta que su número va a salir primero o segundo. El premio es de 30 veces el monto apostado con un máximo del 5 % del total recaudado.

- A cualquiera: El jugador apuesta que su número va a salir sorteado en cualquier posición. El premio es el doble del valor apostado con un máximo del 1 % del total del valor recaudado.
- Un rango a la cabeza: El jugador apuesta que el número que sale primero está en un rango determinado por un valor mínimo y uno máximo. Por ejemplo, que el número que saldrá a la cabeza está entre 101 y 108. El premio que paga es de 55 veces el valor apostado / (máximo - mínimo). El valor no puede superar el 10 % del total recaudado.
- En orden: El jugador apuesta a acertar los 5 números sorteados en el orden exacto en que salieron. Esta apuesta vale $500 y paga el 40 % del total recaudado.
- Aproximacion a la cabeza: El jugador apuesta a un número. Si sale a la cabeza el premio es de 50 veces el valor de la apuesta. Si le erró a la cabeza por uno es 40 veces el valor. Si le erró por dos es 30 veces. Si le erró por tres es 20 veces, Si le erró por 4 es 10 veces. Si le erró por 5 o más ya no tiene premio. En cualquier caso, el máximo del premio es un 6 % del total recaudado. Fórmula: $((50-(cabeza-apuesta).abs()*10)*dineroApostado).min(0{,}06*totalRecaudado).max(0)$

Ejercicio 6: Juegos con Personajes - continuación

Extender y/o analizar las siguientes situaciones sobre el ejercicio Juegos con Personajes de la guía 3.

Parte 1: *Modificaciones sobre la felicidad de Mario*

Cambio de requerimientos: Mario está feliz cuando recolectó al menos 50 unidades o si **alguno** de los elementos que recolectó es igual o superior a 10 metros de altura

- ¿Es necesario modificar las variables de mario para este cambio?
- Plantee un modelo de mario que utilice colecciones y otro que no, para resolver ese requerimiento.
- Analice en cada modelo que sucede cuando la altura de las cosas que encontró mario se modifican al recibir mensajes. Tome como ejemplo la siguiente secuencia y determine si mario es feliz o no luego de su ejecución:

```
mario.encontrar(tipa)
mario.encontrar(aurora)
tipa.recibirTrabajo()
```

Parte 2: *Nuevo Elemento: Estatua de oro*

La estatua de oro otorga un valor variable, que comienza en 60. Cuando recibe trabajo, aumenta el valor en 5. Cuando recibe un ataque, el valor disminuye una cantidad igual a la potencia del ataque. La altura es 5.

Programar el nuevo elemento. ¿Es necesario modificar alguno de los objetos existentes?

Analizar la felicidad de mario luego de la siguiente secuencia:

```
mario.recolectar(estatuaDeOro)
estatuaDeOro.recibirAtaque(20)
```

Según el momento en que se pregunte el valor que otorga el elemento la respuesta varía. Es por eso que el requerimiento debe ser claro sobre ese asunto. La solución cambia según el camino elegido. Analice ambos casos y programe las soluciones:

- Si el valor recolectado depende del momento en que se recolectó el elemento:
  - ¿Es necesario realizar algún cambio a la solución actual?
  - ¿Se puede resolver simplemente manteniendo una colección con los elementos encontrados?
- Si el valor recolectado no depende del momento en que se encontró el elemento, sino que se debe calcular cuando se le pregunta la felicidad a mario
  - ¿Es necesario realizar algún cambio al modelo?
  - ¿Se puede resolver manteniendo un contador del valor en mario?

Parte 3: *Nuevo personaje: fulano*

Agregue otro personaje llamado Fulano. Cada vez que encuentra un elemento, además de darle trabajo al elemento encontrado, se lo da a todos los elementos que alguna vez encontró.

Programe a fulano y piense en lo siguiente:

- ¿Qué sucede si fulano se encuentra dos veces el mismo elemento?
- Analice el polimorfismo de fulano con los otros objetos del juego.

Parte 4: *Análisis de referencias*

Habiendo dejado en claro que para el cálculo de la felicidad de mario:

- La altura de los elementos se preguntan en el momento que se calcula la felicidad
- El valor que otorga cada elemento se pregunta en el momento en que el valor es recolectado

y que si Fulano se le aparece dos veces el mismo elemento, solo envía una vez sola el mensaje `recibirTrabajo()` a ese elemento.

Escriba un test para la siguiente secuencia y dibuje el grafo de referencias luego de la ejecución del mismo. Piense los *asserts* necesarios:

Luisa tiene como personaje activo a mario. Le aparece primero aurora, después el castillo y tipa. El arma de loki es la ballesta. Luego el personaje activo de luisa es loki. a loki se le aparece la estatua de oro. El personaje activo de luisa es fulano. A luisa se le aparece la estatua de oro, tipa, aurora, y tipa de nuevo.